

Реализация гамильтоновых циклов в кольцевых графах.



Цели:

- Изучить гамильтоновость кольцевых графов;
- Создать алгоритм, подтверждающий или опровергающий гамильтоновость в графе.
- Рассмотреть дороги Москвы при вложении в плоскость;
- Рассмотреть гамильтоновость случайных графов;

Задачи:

- Сформулировать и доказать необходимое и достаточное условие кольцевых графов;
- Сконструировать и реализовать на языке Python эффективные алгоритмы, проверяющие выполнение необходимого и достаточного условия гамильтоновости;

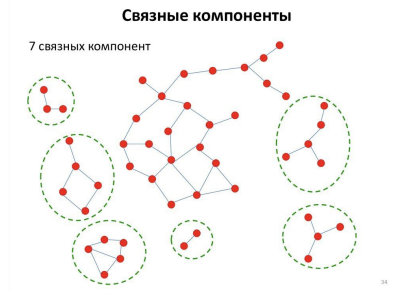
ОСНОВНЫЕ ПОНЯТИЯ:

Граф G — это пара множеств $G=(V,E)$, где $V(G)$ — непустое конечное множество элементов, называемых **вершинами** графа, а E — множество пар элементов из V (необязательно различных), называемых ребрами графа. $E=(u,v):u,v \in V$ — множество **ребер** графа G , состоящее из пар вершин (u,v) . Ребро (u,v) соединяет вершины u и v . **Смежные вершины** — это вершины, соединенные ребром. **Кратные ребра** — это ребра, которые соединяют одну и ту же пару вершин. Если две вершины соединены более чем одним ребром, говорят, что граф содержит кратные ребра.

(Граф G - это совокупность двух конечных множеств: множества точек, которые называются вершинами, и множества ребер. Число вершин графа не может быть равным нулю.)

Гамильтонов граф — граф, содержащий гамильтонов цикл. При этом гамильтоновым циклом является такой цикл (замкнутый путь), который проходит через каждую вершину данного графа ровно по одному разу; то есть простой цикл, в который входят все вершины графа.

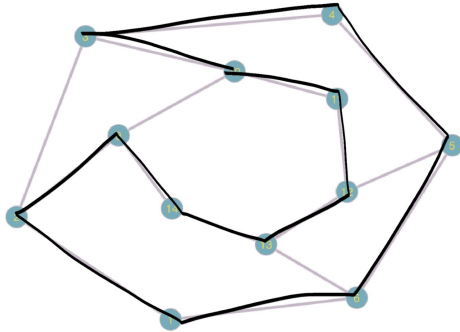
Компонента связности графа — некоторое множество вершин графа такое, что для любых двух вершин из этого множества существует путь из одной в другую, и не существует пути из вершины этого множества в вершину не из этого множества.



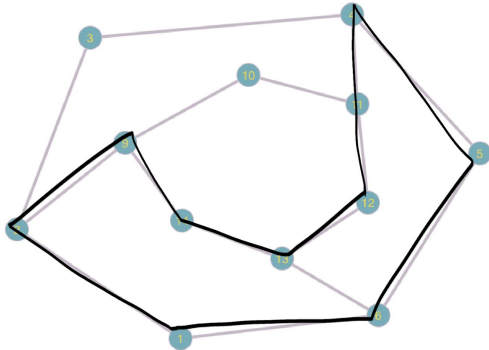
Москва: кольцевые графы (несколько

циклов, соседние из них соединены
рёбрами)

(1)



(2)



Сделаем предположение, что в графах с кольцами, где есть хотя бы две смежные вершины, имеется гамильтонов цикл. Проверим, изобразив произвольный граф: (1)

Действительно, гамильтонов цикл удалось найти.

Теперь соединим дороги несмежными вершинами: (2)

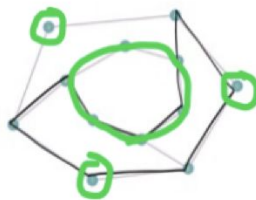
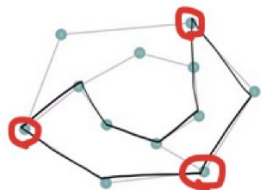
Как это получилось и с чем это связано?

Вывод и доказательство необходимого условия:



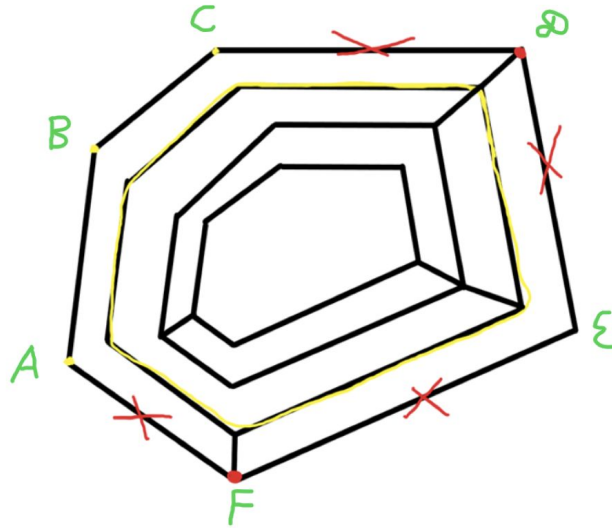
Утверждение 1. Пусть в графе G имеется гамильтонов цикл. Тогда количество $k := c(G - S)$ компонент связности U_1, \dots, U_k , получающихся в результате удаления вершин некоторого непустого подмножества $S \subset V(G)$ графа G , не превосходит количества удаленных вершин:

$$c(G - S) \leq |S|. \quad (3.1)$$



Таким образом, получили, что количество компонент связности больше количества удалённых вершин, следовательно гамильтонов цикл отсутствует.

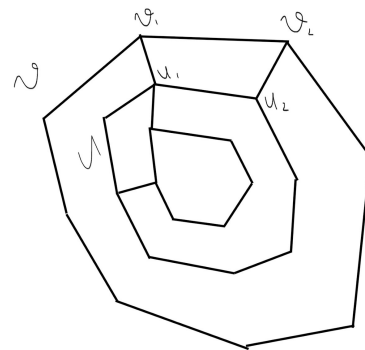
Теперь рассмотрим случай, когда хотя бы в одном из колец соединены несмежные вершины:



У нас две несмежные вершины внешнего кольца (4-го) соединены с третьим кольцом (D и F). Действуем по тому же самому принципу: удаляем эти две вершины и рёбра внешнего кольца, которые они соединяют. Таким образом у нас остаётся в качестве трёх компонентов связности 3 вершины во внешнем кольце (A, B и C) и в качестве одного компонента всё третье кольцо. Итого мы получили 4 компонента связности, а удалили всего 2 вершины, следовательно гамильтонов цикл данный граф содержать не будет.

Вывод и доказательство достаточного условия:

Предположим, что если для любой пары смежных колец V и U найдутся пары смежных вершин V_1, V_2 , принадлежащих V , и U_1, U_2 , принадлежащих U , такие что существуют ребра V_1-U и V_2-U_2 . Тогда гамильтонов цикл есть



Доказательство достаточного условия:

Докажем методом математической индукции:

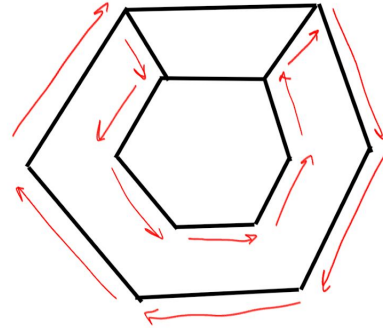
1. База индукции, $n=2$
1. Индукционное предположение. Пусть $n=k$ верно
2. Индукционный переход. $n=k+1$

$k+1$ -внешнее кольцо

k -внутреннем кольцо

Сделать обход по k можем по индукционному предположению, а с k и $k+1$ верно по базе индукции.

Доказано.



Реализация на Python:

На входе задаём размеры графа, а на выходе получаем ответ, выполняются наши условия или нет

```
class Ring:
    """
    Класс хранит число вершин в кольце и номера вершин связей вовне и номера вершин связи внутрь
    """
    count = 0
    rib_into = list() # ребра внутрь
    rib_out = list() # ребра наружу

class Rib:
    """
    ребро между кольцами
    """
    node_cur_ring: int
    node_next_ring: int

    def __init__(self, node_1=None, node_2=None):
        self.node_next_ring = node_2
        self.node_cur_ring = node_1
```

```
for i, ring in enumerate(rings[1:]): # первое кольцо не проверяем, так как нет внутреннего к нему
    ring: Ring
    was_found = False
    near_nodes = get_near_nodes(ring.rib_into, ring.count) # получить список смежных вершин с ребрами внутрь

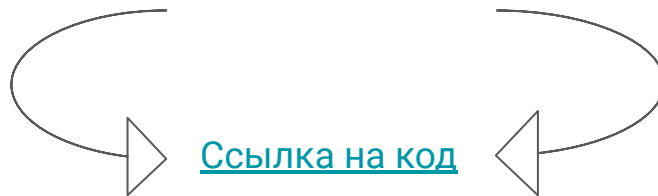
    for nodes in near_nodes: # идем по списку соседних нод
        nodes: tuple
        list_next_1 = get_next_node(ring.rib_into, nodes[0]) # список смежных вершин внешнего внутреннего кольца
        list_next_2 = get_next_node(ring.rib_into, nodes[1]) # список смежных вершин внешнего внутреннего кольца
        count = rings[i].count # количество узлов во внутреннем кольце

    # надо проверить если в списках соседние вершины
    for node_1 in list_next_1:
        for node_2 in list_next_2:
            if abs(node_1 - node_2) % count in [1, count - 1]: # ищем соседние вершины
                was_found = True # нашли - проверка кольца завершена, переходим к следующему
                break
            if was_found:
                break # нашли - проверка кольца завершена, переходим к следующему
        if was_found:
            break # нашли - проверка кольца завершена, переходим к следующему
    if not was_found:
        return False # ничего не нашли, значит нашли кольцо не удовлетворяющее условию
    return True # все кольца прошли проверку
```

```
Задать граф через список ребер
: return
"""
print("Кольца вводят следующим образом: \n"
      "-- задается число колец\n"
      "-- затем вводится число вершин в каждом кольце\n"
      "-- указываются какие вершины с какими связаны вовне (вершины нумеруются сверху по часовой стрелке)")
rings = input_rings()
# rings = test() # раскомментировать для тестирования и закомментировать верхнюю строчку

if check_necessary(rings):
    print(f"Необходимое условие: да\n")
else:
    print(f"Необходимое условие: нет\n")

if check_sufficient(rings):
    print(f"Достаточное условие: да")
else:
    print(f"Достаточное условие: нет")
```



Список используемой литературы:

1. Теория графов А.В. Омельчинко
2. Теория графов Д.В. Карпов
3. <https://mk.cs.msu.ru/images/8/89/Dm-l4-selezn.pdf>