Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа №3
«Метод решения задачи о наидлиннейшем пути в направленном графе без контура»

Выполнил:
ст.гр.953503
Басенко К. А.
Проверил:
Дугинов О. И.

Минск 2022

Постановка задачи

Пусть имеется направленный граф без контура (ориентированный ациклический граф). Требуется найти самый длинный путь между двумя вершинами в этом графе.

Описание алгоритма метода

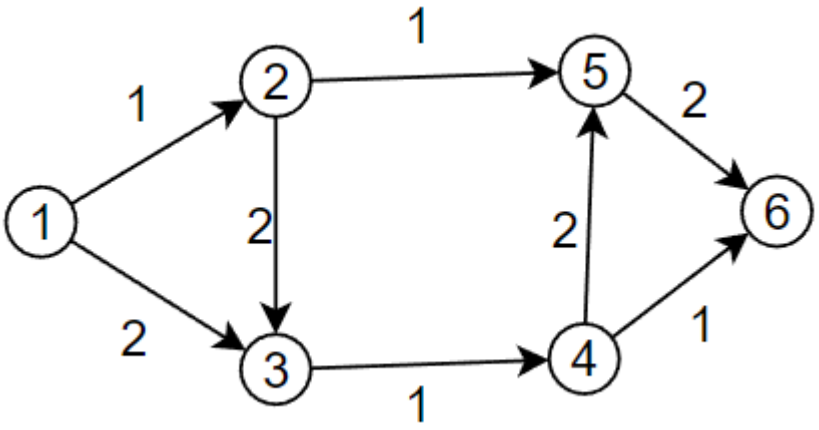
Задача о самом длинном пути — это задача поиска простого пути максимальной длины в заданном графе.Самый длинный путь A между двумя заданными вершинами s и t во взвешенном графе G — это то же самое, что и кратчайший путь в графе −G, полученном из G путём замены всех весов на веса с обратным знаком. Таким образом, если кратчайший путь можно найти в −G, то можно найти и самый длинный путь в G.

Для большинства графов такое преобразование бесполезно, поскольку создаёт циклы отрицательной длины в −G. Но если G является ориентированным ациклическим графом, невозможно создать отрицательный цикл и самый длинный путь в G может быть найден за линейное время, применив алгоритм поиска кратчайшего пути в −G (тоже ориентированный ациклический граф), который работает за линейное время. Например, для любой вершины v в ориентированном ациклическом графе длина самого длинного пути, заканчивающегося в v, может быть получена выполнением следующих шагов:

- Осуществляем топологическую сортировку заданного ориентированного ациклического графа (ОАГ).
- Для каждой вершины v ОАГ в топологической сортировке вычисляем длину самого длинного пути, завершающегося в вершине v путём просмотра входящих дуг от соседей и добавления веса входяще дуги к максимальной длине в записях этих соседей. Если v не имеет входящих дуг, присваиваем длину самого длинного пути, кончающегося в v, нулю.

Когда это будет сделано, самый длинный путь во всём графе можно получить, начав с вершины v с самым большим записанным значением и проходя в обратном порядке, выбирая входящую дугу, у которой запись в начальной вершине имеет наибольшее значение.

Результат работы
Тест 1



Стартовая вершина: 1
Конечная вершина: 6

Результат:

```
Наидлинейший путь из 1 в 6: [1, 2, 3, 4, 5, 6]
```

Код программы

```
from math import inf
from tracemalloc import start



class Arc:
    def __init__(self, node1, node2, cost):
        self.node1 = node1
        self.node2 = node2
        self.cost = cost



def get_topological_order(graph: list, nodes: list):
    color = ["white"] * len(nodes)
    result = []


    def dfs(u: int):
        color[u - 1] = "grey"
        for i in graph:
            if i.node1 == u:
                if color[i.node2 - 1] == "white":
                    dfs(i.node2)
        color[u - 1] = "black"
        result.insert(0, u)
    for i in nodes:
        if color[i - 1] == "white":
            dfs(i)
    return result



def remove_redudant_nodes(ordered_nodes: list):
    is_start = False
    i = 0
    while i < len(ordered_nodes):
        if ordered_nodes[i] == start_node:
            is_start = True
        elif ordered_nodes[i] == finish_node:
            if not is_start:
                return None
            else:
                return ordered_nodes[0: i + 1]
        elif not is_start:
            ordered_nodes.remove(i)
            i = i - 1
        i = i + 1
```

```
def get_longest_path(nodes: list, graph: list, start_node: list,
finish_node: list):
    ordered_nodes = get_topological_order(graph, nodes)
    ordered_nodes = remove_redudant_nodes(ordered_nodes)
    if ordered_nodes == None:
        print("No paths's")
        return None
    opt = {}
    x = {}
    for i in ordered_nodes:
        if i == start_node:
            opt.update({i: 0})
            x.update({i: None})
        else:
            max = -inf
            previous_node = 0
            for arc in graph:
                if arc.node2 == i:
                    value = opt[arc.node1] + arc.cost
                    if max < value:
                        max = value
                        previous_node = arc.node1
            opt.update({i: max})
            x.update({i: previous_node})
    path = []
    j = finish_node
    while True:
        if j == start_node:
            path.insert(0, start_node)
            break
        path.insert(0, j)
        j = x[j]
    return path



if __name__ == "__main__":
    nodes = [1, 2, 3, 4, 5, 6]
    graph = [
        Arc(1, 2, 1), Arc(1, 3, 2),
        Arc(2, 5, 1), Arc(3, 4, 1),
        Arc(4, 5, 2), Arc(4, 6, 1),
        Arc(5, 6, 2), Arc(2, 3, 2)
    ]
    start_node = 1
    finish_node = 6
```

```
path = get_longest_path(nodes, graph, start_node, finish_node)
print(f"Наидлинейший путь из {start_node} в {finish_node}: {path}")
```