Графы. III

Тимофей Хирьянов

1 Задача Эйлера

Задача 1. *Задача о кёнигсбергских мостах (нем. Königsberger Brückenproblem) — старинная математическая задача, в которой спрашивалось, как можно пройти по всем семи мостам Кёнигсберга, не проходя ни по одному из них дважды.*

Определение 1. *Эйлеров цикл — это цикл графа, проходящий через каждое ребро (дугу) графа ровно по одному разу.*

Определение 2. *Эйлеров граф — граф, содержащий эйлеров цикл.*

Определение 3. *Полуэйлеров граф — граф, содержащий эйлеров путь (цепь).*

Теорема 1. *Граф является Эйлеровым $\Leftrightarrow$ кратность всех вершин четная.*

Теорема 2. *Граф является полуэйлеровым $\Leftrightarrow$ кратность всех вершин четная, кроме двух.*

Доказательство этих двух теорем опустим.

2 Задача Гамильтона

Задача 2. *Задача Гамильтона — посетить каждую вершину ровно один раз.*

Определение 4. *Гамильтонов граф — граф, содержащий гамильтонов цикл.*

Определение 5. *Гамильтонов цикл — цикл, содержащий все вершины данного графа.*

Определение 6. *Полугамильтонов граф — граф, содержащий гамильтонов путь.*

3 Задача о китайском почтальоне

Задача 3. *Задача о китайском почтальоне — посетить все ребра не менее одного раза, при этом так, чтобы суммарная длина траектории путь была минимальна.*

Чем эта задача похожа на задачу Эйлера? Тем, что мы посещаем ребра, а не вершины, каждое ребро обязаны посетить хотя бы один раз. Очевидно, что в случае если граф Эйлеров, то задача имеет однозначное решение: мы посещаем каждое ребро два раза. Но задача почтальона несколько шире — мы можем посетить каждое ребро *как минимум один раз*. То есть, по одному ребру можно проходить два раза. И вот тут возникает вопрос: по какому проходить два раза так, чтобы путь был *наименьшим*. То есть наша задача — сделать из неэйлерового графа хотя бы полуэйлеров. Эта задача нетривиальна.

4 Задача коммивояжера

Задача 4. *Найти оптимальный по длине гамильтонов цикл.*

Заметим, что в полносвязном графе задача Гамильтона не возникает, но имеет место задача коммивояжера.

4.1 Метод полного перебора

Можно перебрать все возможные пути, а потом из них выбрать наименьший.Этот алгоритм выполняется за $O(n!)$. Приведем решение задачи коммивояжера:

```
def voyager(G, node, path = [], pathlen = 0, used = set()):
    global optimalpath, optimalpathlen
    if len(G) == len(used):
        if pathlen < optimalpathlen and node in G[path[-1]]:
            optimalpathlen = pathlen
            opltimalpath  = path
    else:
        for neighbour in G[node]:
            if neighbour not in used:
                voyager(G, neighbour, path + [neighbour],pathlen +
                        G[node][neighbour], used + {neighbour})
```