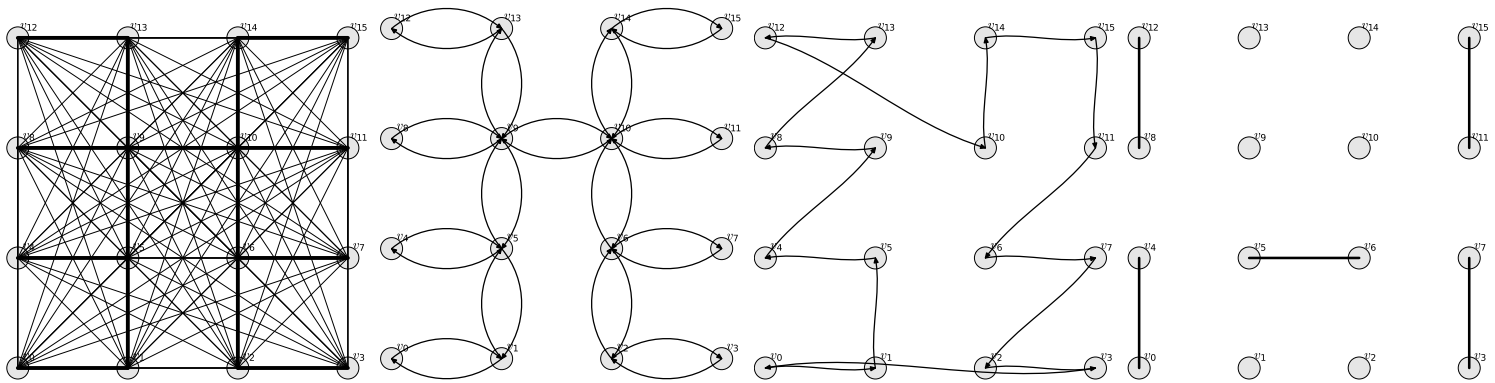
Алгоритм Кристофидеса

Н. Н. Кузюрин С. А. Фомин

8 марта 2023 г.



Определение

Эйлеровым путем *в графе называется произвольный путь, проходящий через каждое ребро графа в точности один раз.*

Определение

Замкнутый эйлеров путь называется ***эйлеровым обходом*** *или* ***эйлеровым циклом****.*

Определение

Эйлеров граф *— граф, в котором существует эйлеров обход.*

Теорема

Эйлеров цикл существует $\Leftrightarrow$ граф связный и все его вершины четной степени.

```
def EulerCircuitExists(G):
    for v in G.nodes():      #Проверка необходимых
        if G.degree(v) % 2 != 0: # условий существования
            return False
    return True
```

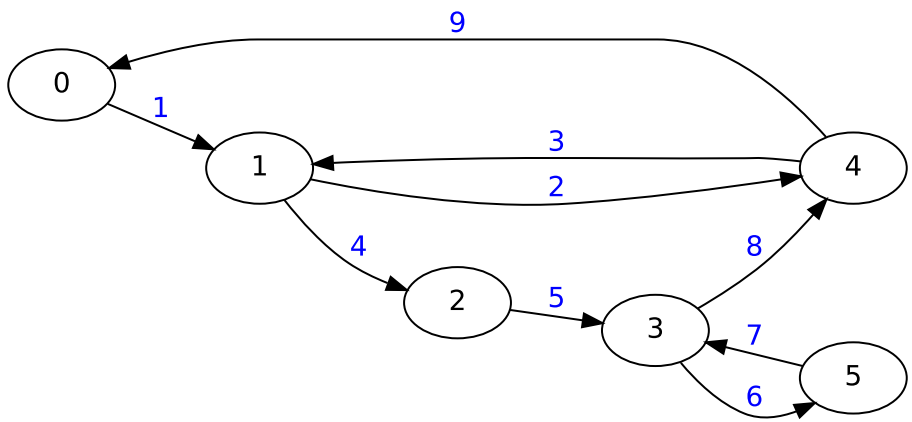
Доказательство.

Достаточность $\leftarrow$ алгоритм нахождения эйлерова пути.

Необходимость:
если вершина v в эйлеровом обходе k раз
$\Rightarrow$ степень этой вершины в графе $2k$. $\square$

Алгоритм нахождения Эйлерова цикла

```
def EulerCircuit(G):# next(iter(G))
    EP = [ 0 ]
    while G.number_of_edges() > 0:
        for i, v in enumerate(EP):
            # куда бы добавить цикл?
            if G.degree(EP[i]) > 0:
                break
        while G.degree(v) > 0:
            #пока не в «тупике»
            w = next(G.neighbors(v))
            G.remove_edge(v, w)
            i += 1
            EP.insert(i, w)
            v = w  # и повторяем все с w
    return EP
```




```
[0] + (0 : <12340>) -> [0, 1, 2, 3, 4, 0]
[0, 1, 2, 3, 4, 0] + (1 : <41>) -> [0, 1, 4, 1, 2, 3, 4, 0]
[0, 1, 4, 1, 2, 3, 4, 0] + (3 : <53>) -> [0, 1, 4, 1, 2, 3, 5, 3, 4, 0]
[0, 1, 4, 1, 2, 3, 5, 3, 4, 0]
```

Задача

«Коммивояжер», «TSP[a]». Заданы неориентированный граф из n вершин-городов, и $d_{ij} \equiv d(v_i, v_j)$ — положительные целые расстояния между городами.
Чему равна наименьшая возможная длина гамильтонова цикла (кольцевого маршрута, проходящего по одному разу через все города)? т. е. нужно найти минимально возможное значение суммы

$$\min_{p \in \begin{pmatrix} 1 & 2 & \dots & n \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}} \sum_{i=1}^{n-1} d_{p_i, p_{i+1}} + d_{p_n, p_1}, \tag{1}$$

где минимум берется по всем перестановкам p чисел $1, \dots, n$.

[a] В англоязычной литературе — Traveling Salesman Problem.

Определение

Задача коммивояжера (задача 1 «TSP») называется ***метрической****, если для матрицы расстояний выполнено неравенство треугольника:*

$$\forall i, j, k \;\; d_{ik} \leq d_{ij} + d_{jk}.$$

Определение

Задача коммивояжера (задача 1 «TSP») называется ***метрической****, если для матрицы расстояний выполнено неравенство треугольника:*

$$\forall i, j, k \;\; d_{ik} \leq d_{ij} + d_{jk}.$$

В метрической TSP — граф должен быть полным!

Определение

Задача коммивояжера (задача 1 «TSP») называется ***метрической****, если для матрицы расстояний выполнено неравенство треугольника:*

$$\forall i, j, k \;\; d_{ik} \leq d_{ij} + d_{jk}.$$

В метрической TSP — граф должен быть полным!

Лемма

Метрическая задача коммивояжера $\mathcal{NP}$-полна.

Определение

Задача коммивояжера (задача 1 «TSP») называется ***метрической****, если для матрицы расстояний выполнено неравенство треугольника:*

$$\forall i, j, k \;\; d_{ik} \leq d_{ij} + d_{jk}.$$

В метрической TSP — граф должен быть полным!

Лемма

Метрическая задача коммивояжера $\mathcal{NP}$-полна.

Доказательство.

1. «Гамильтонов цикл» (единичные веса) — $\mathcal{NP}$-полна.

Определение

Задача коммивояжера (задача 1 «TSP») называется ***метрической****, если для матрицы расстояний выполнено неравенство треугольника:*

$$\forall i, j, k \;\; d_{ik} \leq d_{ij} + d_{jk}.$$

В метрической TSP — граф должен быть полным!

Лемма

Метрическая задача коммивояжера $\mathcal{NP}$-полна.

Доказательство.

1. «Гамильтонов цикл» (единичные веса) — $\mathcal{NP}$-полна.
2. Дополним до полного графа ребрами с весом 2.

Определение

Задача коммивояжера (задача 1 «TSP») называется ***метрической****, если для матрицы расстояний выполнено неравенство треугольника:*

$$\forall i, j, k \;\; d_{ik} \leq d_{ij} + d_{jk}.$$

В метрической TSP — граф должен быть полным!

Лемма

Метрическая задача коммивояжера $\mathcal{NP}$-полна.

Доказательство.

1. «Гамильтонов цикл» (единичные веса) — $\mathcal{NP}$-полна.
2. Дополним до полного графа ребрами с весом 2.
3. Граф с весами 1 и 2 — метрический.

Определение

Задача коммивояжера (задача 1 «TSP») называется ***метрической****, если для матрицы расстояний выполнено неравенство треугольника:*

$$\forall i, j, k \;\; d_{ik} \leq d_{ij} + d_{jk}.$$

В метрической TSP — граф должен быть полным!

Лемма

Метрическая задача коммивояжера $\mathcal{NP}$-полна.

Доказательство.

1. «Гамильтонов цикл» (единичные веса) — $\mathcal{NP}$-полна.
2. Дополним до полного графа ребрами с весом 2.
3. Граф с весами 1 и 2 — метрический.
4. Если оптимум не включает в себя ребра с весом 2 — то это гамильтонов цикл для исходного графа.

□

Задача

«Минимальное остовное дерево» (Minimum Spanning Tree)
Задан связный неориентированный граф $G = (V, E)$, *где* $V = \{v_1, \ldots, v_n\}$*—множество вершин,* $|V| = n$, *E—множество ребер между ними, и весовая функция* $w : E \to Z+$[a]
Требуется найти наименьший возможный вес остовного дерева, т. е.

$$\min \sum_{(i,j)\in T} w(v_i, v_j), \tag{2}$$

где минимум берется по всем остовным деревьям на n вершинах (по всем множествам T из $(n-1)$ *дуг, связывающим все n вершин в единую сеть).*

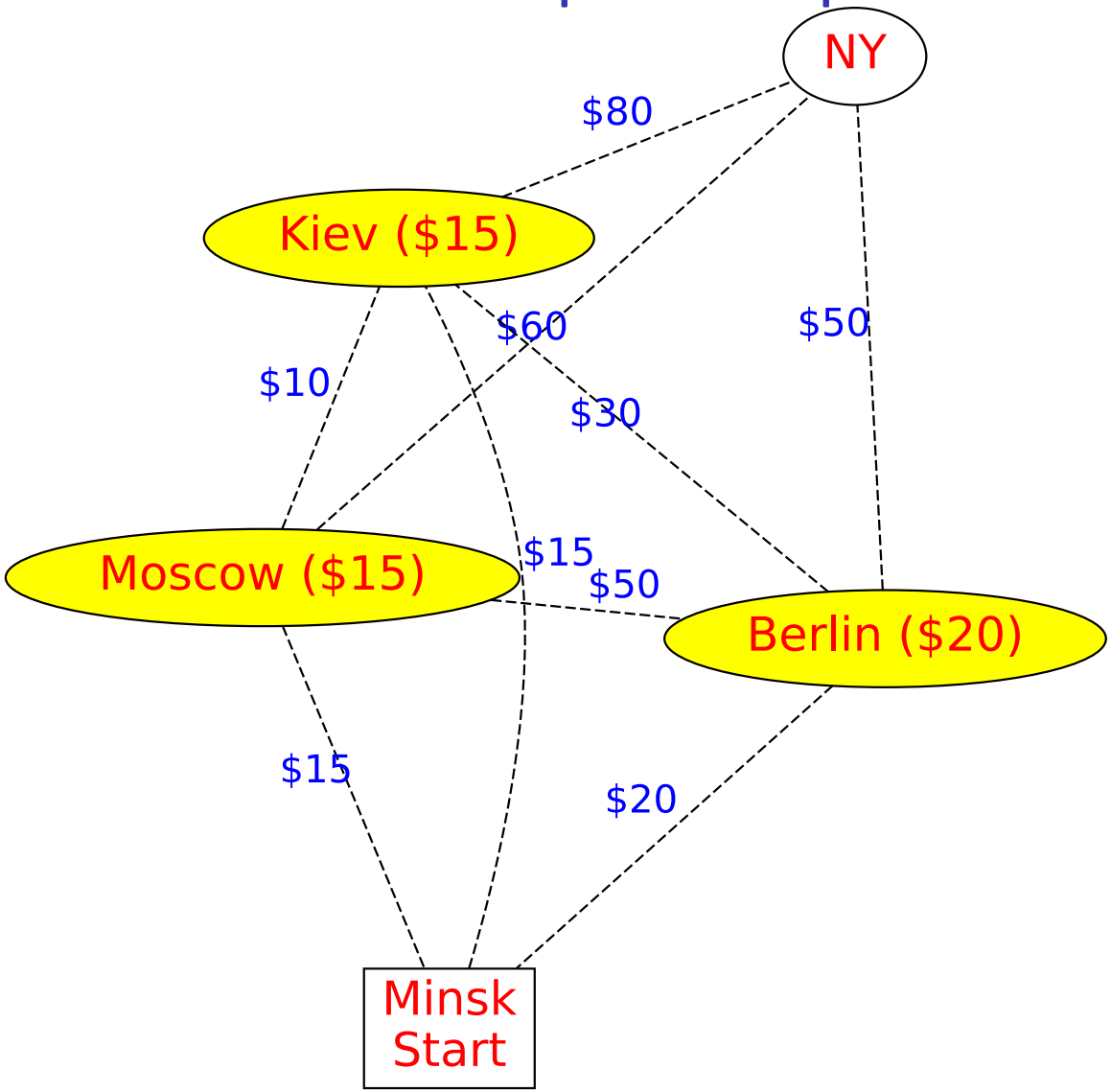
---

[a] Можно вводить положительные целые веса на ребрах, как $w_{ij} \equiv w(v_i, v_j)$.

Алгоритм Прима

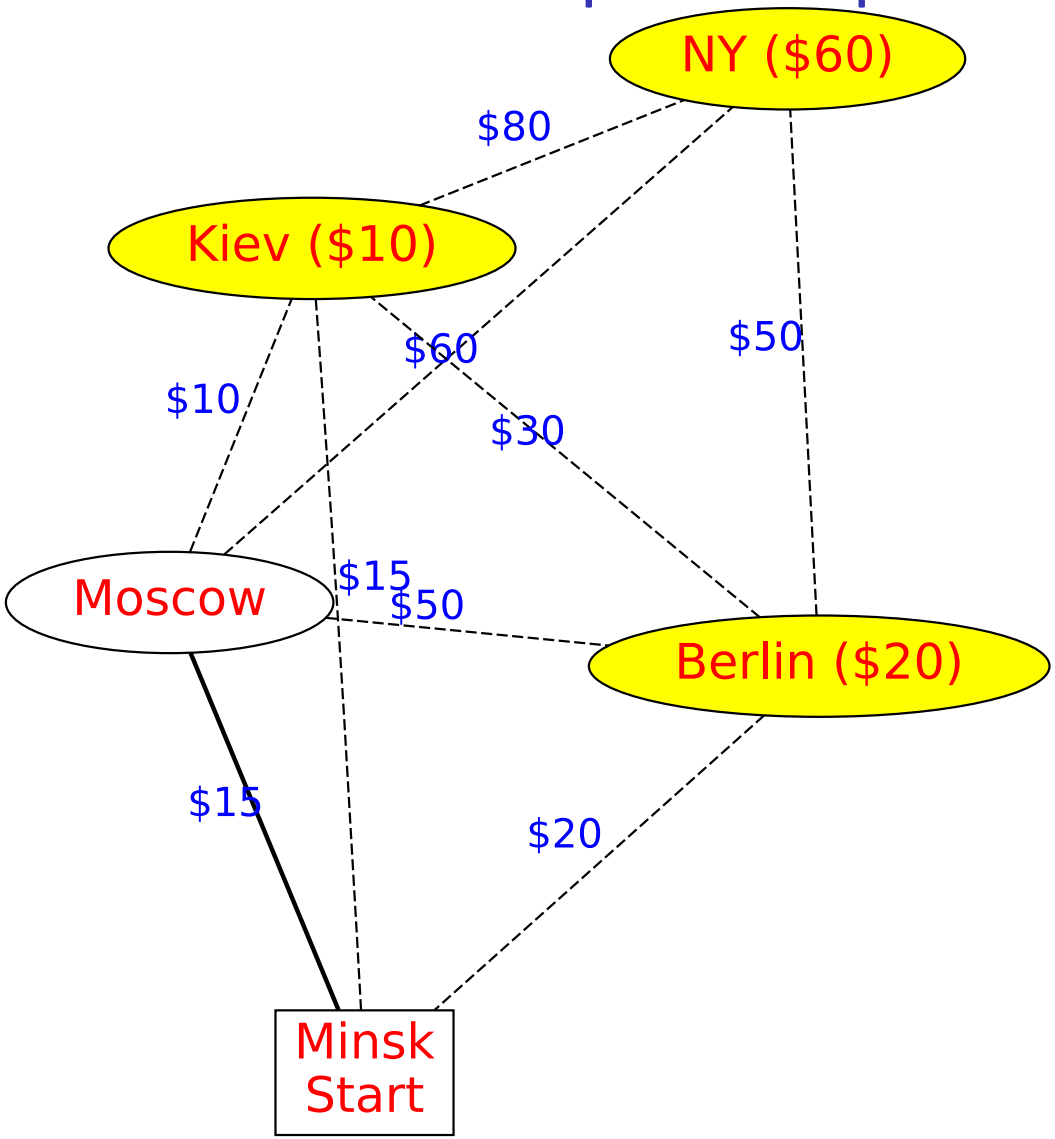
```
def MST_Prim(G, s):
    MST = {} #хэш (узел: предшественник)
    ToVisit = {s: 0}  # граничащие с MST, (узел: стоимость)
    Ancestor = {s: s} #хэш: вершины из которых включают другие
    while ToVisit:  # пока есть непосещенные вершины
        v = argmin(ToVisit) # ближайшая достижимая вершина
        MST[v] = Ancestor[v] # запоминаем, откуда пришли
        del ToVisit[v]
        del Ancestor[v] # больше не посещать
        for w in G.neighbors(v): # для всех соседей вершины v
            if w not in MST:  # которые еще не в MST
                # обновляем стоимость включения в MST
                if (w not in ToVisit
                      or G[v][w]["weight"] < ToVisit[w]):
                    ToVisit[w] = G[v][w]["weight"]
                    Ancestor[w] = v
    return MST
```

Выполнение алгоритма Прима



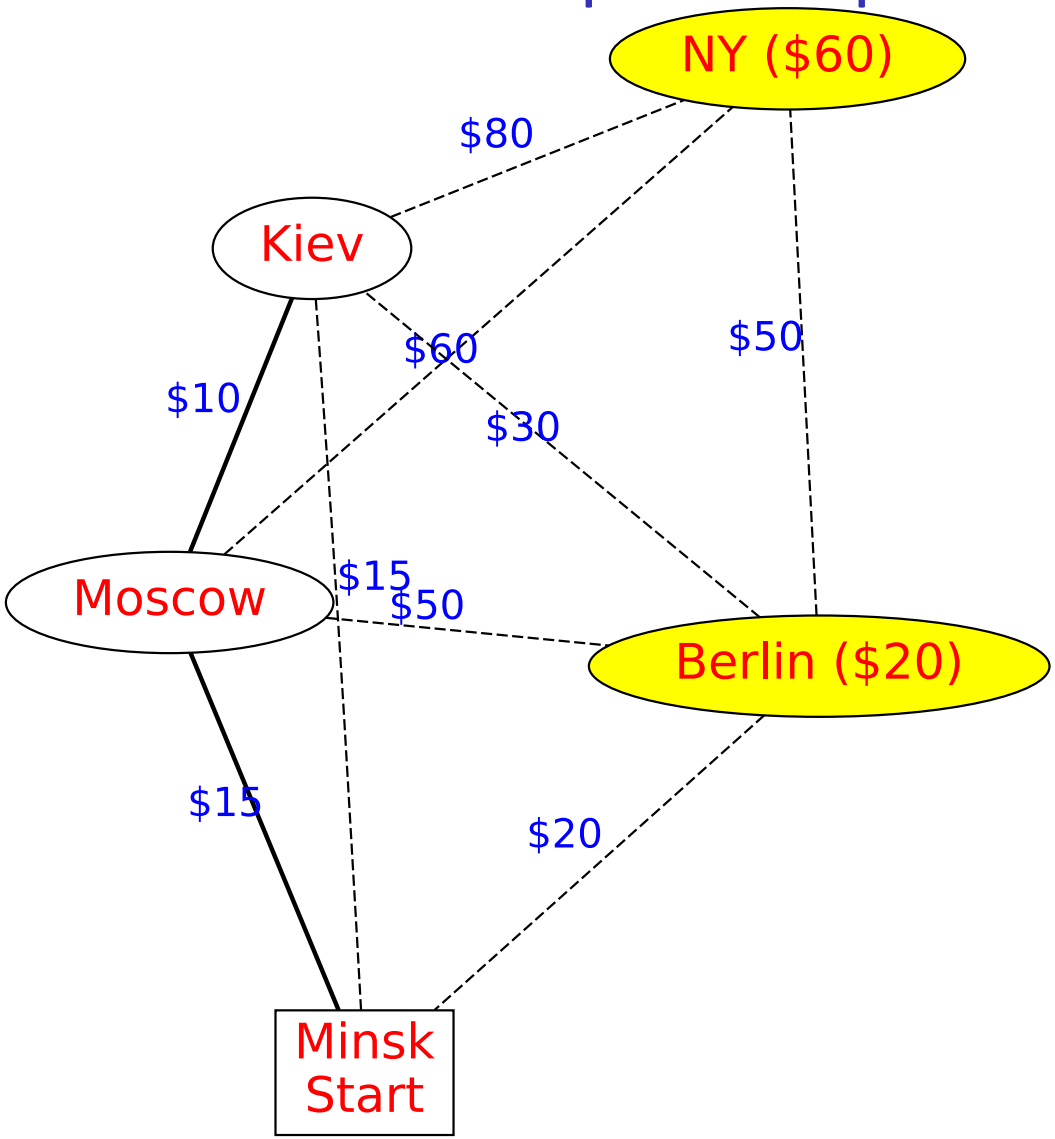
Итерация № 1 Стоимость MST: 0

Выполнение алгоритма Прима



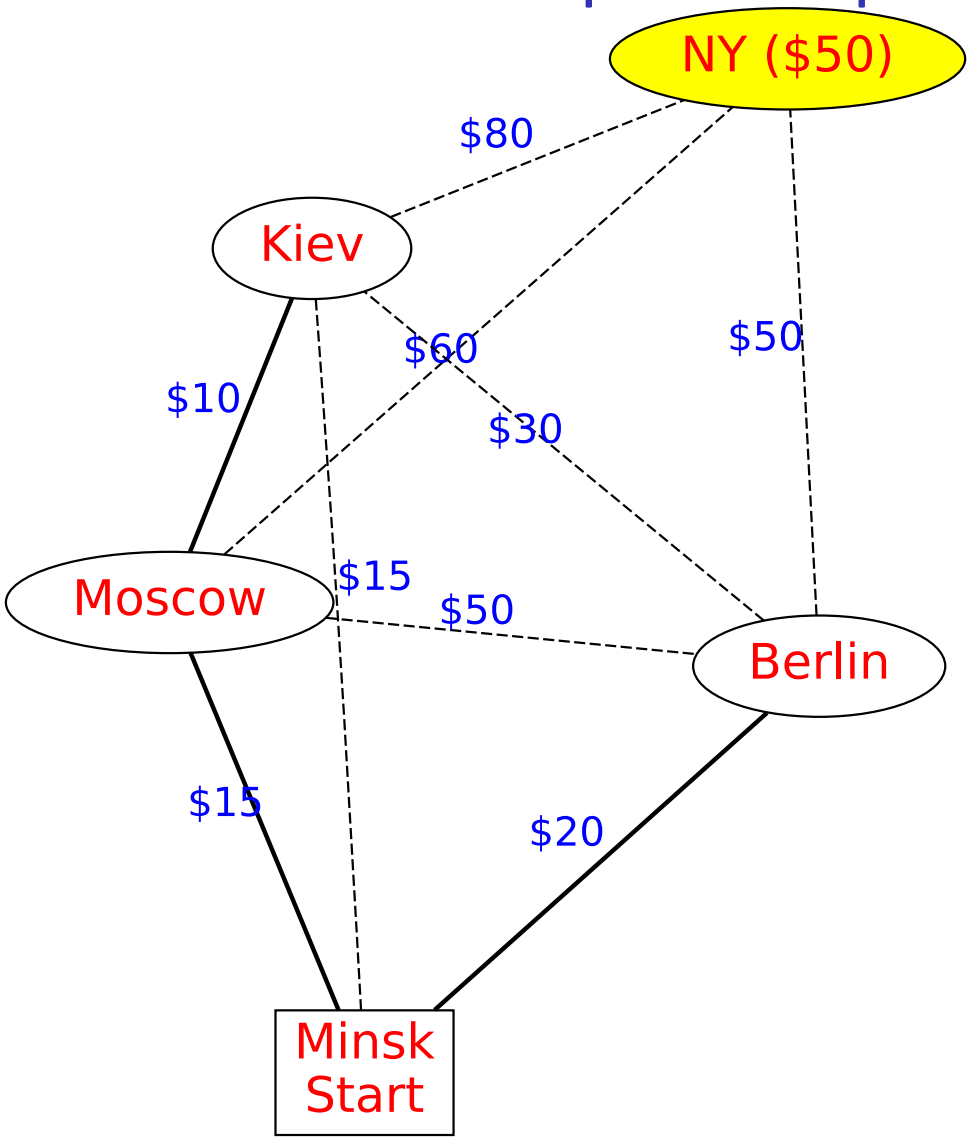
Итерация № 2 Стоимость MST: 15

Выполнение алгоритма Прима



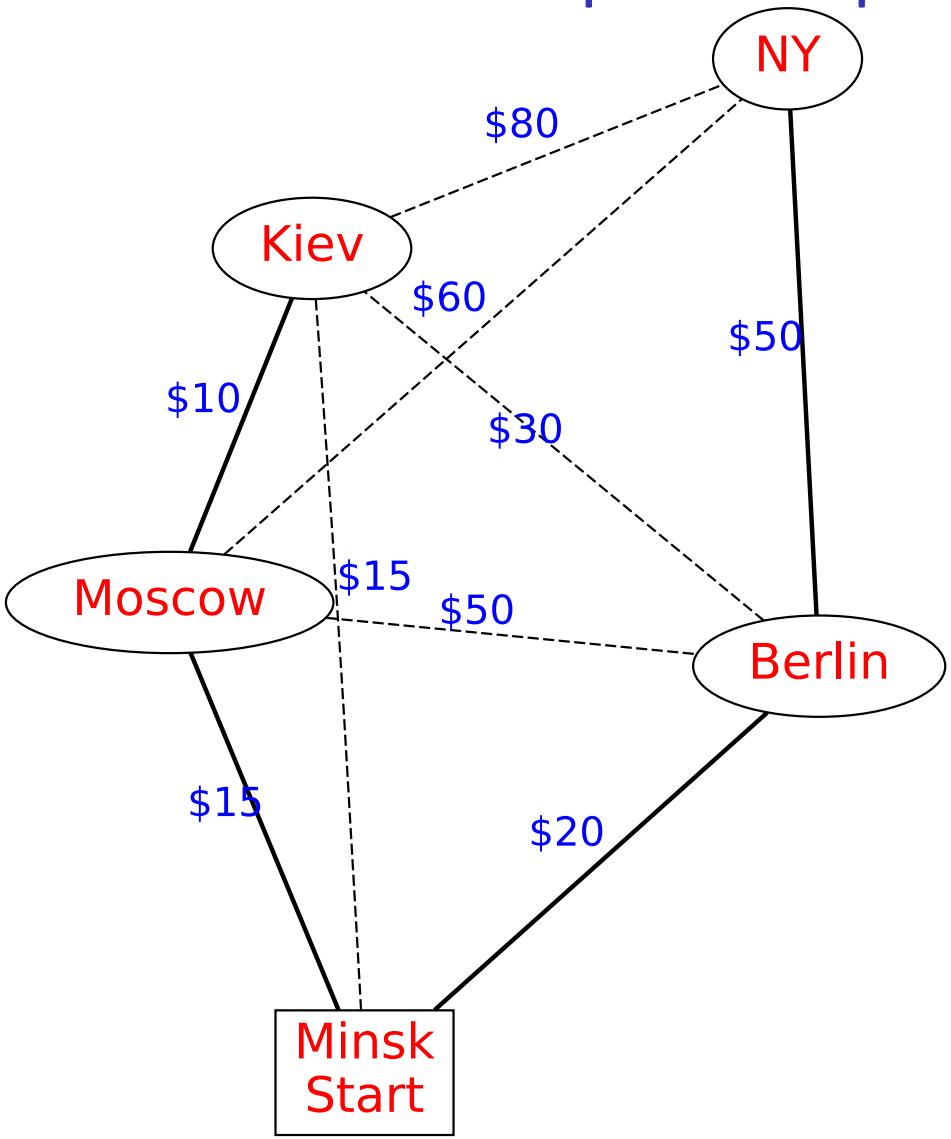
Итерация № 3 Стоимость MST: 25

Выполнение алгоритма Прима



Итерация № 4 Стоимость MST: 45

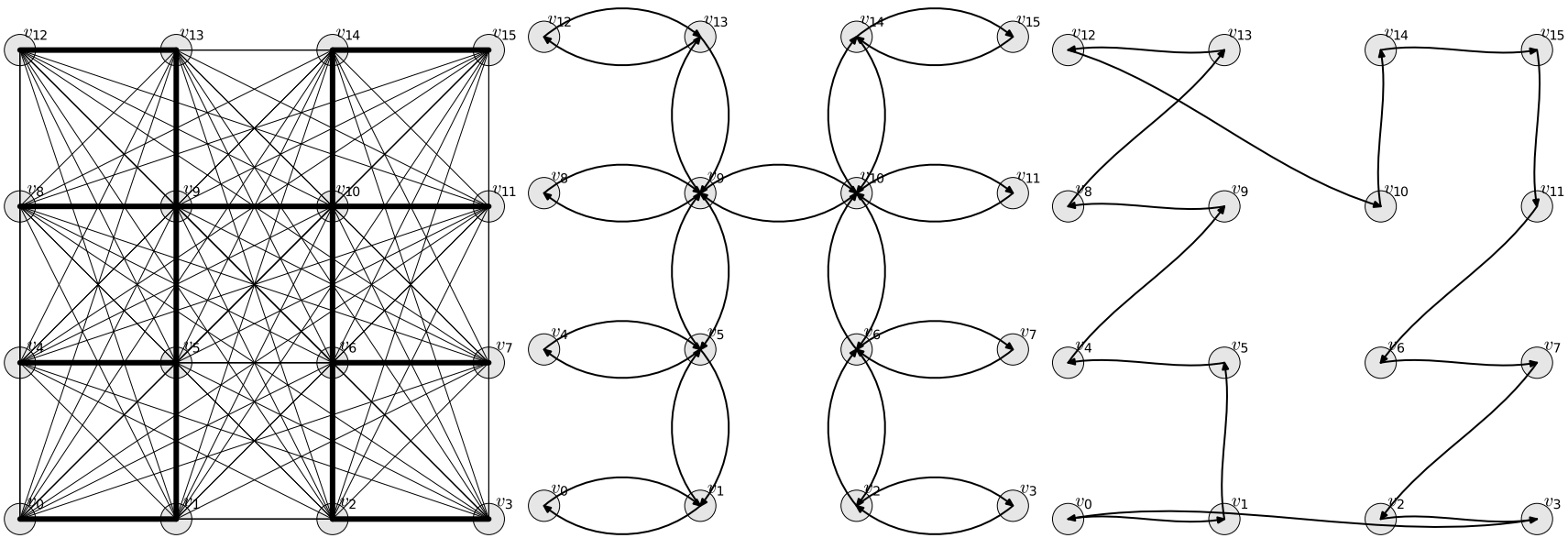
Выполнение алгоритма Прима



Итерация № 5 Стоимость MST: 95

Приближенный алгоритм-1 для метрической «TSP»

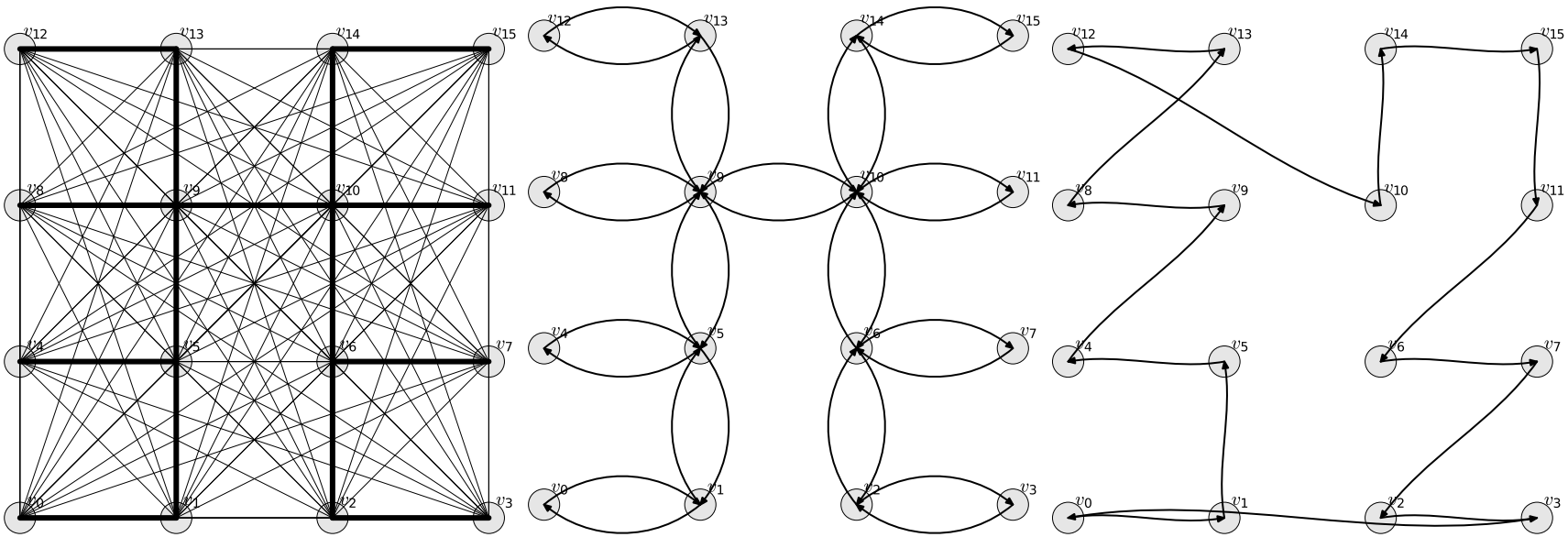
1. Найти минимальное остовное дерево T с матрицей весов $[d_{ij}]$.
2. Построить эйлеров граф G и эйлеров обход в G, продублировав все ребра дерева T.
3. Из эйлерова маршрута (обхода) гамильтонов цикл строится путем последовательного вычеркивания вершин, встретившихся ранее.



Алгоритм-1 2-приближенный

Доказательство.

1. Кратчайший гамильтонов цикл не меньше MST.
2. Длина эйлерова маршрута в G равна двум MST.
3. Неравенство треугольника $\Rightarrow$ полученный гамильтонов цикл не длиннее эйлерова обхода $\Rightarrow$ не длиннее двух MST $\Rightarrow$ ...



Определение

Паросочетание[a] *— подмножество ребер графа, такое, что никакие два ребра из этого подмножества не инцидентны какой-либо одной вершине.*

---

[a] *Matching* в англоязычной литературе.

Определение

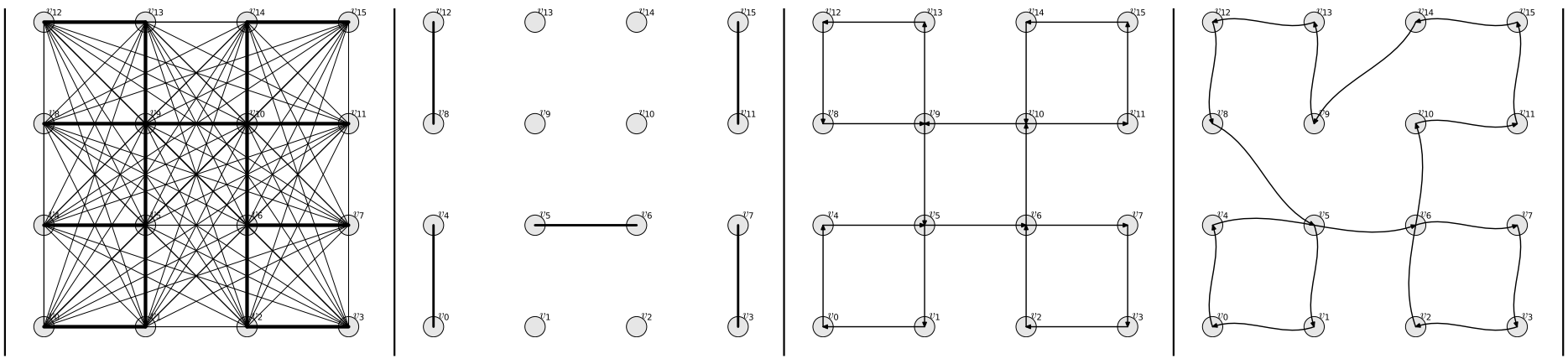
Совершенное паросочетание[a] *— паросочетание, покрывающее все вершины графа.*

---

[a] *Perfect matching* в англоязычной литературе.

Алгоритм Кристофидеса для метрической «TSP»

1. Найти MST T.
2. $N(T)$ — вершины нечетной степени в T
3. M — кратчайшее совершенное паросочетание с множеством вершин $N(T)$.
4. G_E эйлеров граф с вершинами $\{v_1, \ldots, v_n\}$ и ребрами $T \cup M$.
5. Найти эйлеров обход P_E в G_E.
6. Построить гамильтонов цикл P_H из P_E последовательным вычеркиванием посещенных вершин.

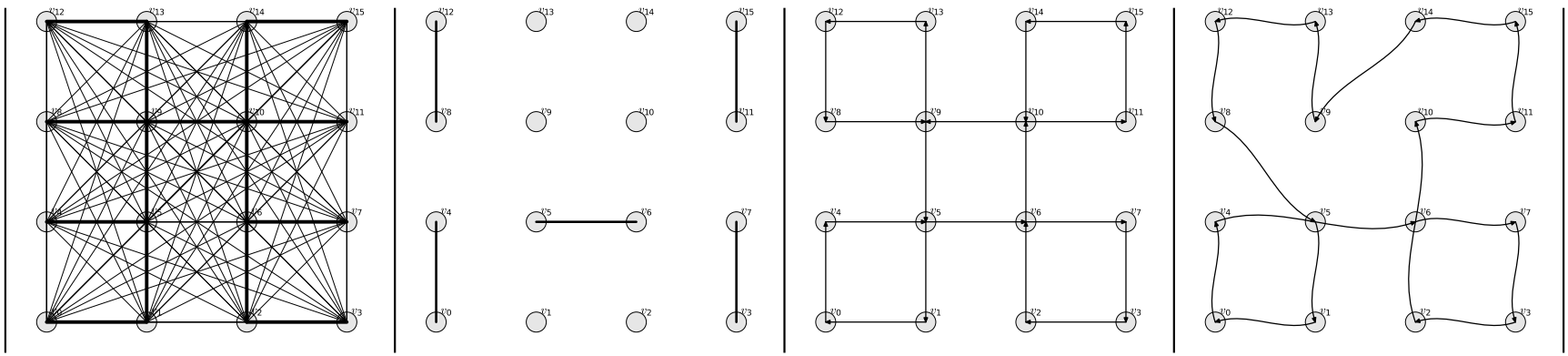


Алгоритм Кристофидеса — $\frac{3}{2}$-приближенный

G_E — эйлеров и связен. P_H — результат. P_H^* — оптимум. c — стоимость

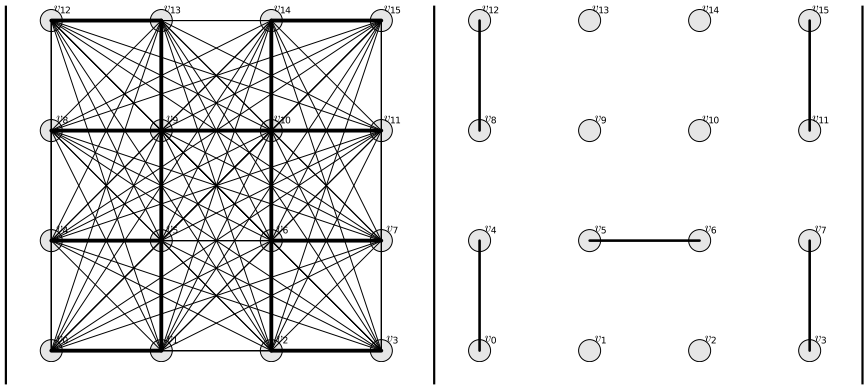
$$c(P_H) \leq c(P_E) = c(G_E) = c(T) + c(M).$$

$$c(T) \leq c(P_H^*) \leq c(P_H).$$



Алгоритм Кристофидеса — $\frac{3}{2}$-приближенный

$i_1, i_2, \ldots, i_{2m}$ — вершины нечетной степени в T (в порядке P_H^*).



Рассмотрим паросочетания:

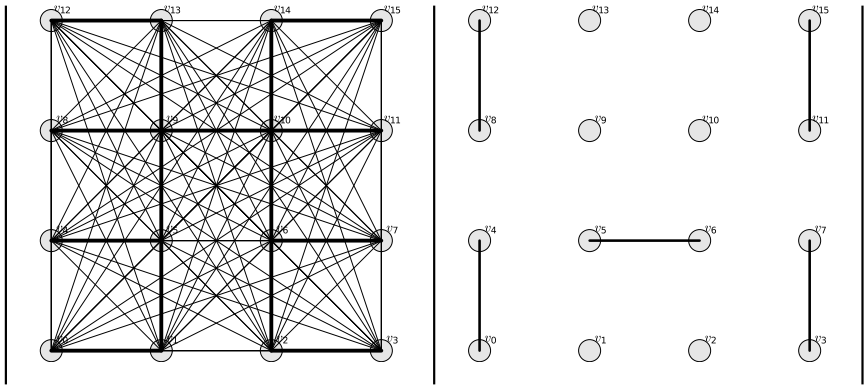
$$M_1 = \{i_1, i_2\},\ \{i_3, i_4\},\ \ \ldots,\ \{i_{2m-1}, i_{2m}\},$$
$$M_2 = \{i_2, i_3\},\ \{i_4, i_5\},\ \ \ldots,\ \{i_{2m}, i_1\}.$$

Из неравенства треугольника $\Rightarrow c(P_H^*) \geq c(M_1) + c(M_2)$.
M — кратчайшее $\Rightarrow c(M_1) \geq c(M), c(M_2) \geq c(M)$.

$$c(P_H^*) \geq 2c(M) \ \Rightarrow\ c(M) \leq \frac{1}{2}c(P_H^*).$$

Алгоритм Кристофидеса — $\frac{3}{2}$-приближенный

$i_1, i_2, \ldots, i_{2m}$ — вершины нечетной степени в T (в порядке P_H^*).



Рассмотрим паросочетания:

$$M_1 = \{i_1, i_2\},\ \{i_3, i_4\},\ \ \ldots,\ \{i_{2m-1}, i_{2m}\},$$
$$M_2 = \{i_2, i_3\},\ \{i_4, i_5\},\ \ \ldots,\ \{i_{2m}, i_1\}.$$

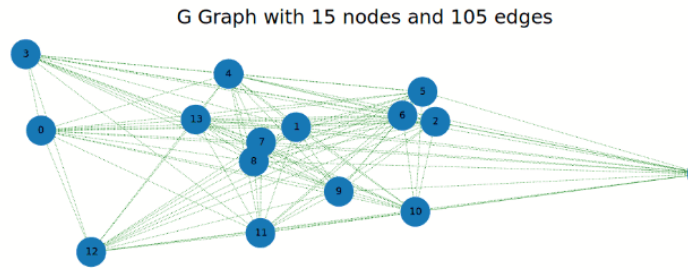
Из неравенства треугольника $\Rightarrow c(P_H^*) \geq c(M_1) + c(M_2)$.
M — кратчайшее $\Rightarrow c(M_1) \geq c(M), c(M_2) \geq c(M)$.

$$c(P_H^*) \geq 2c(M) \ \Rightarrow\ c(M) \leq \frac{1}{2}c(P_H^*).$$

$$c(P_H) \leq c(T) + c(M) \leq c(P_H^*) + \frac{1}{2}c(P_H^*) = \frac{3}{2}c(P_H^*).$$

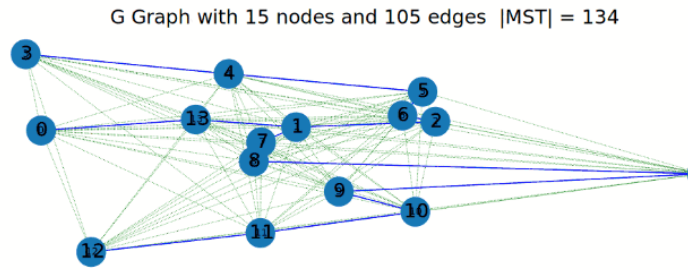
Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                    init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```



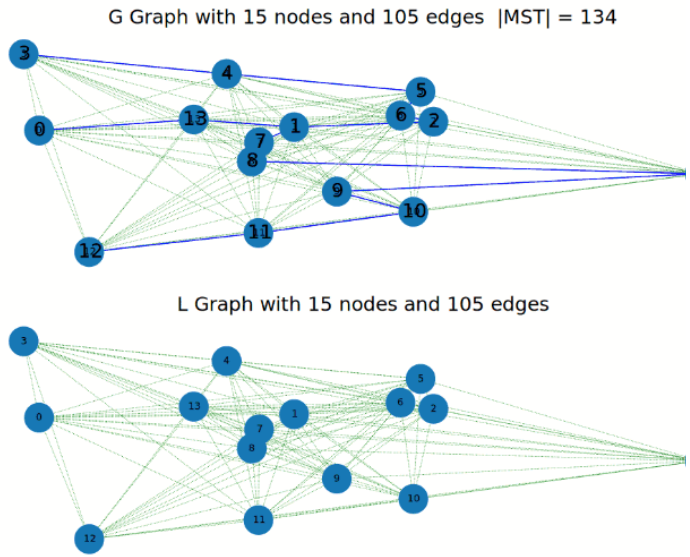
Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```



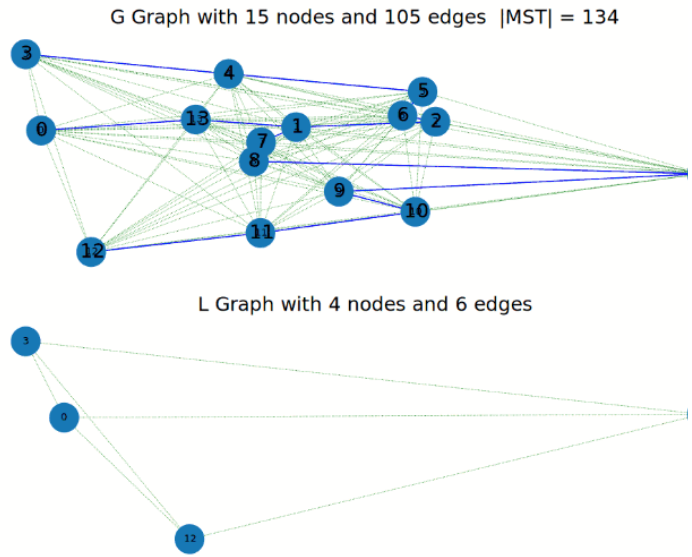
Выполнение алгоритма Кристофидеса

```
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```



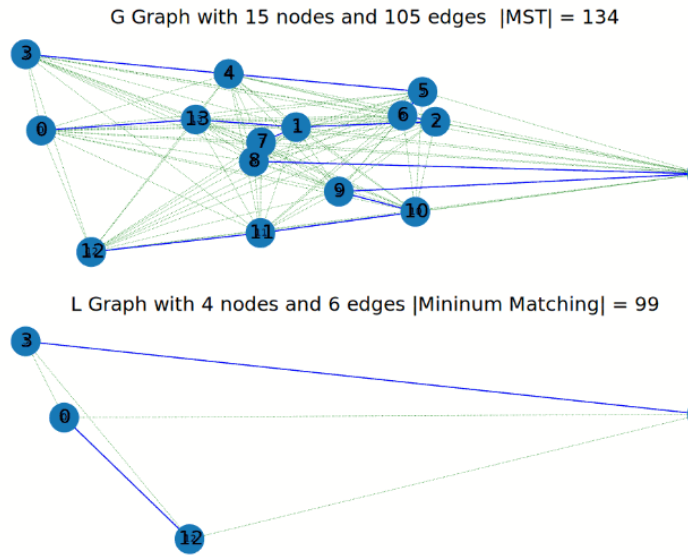
Выполнение алгоритма Кристофидеса

```
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```



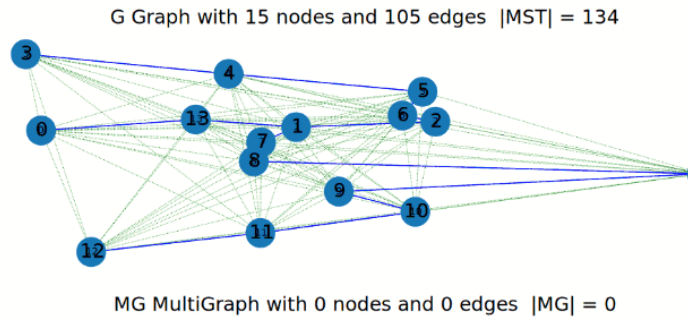
Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```



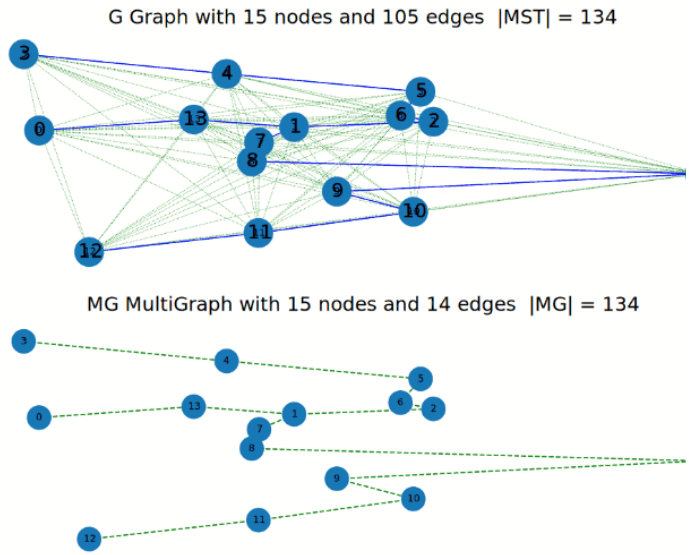
Выполнение алгоритма Кристофидеса

```
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```



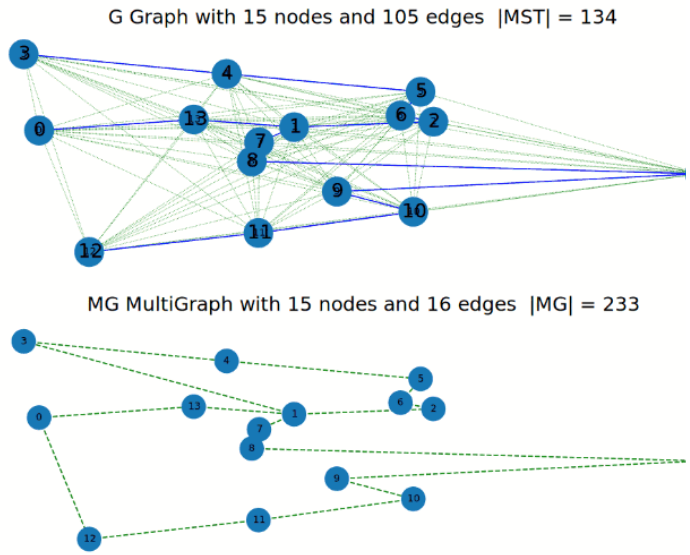
Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```



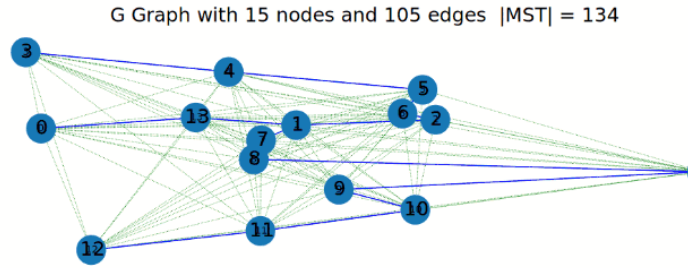
Выполнение алгоритма Кристофидеса

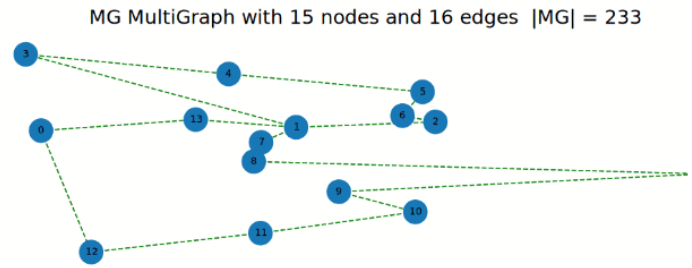
```
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```



Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

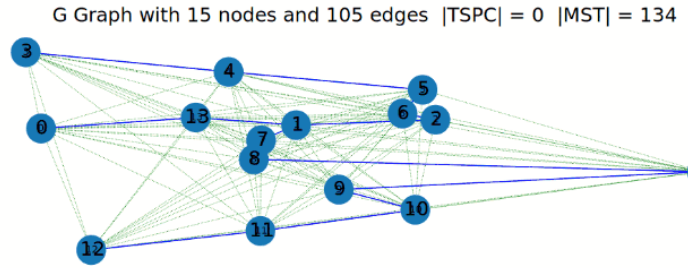


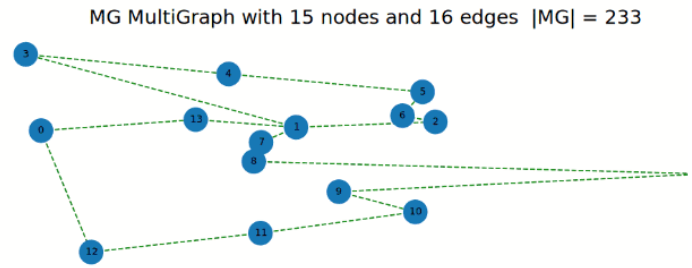


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour														
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                    init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

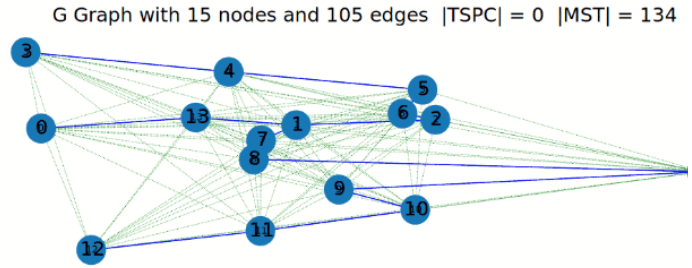


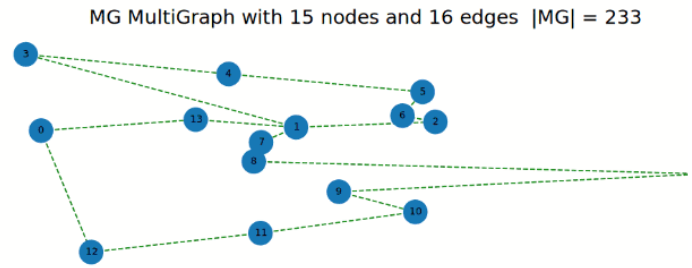


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour														
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

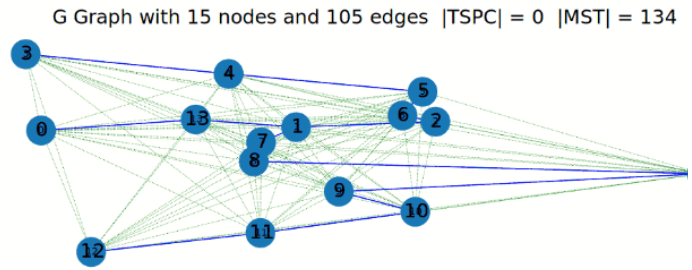


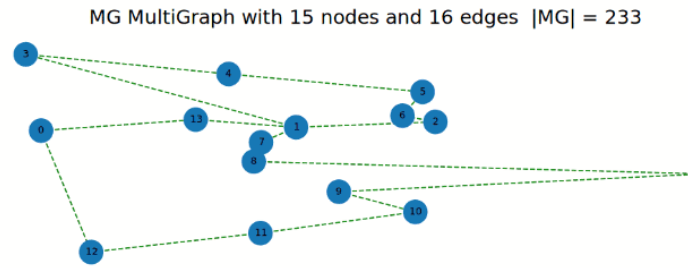


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour														
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

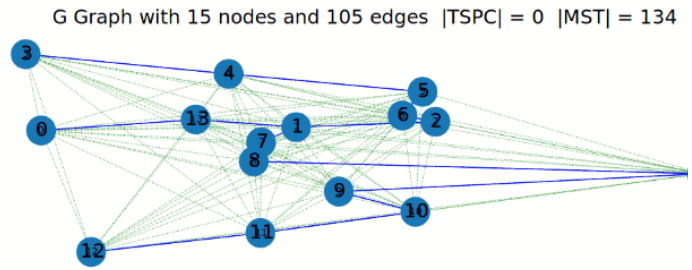


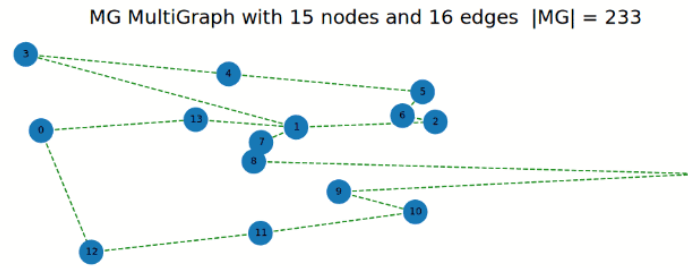


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0													
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

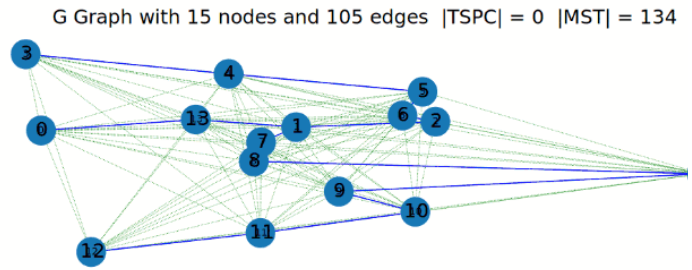


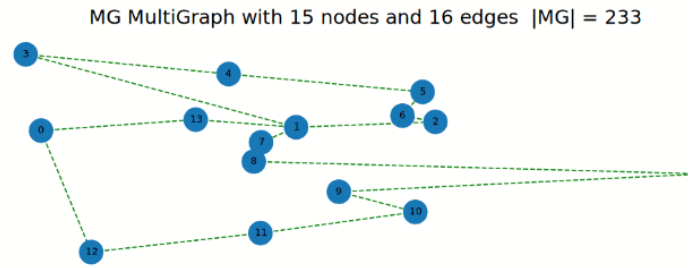


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0													
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

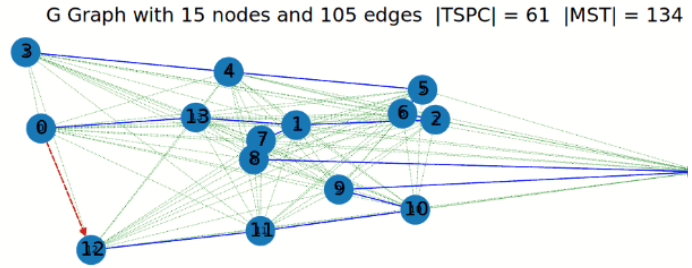


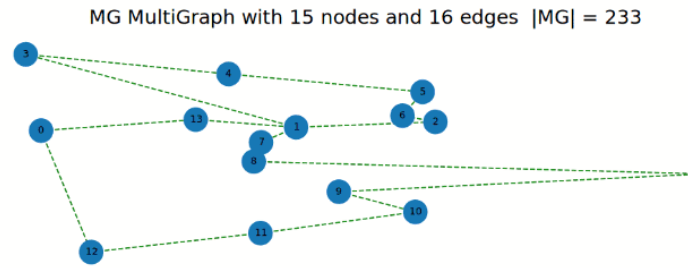


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0													
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                    init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

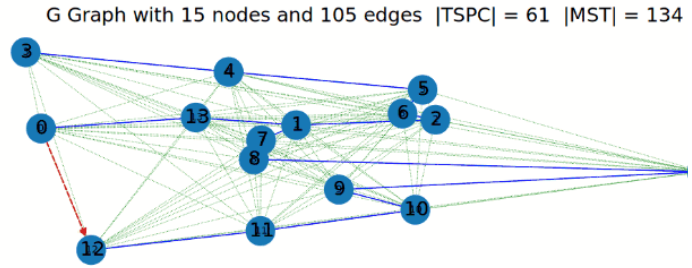


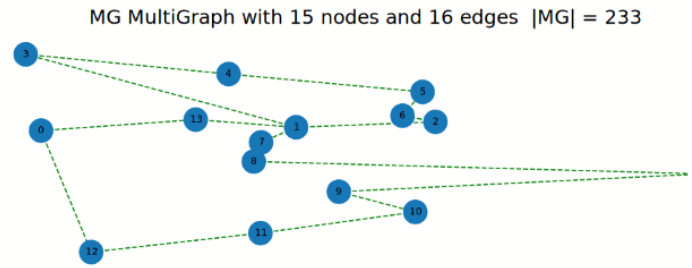


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12												
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

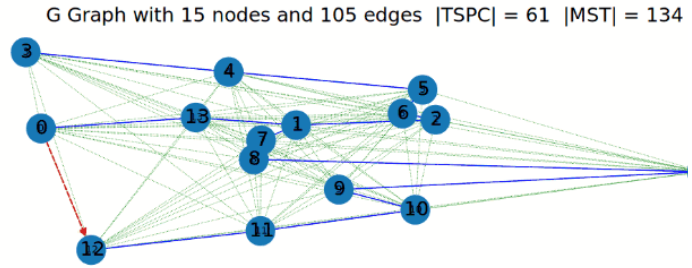


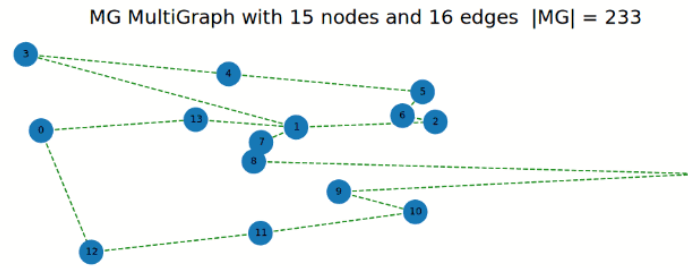


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12												
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

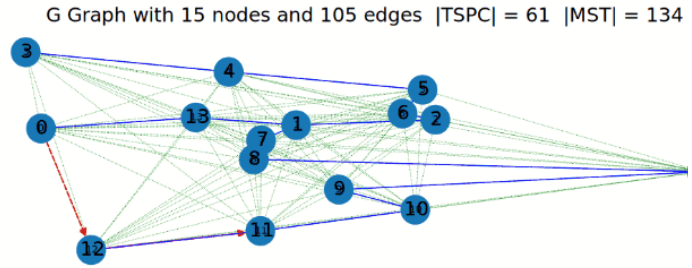


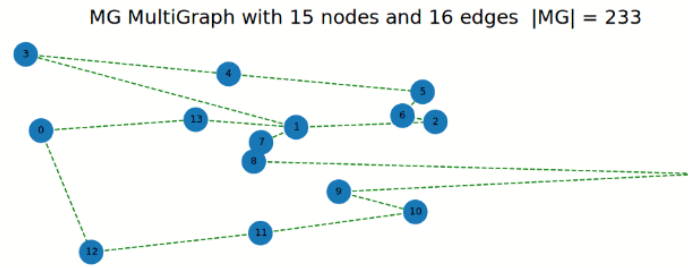


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12												
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

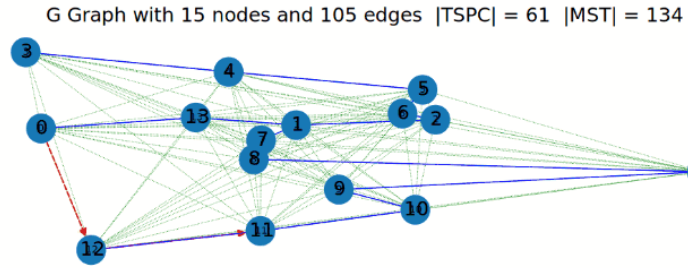


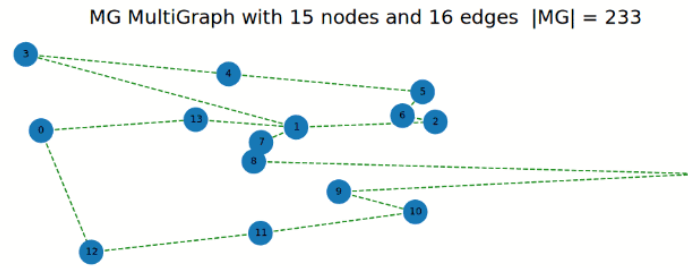


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11											
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

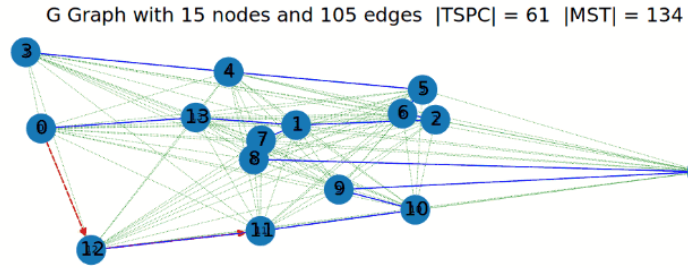


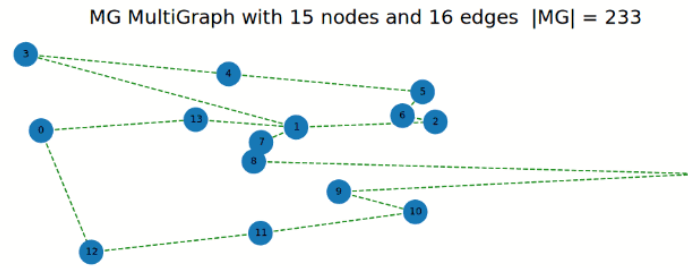


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11											
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

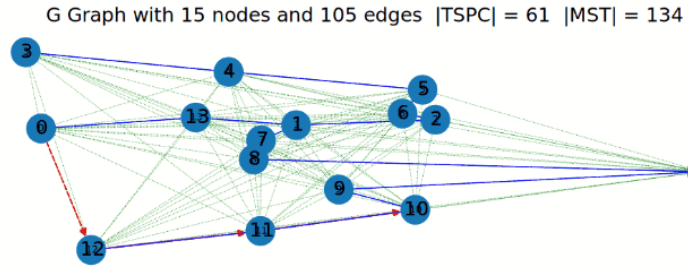


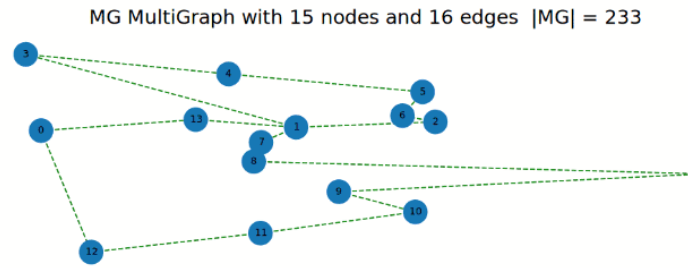


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11											
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                    init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

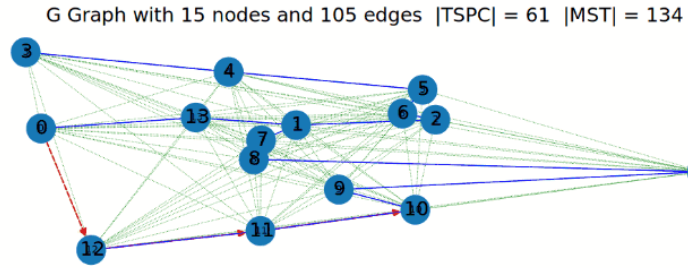


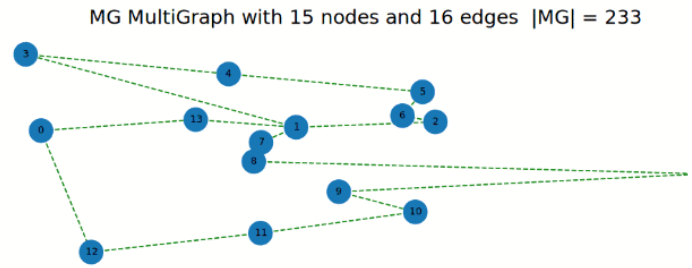


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10										
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

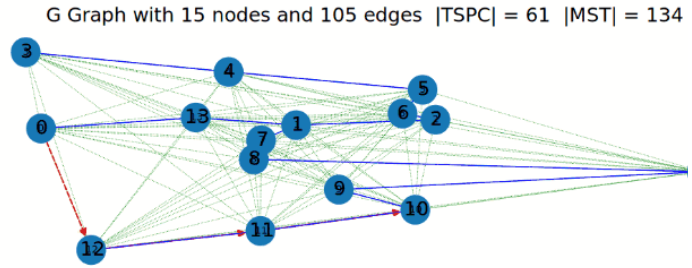


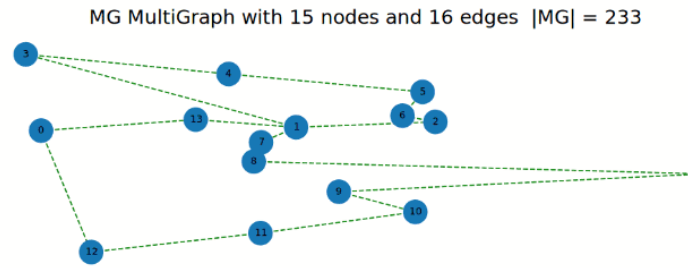


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10										
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

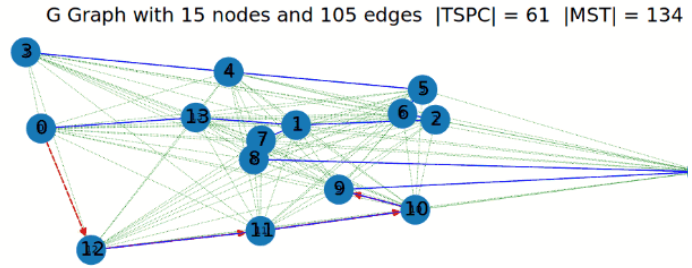


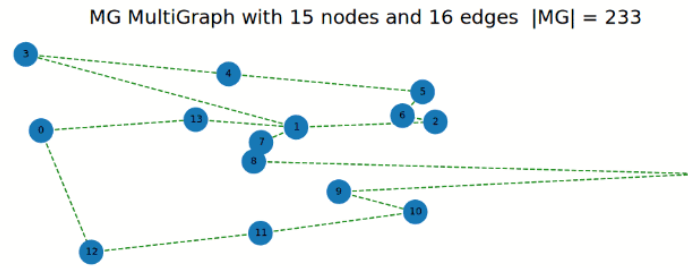


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10										
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

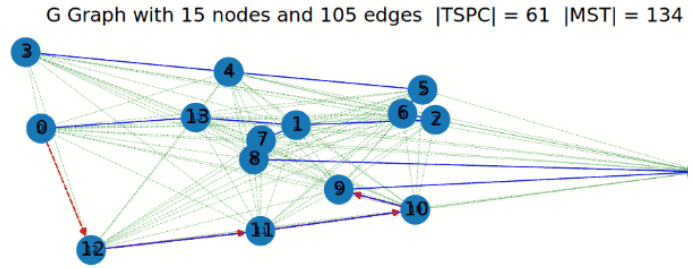


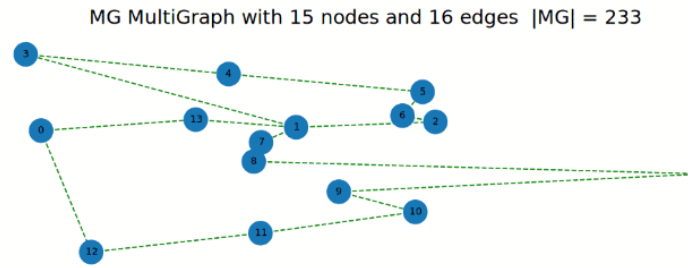


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10	9									
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                    init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

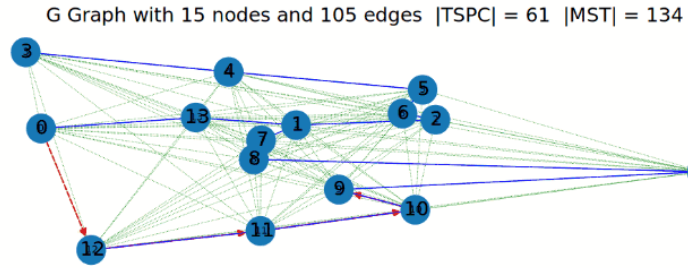


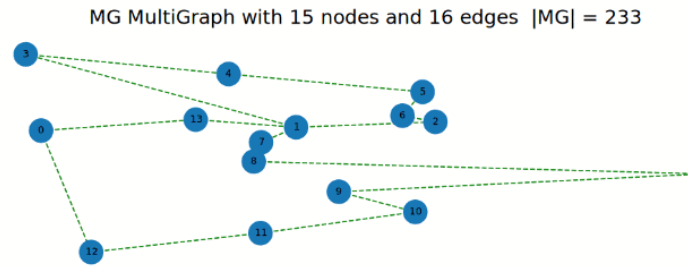


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10	9									
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

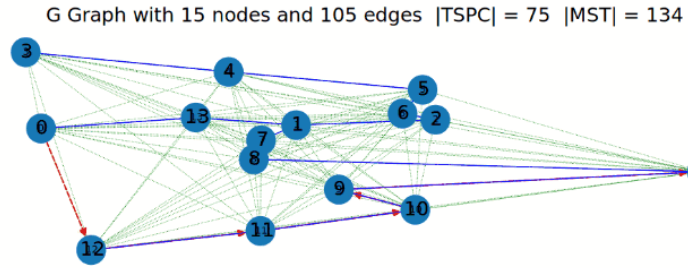


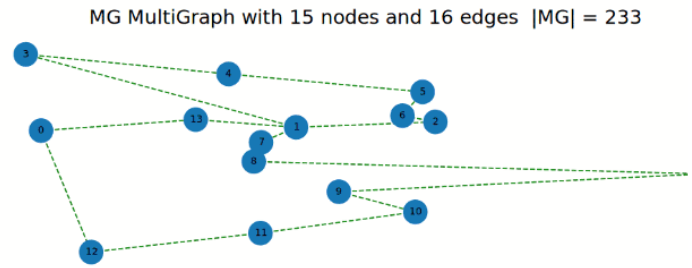


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10	9									
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

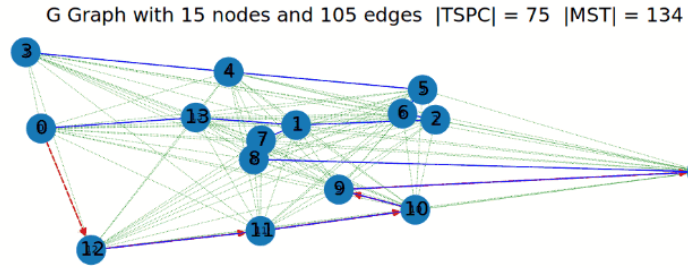


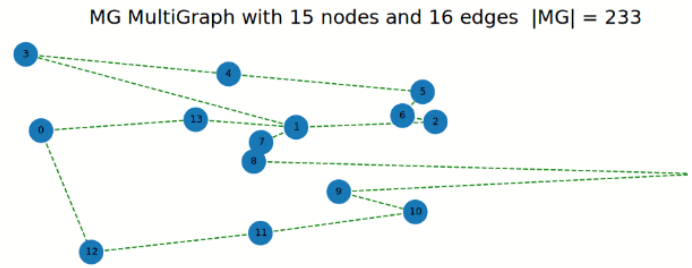


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10	9	14								
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

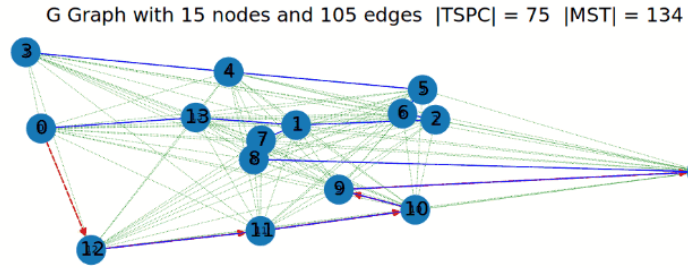


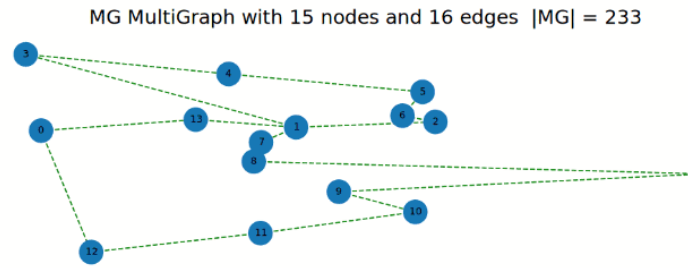


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10	9	14								
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

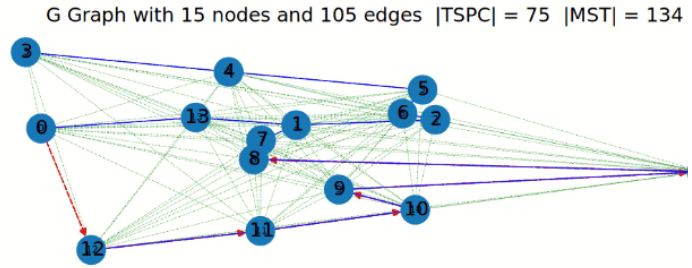


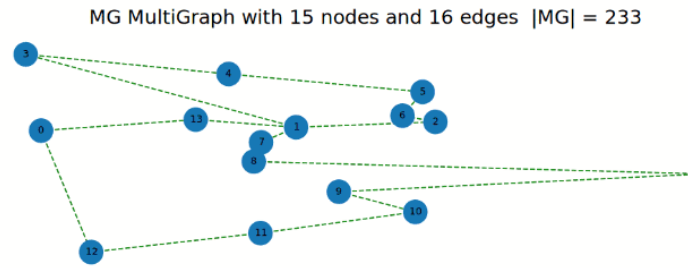


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10	9	14								
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

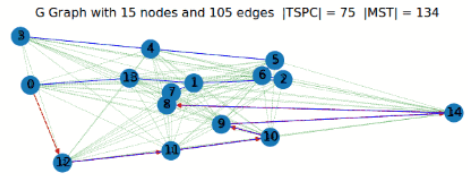


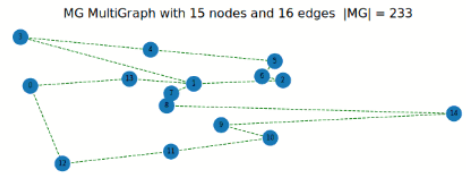


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1
cristo_tour	0	12	11	10	9	14	8							
greedy_tsp_tour														
sa_tsp_tour														

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

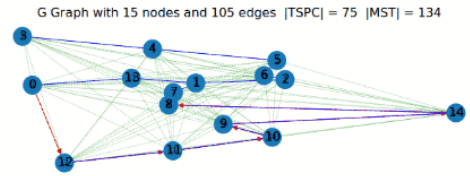


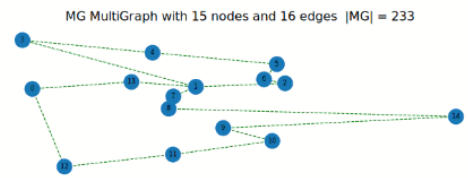


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8									
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                  init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

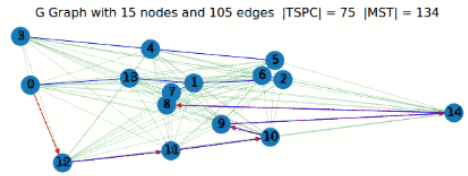


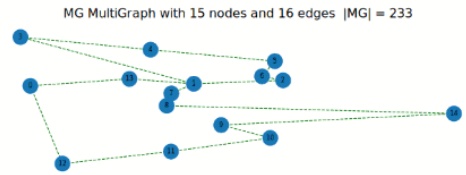


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8									
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

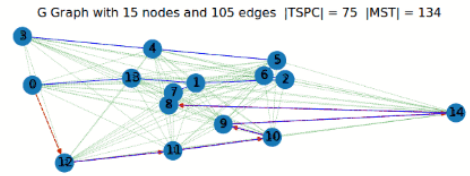


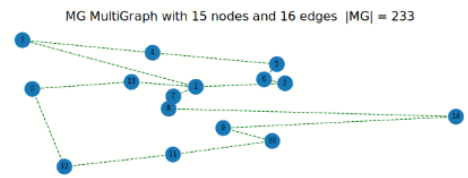


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7								
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                  init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

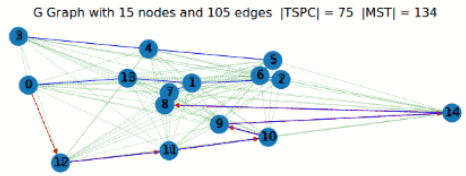


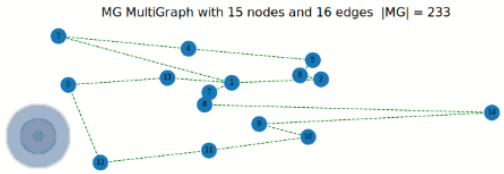


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7								
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

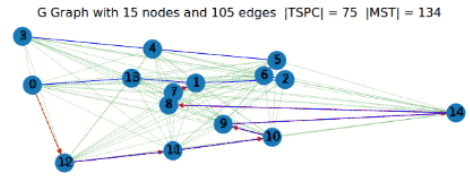


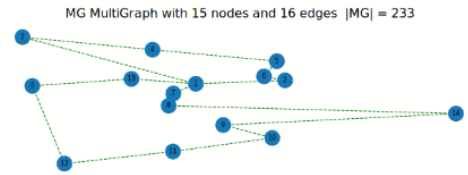


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7								
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

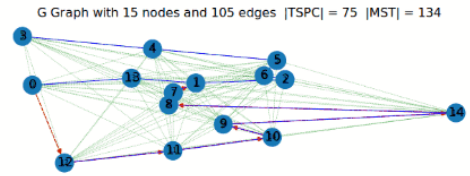


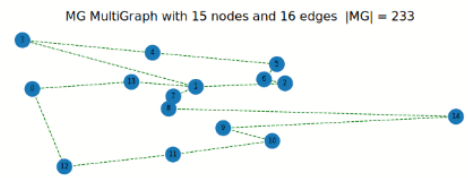


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1							
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                  init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

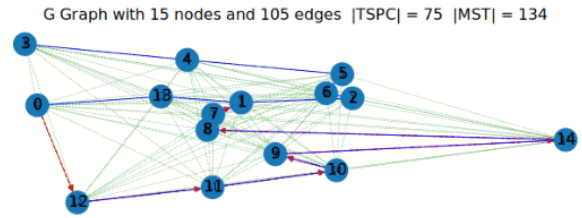


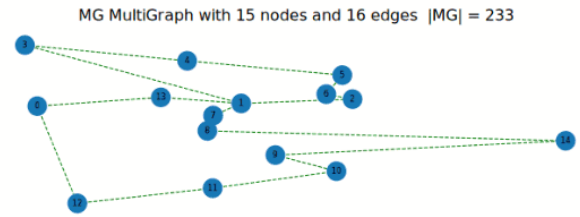


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1							
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

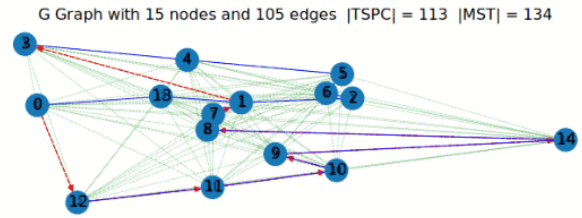


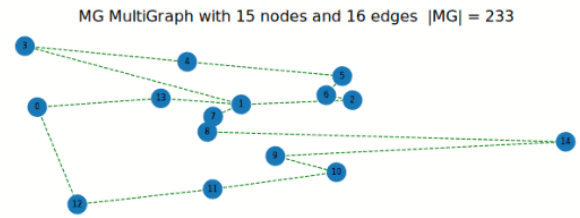


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1							
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

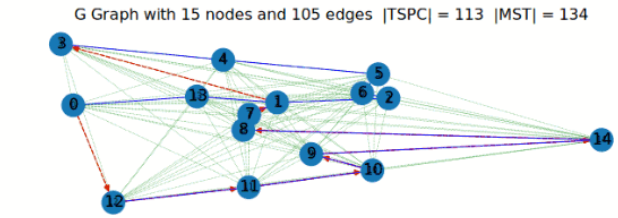


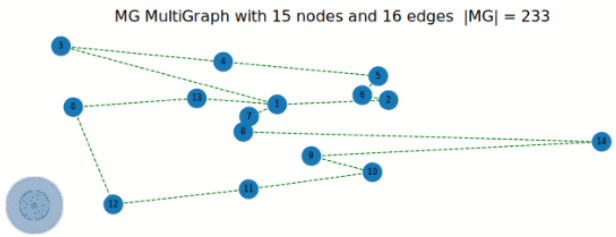


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1	3						
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                  init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

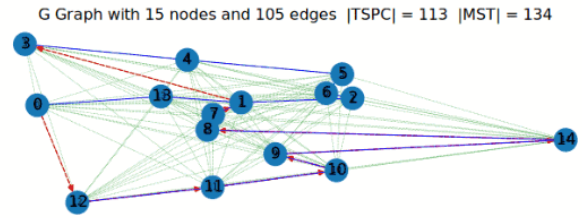


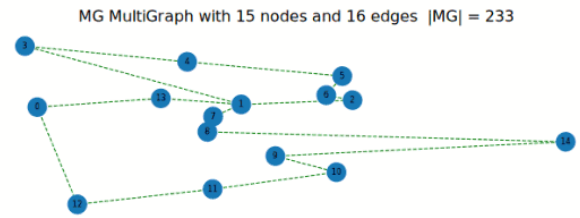


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1	3						
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

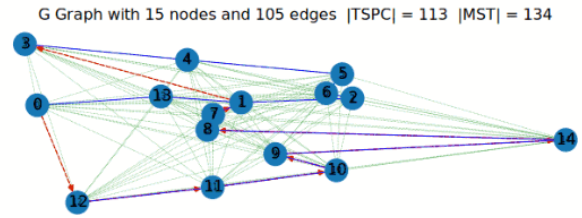


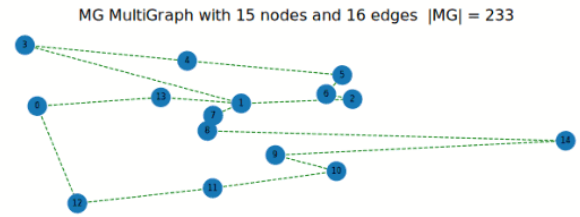


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1	3						
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                  init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

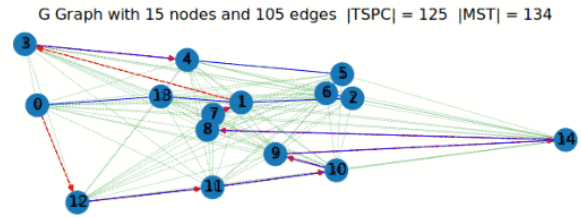


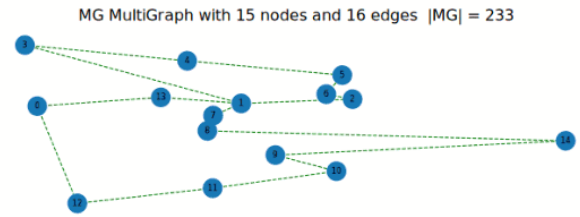


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1	3						
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                 init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

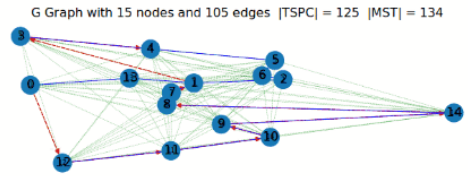


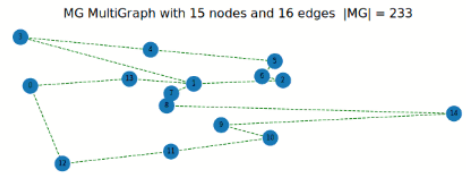


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1	3	4					
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

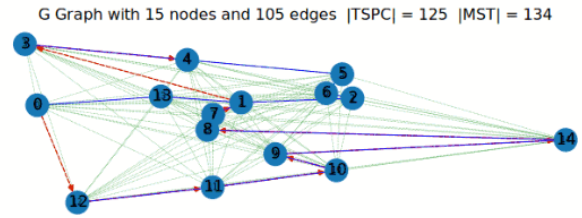


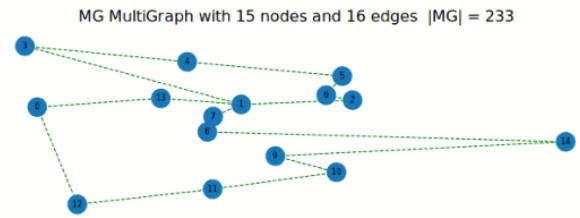


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1	3	4					
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                  init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```

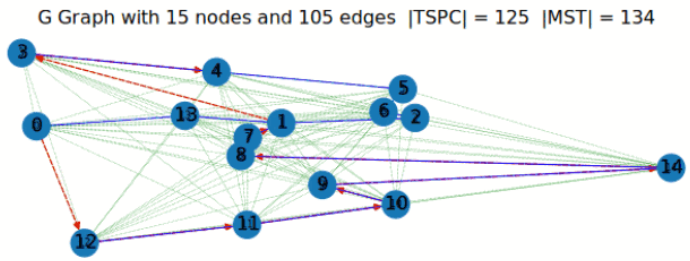


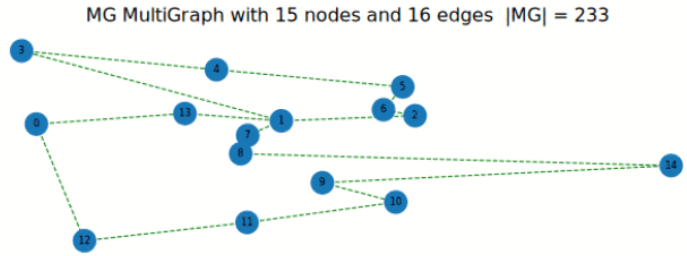


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13	13, 0
cristo_tour	0	12	11	10	9	14	8	7	1	3	4					
greedy_tsp_tour																
sa_tsp_tour																

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                  init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

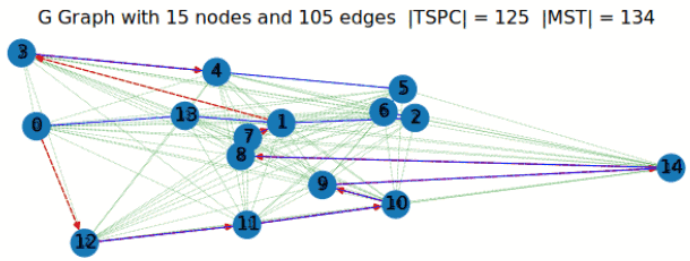


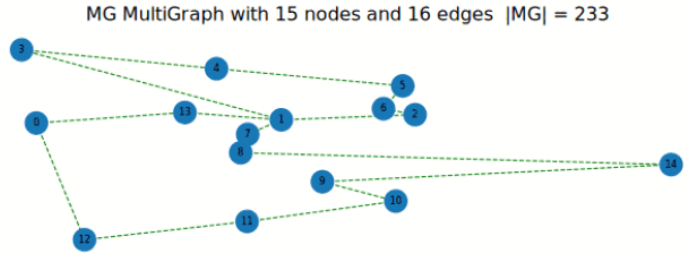


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4				
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                         init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

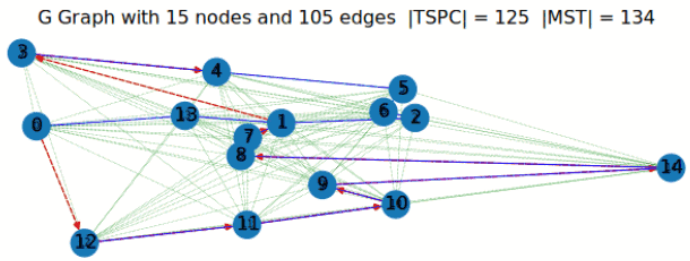


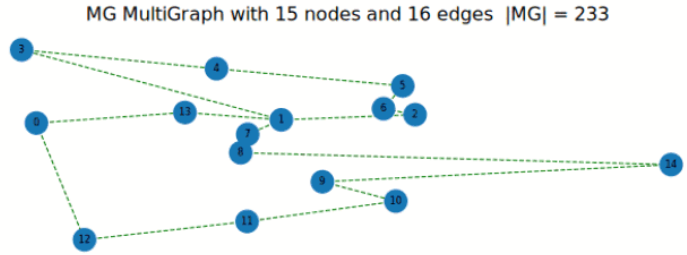


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4				
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

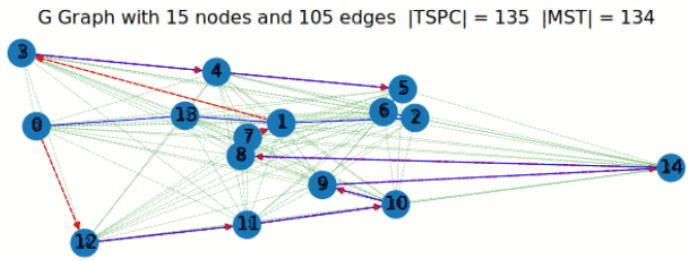


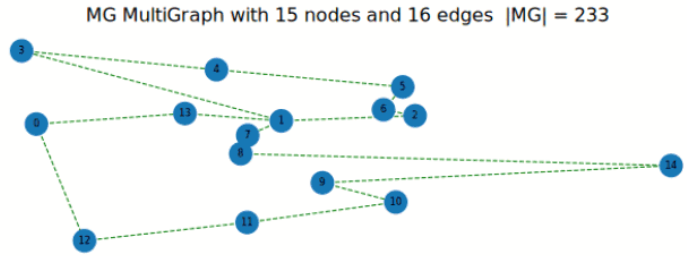


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4				
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

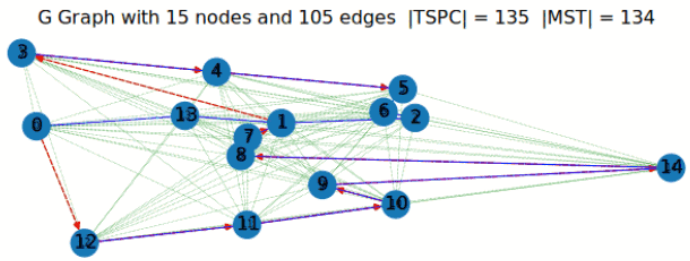


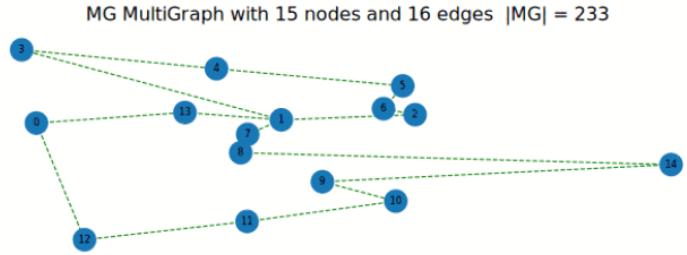


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5			
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                  init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

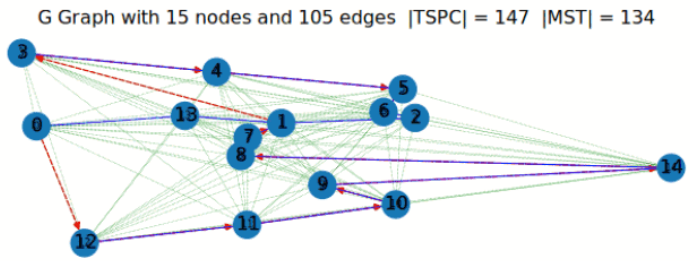


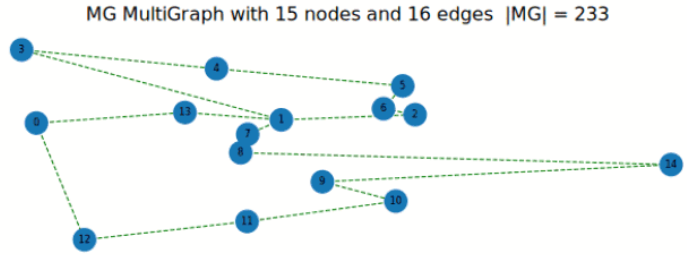


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5			
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

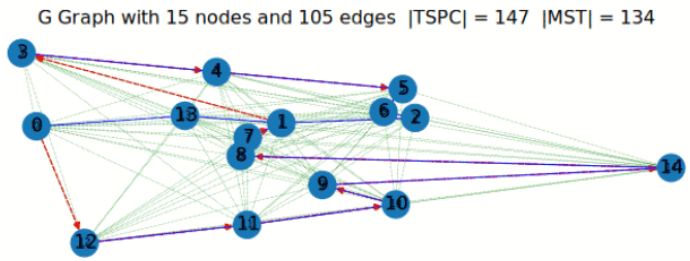


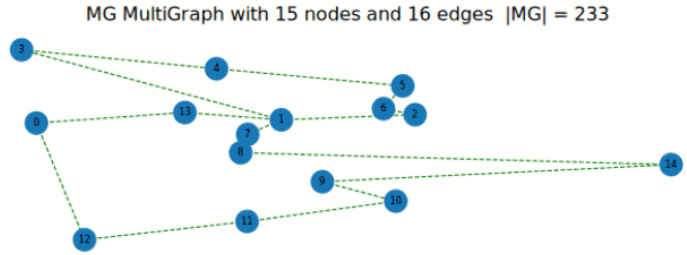


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6		
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

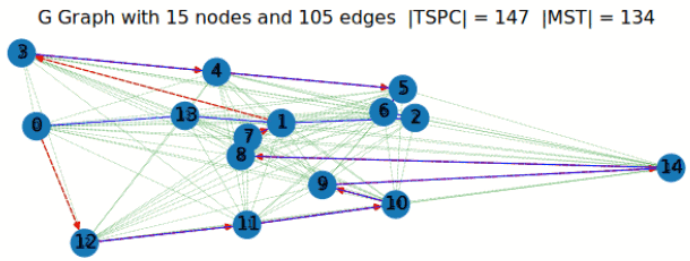


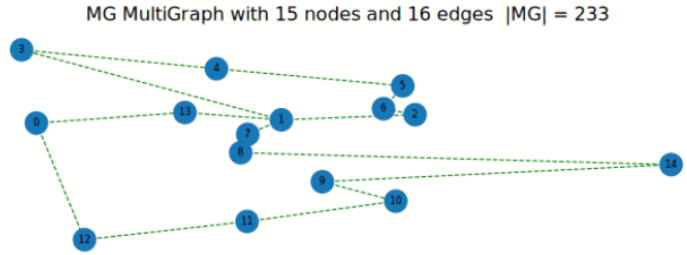


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6		
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

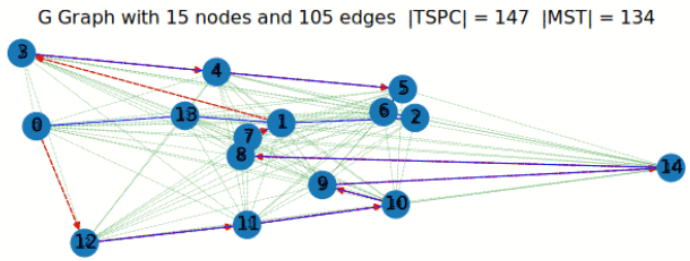


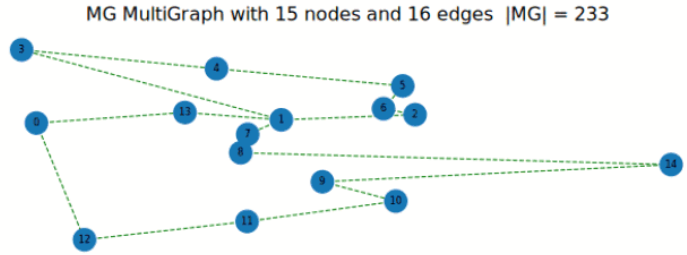


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6		
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

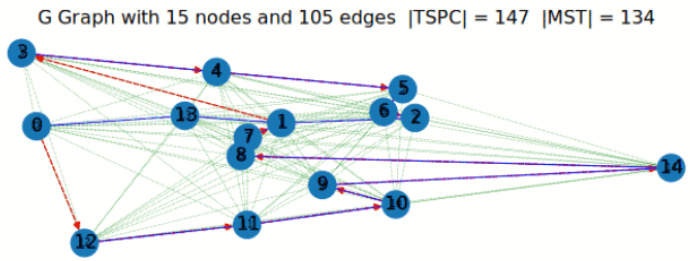


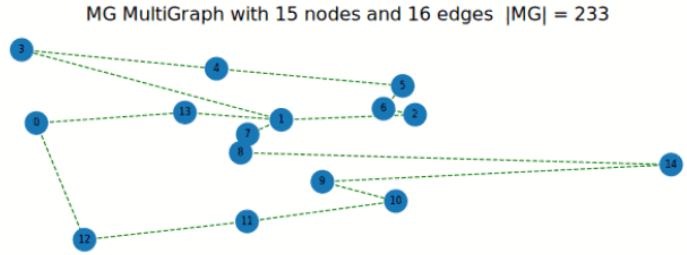


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6		
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

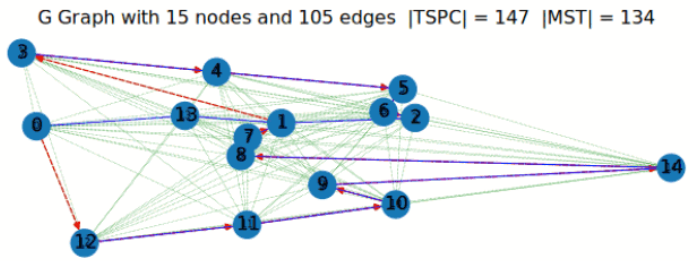


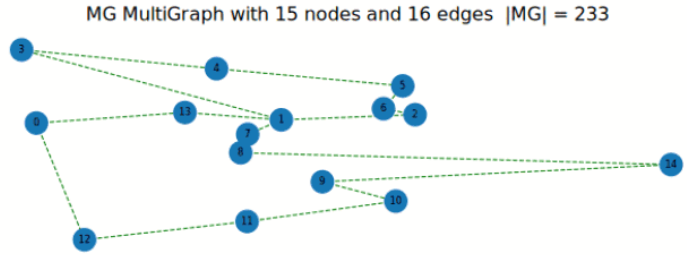


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

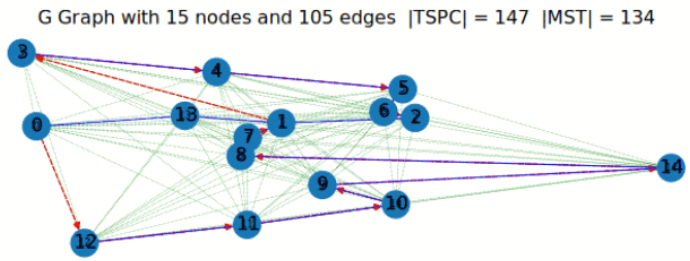


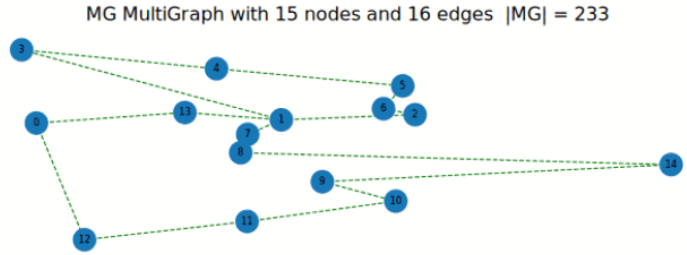


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

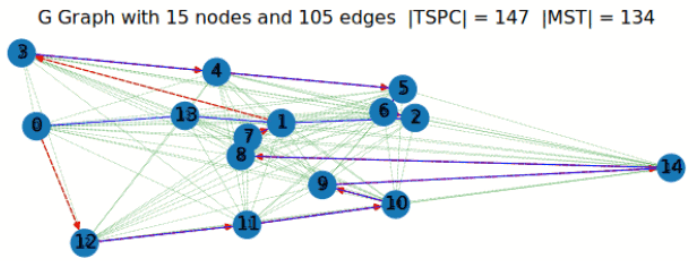


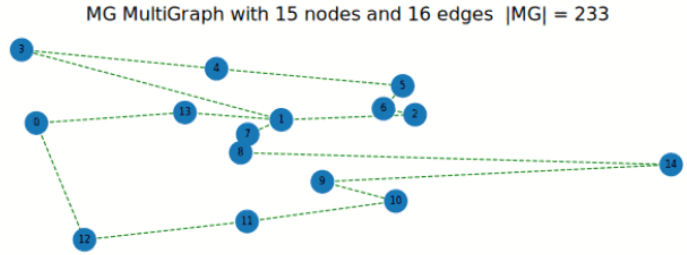


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

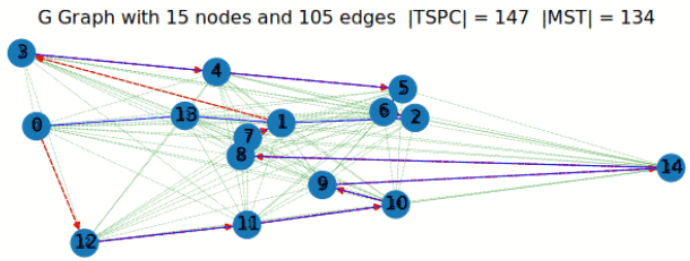


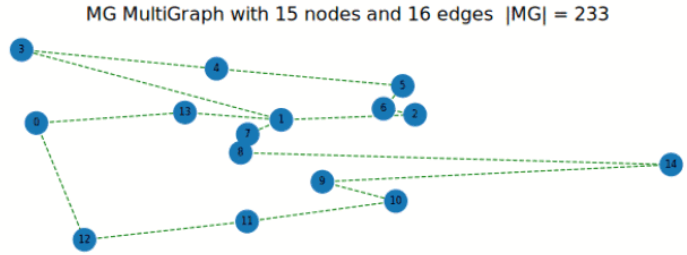


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                         init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

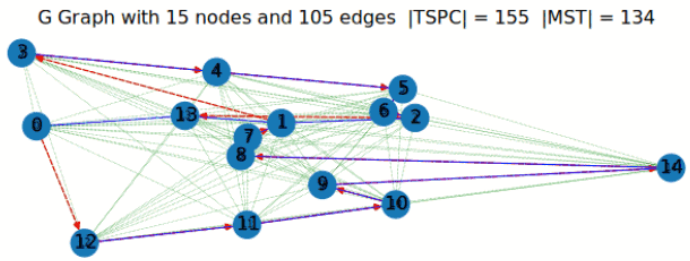


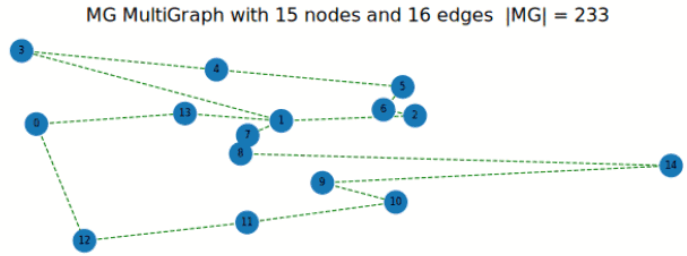


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

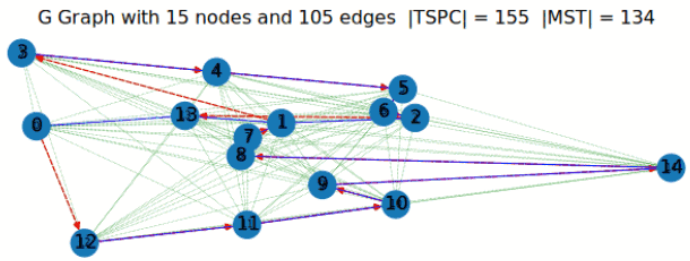


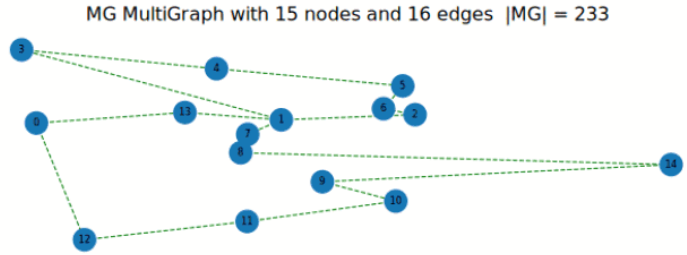


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	13
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

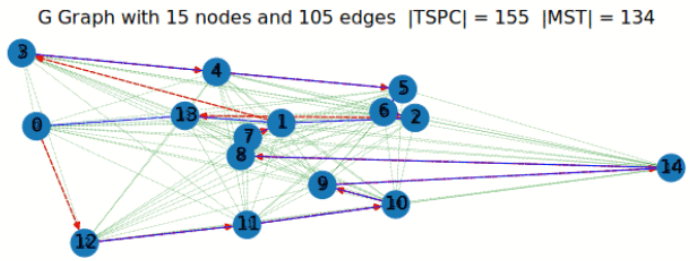


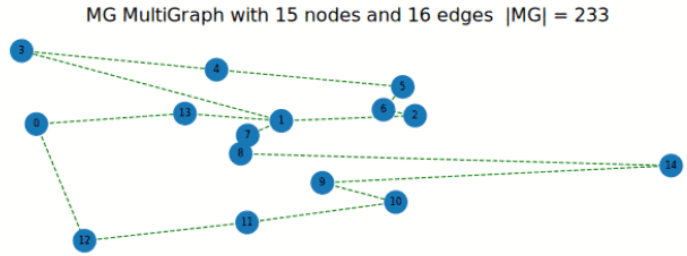


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	13
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST
```

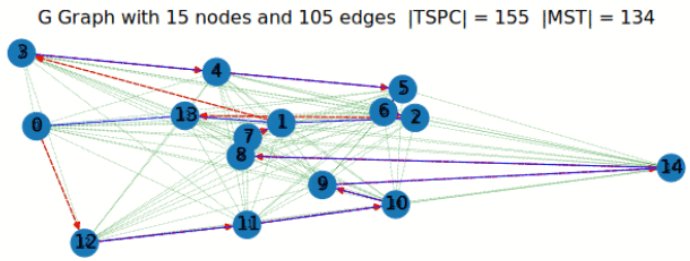


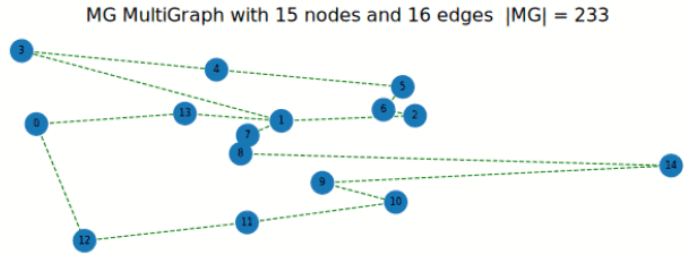


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	13
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                  init_cycle=cristo_tour, max_iteratio

    return cristo_tour, MST
```

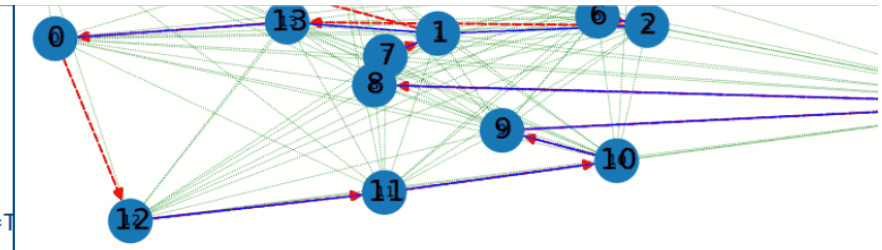




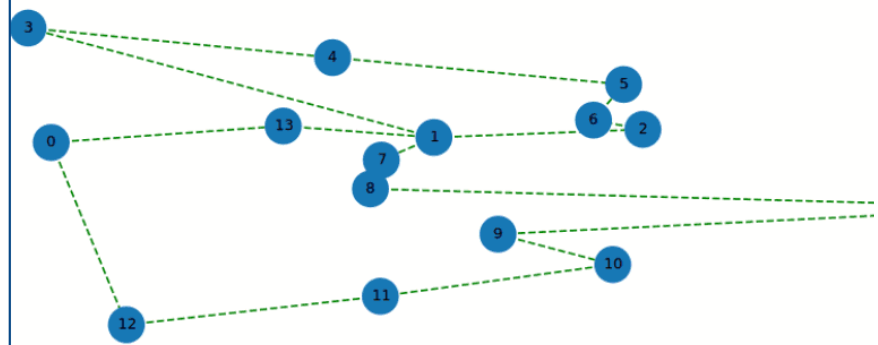
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	13
greedy_tsp_tour															
sa_tsp_tour															

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                    init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```



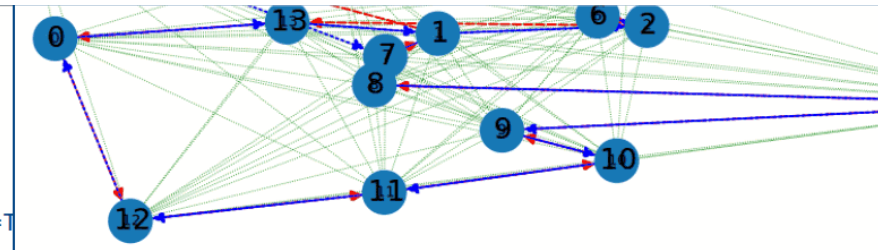
MG MultiGraph with 15 nodes and 16 edges |MG| =



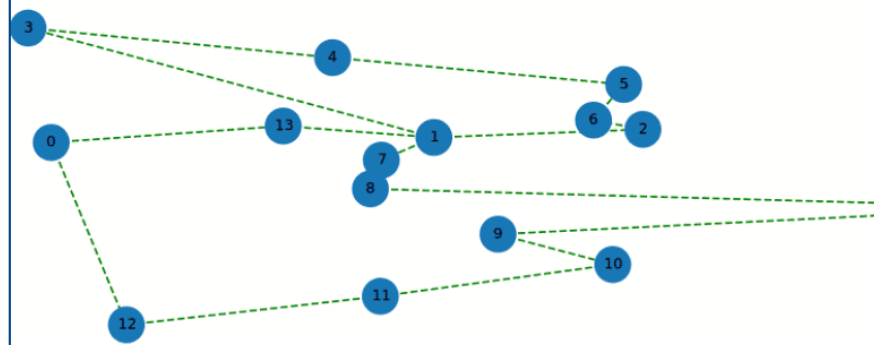
1	2	3	4	5	6	7	8	9	10	11	
12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6
12	11	10	9	14	8	7	1	3	4	5	

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                    init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```



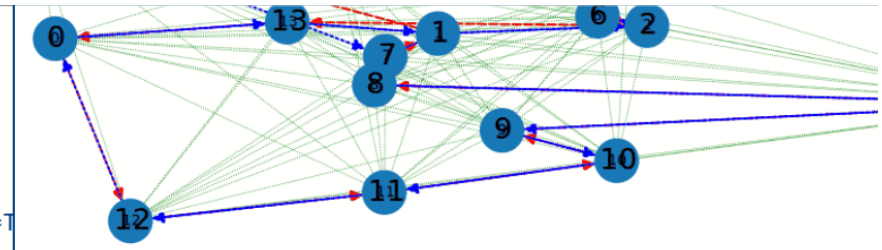
MG MultiGraph with 15 nodes and 16 edges |MG| =



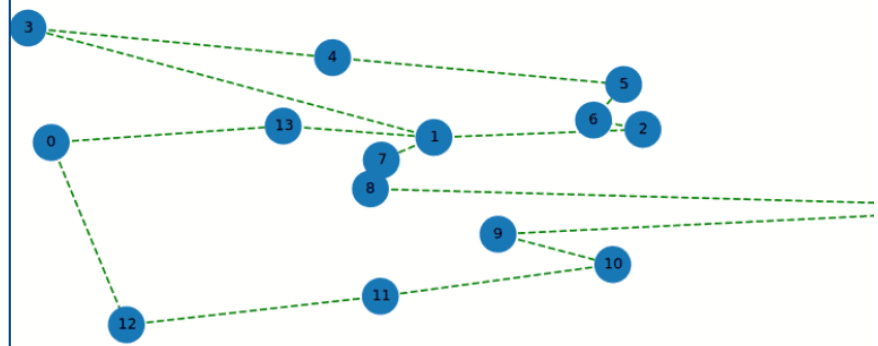
1	2	3	4	5	6	7	8	9	10	11	
12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6
12	11	10	9	14	8	7	1	3	4	5	

Выполнение алгоритма Кристофидеса

```
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                  init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```



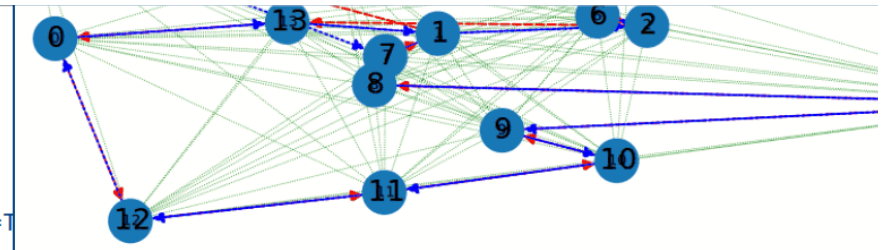
MG MultiGraph with 15 nodes and 16 edges |MG| =



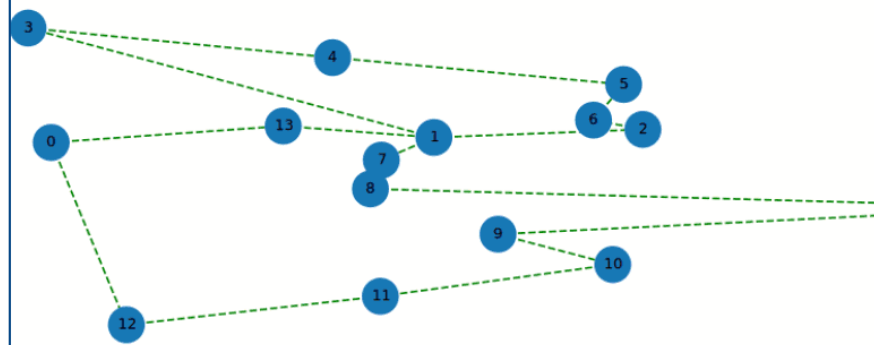
1	2	3	4	5	6	7	8	9	10	11	
12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6
12	11	10	9	14	8	7	1	3	4	5	

Выполнение алгоритма Кристофидеса

```python
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                    init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```



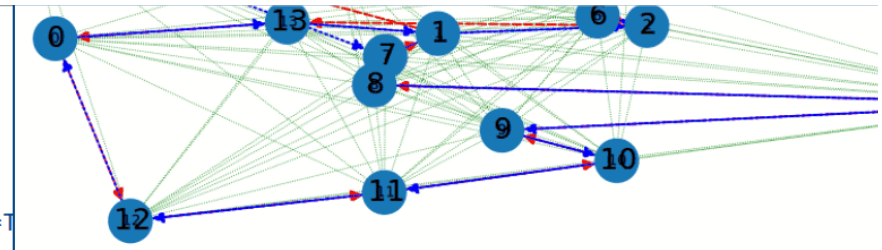
MG MultiGraph with 15 nodes and 16 edges |MG| =



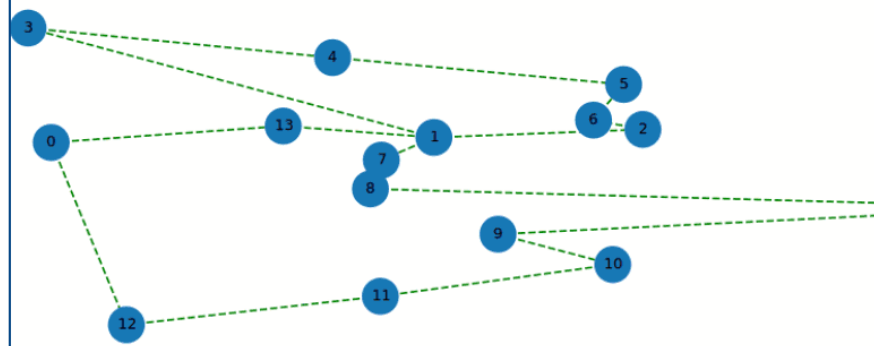
1	2	3	4	5	6	7	8	9	10	11	
12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6
12	11	10	9	14	8	7	1	3	4	5	

Выполнение алгоритма Кристофидеса

```
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```



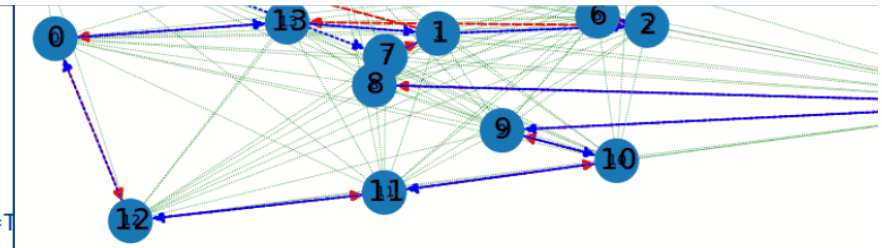
MG MultiGraph with 15 nodes and 16 edges |MG| =



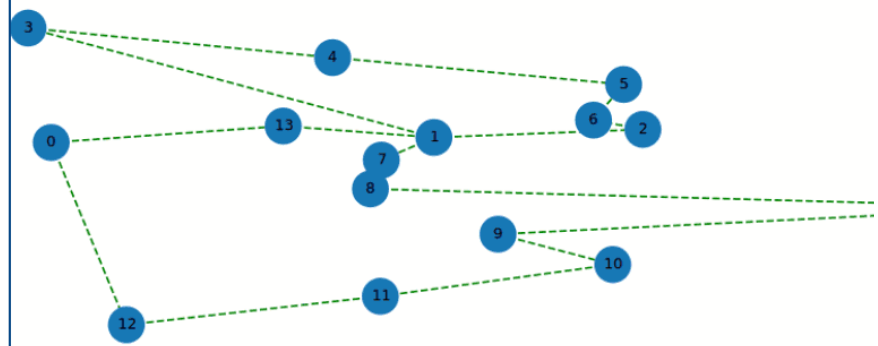
1	2	3	4	5	6	7	8	9	10	11	
12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6
12	11	10	9	14	8	7	1	3	4	5	

Выполнение алгоритма Кристофидеса

```
def tsp_cristofides(G):
    MST = nx.minimum_spanning_tree(G)
    L = G.copy()
    L.remove_nodes_from([v for v, degree in MST.degree
                         if not (degree % 2)])
    min_matching = nx.min_weight_matching(L, maxcardinality=T
    MG = nx.MultiGraph()
    MG.add_edges_from(MST.edges)
    MG.add_edges_from(min_matching)

    euler_cycle = list(nx.eulerian_circuit(MG))
    cristo_tour = []
    start_node = euler_cycle[0][0]
    cristo_tour.append(start_node)
    for u, v in euler_cycle:
        if v not in cristo_tour:
            cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                        init_cycle=cristo_tour, max_iteratior

    return cristo_tour, MST
```



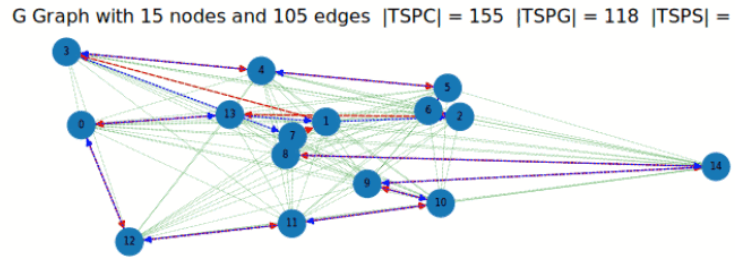
MG MultiGraph with 15 nodes and 16 edges |MG| =

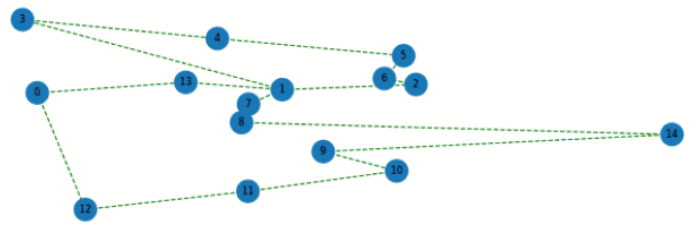


1	2	3	4	5	6	7	8	9	10	11	
12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6
12	11	10	9	14	8	7	1	3	4	5	

Выполнение алгоритма Кристофидеса

```
        cristo_tour.append(v)
    cristo_tour.append(start_node)

    greedy_tsp_tour = nx_app.greedy_tsp(G)
    sa_tsp_tour = nx_app.simulated_annealing_tsp(G,
                                                 init_cycle=cristo_tour, max_iteration

    return cristo_tour, MST



if __name__ == "__main__":
    G = subway_graph()
    cristo_tour, mst = tsp_cristofides(G)

    cristo_edges = list(nx.utils.pairwise(cristo_tour))
    for u, v, d in G.edges(data=True):
        if not ((u, v) in cristo_edges or (u, v) in cristo_ed
            d['weight'] = 2

    cristo_tour = tsp_cristofides(G)
    pass


'''
ToDo
```





	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
euler_cycle	0, 12	12, 11	11, 10	10, 9	9, 14	14, 8	8, 7	7, 1	1, 3	3, 4	4, 5	5, 6	6, 2	2, 1	1, 13
cristo_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	13
greedy_tsp_tour	0	13	1	2	6	5	4	3	7	8	14	9	10	11	12
sa_tsp_tour	0	12	11	10	9	14	8	7	1	3	4	5	6	2	13

Интернет поддержка курса

`http://discopal.ispras.ru/`

Вопросы?