Лабораторная работа № 9 по курсу дискретного анализа: Графы

Выполнил студент группы М8О-308Б-21 МАИ *Белоносов Кирилл.*

Условие

1. Общая постановка задачи: Реализовать алгоритм на графе
2. Вариант задания(2):

   Алгоритм: Задан взвешенный неориентированный граф, состоящий из n вершин и m ребер. Вершины пронумерованы целыми числами от 1 до n. Необходимо найти длину кратчайшего пути из вершины с номером start в вершину с номером finish при помощи алгоритма Беллмана- Форда. Длина пути равна сумме весов ребер на этом пути. Обратите внимание, что в данном варианте веса ребер могут быть отрицательными, поскольку алгоритм умеет с ними работать. Граф не содержит петель, кратных ребер и циклов отрицательного веса.

Метод решения

Для реализации алгоритма Форда-Беллмана используется подход динамического программированя. Для этого заведем следующую динамику: Пусть dp[k][u] - путь кратчайшей длины заканчивающийся в вершине u длины k. Пусть s - стартовая вершина, тогда dp[0][s] = 0, а dp[0][u] = $+\infty$. Тогда переход мы определим как dp[k][u] = min(dp[k][u], dp[k][v] + w). За n - 1 данный алгоритм находит кратчайшие пути до каждой вершины из стартовой. Также алгоритм отслеживает отрицательные циклы - если n шаге пути уменьшились, значит был обнаружен отрицательный цикл. Сложность O(nm)

Описание программы

Опишем реализацию алгоритма в отдельной функции. Переменная end скажет какой элемент из нашей динамики необходимо вернуть.

```
struct Node {
    int u, v;
    ll w;
};

ll ford_bellman(int start, int end, int n, const vector<Node>& g) {
    vector<ll> dp(n, INF);
    dp[start] = 0;
    for(int i = 0; i < n - 1; ++i) {
        bool change = false;
```

```
        for(size_t j = 0; j < g.size(); ++j) {
            if (dp[g[j].u] + g[j].w < dp[g[j].v]) {
                dp[g[j].v] = dp[g[j].u] + g[j].w;
                change = true;
            }
        }
        if(!change)
            break;
    }
    return dp[end];
}
```

В точке входа мы считываем граф и вызываем функцию для алгоритма Форда-Беллмана. Если из цикла нам пришло значение равное INF, значит данная вершина не достижима и выведем, что нет решений.

```
int main() {
    cin.tie(0);
    ios::sync_with_stdio(false);
    int n, m, s, e;
    cin >> n >> m >> s >> e;
    vector<Node> g(m);
    for(int i = 0; i != m; ++i) {
        int u, v;
        ll w;
        cin >> u >> v >> w;
        --u;
        --v;
        g[i] = {u, v, w};
    }
    ll path = ford_bellman(--s, --e, n, g);
    if(path == INF) {
        cout << "No solution\n";
    } else {
        cout << path << "\n";
    }
    return 0;
}
```

Дневник отладки

Попытка 1-11

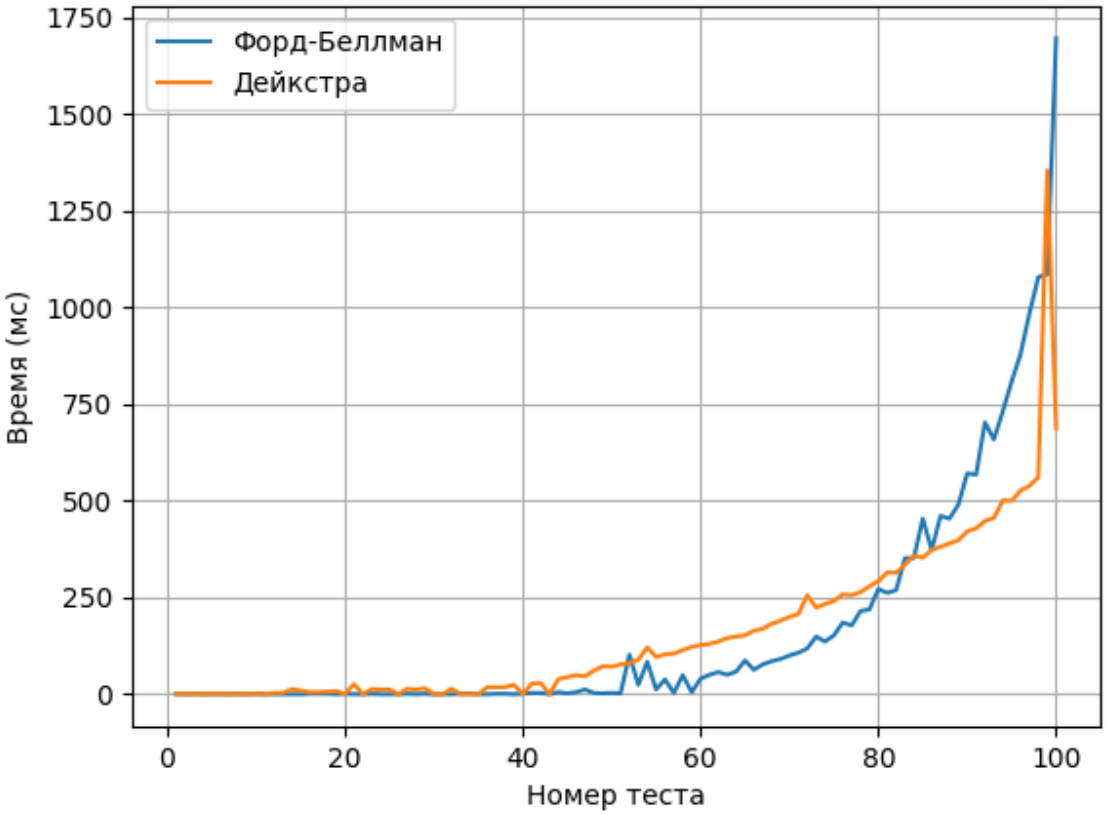
В данной попытка я исправлял ошибки реализации, также я добавил битовый флаг, чтобы закончить поиск новых путей раньше времени. Это помогло справиться с ошибкой TL.

0.1 Результаты тестирования

Тест производительности

Тестирование программы производилось на 100 сгенерированных тестах на нагруженных графах с шагом 100

Вставка:



Сложившаяся ситуация не особо удивляет, для разряженных графов Дейкстра работает чуть лучше. В моих тестах графы были разряженными что и позволило ему добиться чуть лучшей ассимптотики

Выводы

В результате данной лабораторной работы, я познакомился с основными алгоритмами поиска кратчайшего пути в нагруженном графе. Реализовал алгоритм Форда-Беллмана. Интересно отметить, что изученные ранее методы решения задач с помощью динамического программирования появляются здесь. Также для тестирования программы я реализовал алгоритм Дейкстры. Нельзя не отметить плюс алгоритма Форда-Белллмана

в том, что он может находить отрицательные циклы. Хотя теория алгоритмов на графах является довольно новой, нельзя не увидеть большой потенциал её применения в прокладывании маршрутов или искуственном интеллекте.