Лабораторная работа №9 по курсу Дискретного Анализа: Графы

Выполнил студент группы 08-308 МАИ Жерлыгин Максим Андреевич

Условие

Разработать программу на языке C или C++, реализующую указанный алгоритм согласно заданию:

Вариант 5:

Задан взвешенный ориентированный 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.

Формат входных данных

В первой строке заданы $1 \leq n \leq 10^5, 1 \leq m \leq 3 * 10^5, 1 \leq start \leq n$ и $1 \leq finish \leq n$. В следующих m строках записаны ребра. Каждая строка содержит три числа – номера вершин, соединенных ребром, и вес данного ребра. Вес ребра – целое число от -10^9 до 10^9.

Формат результата

Необходимо вывести одно число – длину кратчайшего пути между указанными вершинами. Если пути между указанными вершинами не существует, следует вывести строку **"No solution"** (без кавычек).

Метод решения

В отличие от алгоритма Дейкстры, этот алгоритм применим также и к графам, содержащим рёбра отрицательного веса. Мы считаем, что граф не содержит цикла отрицательного веса.

Заведём массив расстояний $d[0...n-1]$, который после отработки алгоритма будет содержать ответ на задачу (в каждом индексе будет храниться кратчайший путь до вершин). В начале работы мы заполняем его следующим образом: $d[start] = 0$, а все остальные элементы равны ∞.

Алгоритм Форда-Беллмана состоит из несколько фаз. На каждой фазе просматриваются все рёбра графа, и алгоритм пытается произвести релаксацию (relax) вдоль каждого ребра $(from, to)$ стоимости cost — это попытка улучшить значение $d[to]$ значением $d[from]+cost$. То есть это значит, что мы пытаемся улучшить ответ для вершины to, пользуясь ребром $(from, to)$ и текущим ответом для вершины from.

Утверждается, что достаточно $n-1$ фазы алгоритма, чтобы корректно посчитать длины всех кратчайших путей в графе. Для недостижимых вершин расстояние останется равным ∞.

Исходный код

Основной частью программы является реализация алгортима Беллмана-Форда, который принимает граф $graph$ по ссылке, количество вершин n, номер начальной вершины $start$ и номер конечной вершины $finish$.

```
int64_t BellmanFord(std::vector<edge> &graph, uint32_t n, uint32_t start, uint32_t finish) {
    std::vector<int64_t> d (n, INF);
    d[start] = 0;
    for (;;) {
        bool any = false;
        for (size_t j = 0; j < graph.size(); ++j)
            if (d[graph[j].from - 1] < INF)
                if (d[graph[j].to - 1] > d[graph[j].from - 1] + graph[j].cost) {
                    d[graph[j].to - 1] = d[graph[j].from - 1] + graph[j].cost;
                    any = true;
                }
        if (!any) break;
    }
    return d[finish];
}
```

В данном алгоритме удобнее всего хранить граф в ребрах. Для этого я создал струтуру edge, в которой были 3 значения: два типа uint_32t from и to, так как номер вершины не может быть меньше 1, и одно типа int_64t cost.

```
struct edge {
    uint32_t from;
    uint32_t to;
    int64_t cost;
};
```

Алгоритм просматривает ребра и, пока мы не дошли до конца, не заканчивает цикл. Внутри цикла алгоритм сравнивает значение в массиве и если оно больше, чем сумма предыдущего и веса, то к предыдщщему значению прибавляет вес ребра. Для его ускорения я создал переменную any которая проверяет изменилось ли значение в массиве d[], если да, то заканчивает цикл, чтобы избежать ненужных просмотров и сравнений.

Пример работы

```
mmaxim2710@DESKTOP-RDPBU3D:/mnt/c/Users/mmaxi/Desktop/coursera/DA_ex/
    lab9$ make clean
rm -f *.o solution
mmaxim2710@DESKTOP-RDPBU3D:/mnt/c/Users/mmaxi/Desktop/coursera/DA_ex/
    lab9$ make
g++ -std=c++17 -pedantic -Wall -O2 -c main.cpp -o main.o
g++ -std=c++17 -pedantic -Wall -O2 main.o -o solution
mmaxim2710@DESKTOP-RDPBU3D:/mnt/c/Users/mmaxi/Desktop/coursera/DA_ex/
    lab9$ ./solution
5 6 1 5
1 2 2
1 3 0
3 2 -1
2 4 1
3 4 4
4 5 5
5
```

Вывод

При написании этой лабораторной я не столкнулся с трудностями, так как для нее были нужны базовые знания графов, а также я уже встречался с этим алгоритмом в курсе дискретной математики.

Корректность алгоритма:
Для недостижимых из start вершин алгоритм отработает корректно: для них метка $d[]$ так и останется равной бесконечности.
Нужно доказать: после выполнения i фаз алгоритм Форда-Беллмана корректно находит все кратчайшие пути, длина которых (по числу рёбер) не превосходит i.
То есть для любой вершины $from$ обозначим через k число рёбер в кратчайшем пути до неё (если таких путей несколько, можно взять любой). Тогда это утверждение говорит о том, что после k фаз этот кратчайший путь будет найден гарантированно.

Доказательство. Рассмотрим произвольную вершину from, до которой существует путь из стартовой вершины $start$, и рассмотрим кратчайший путь до неё: $(p_0 = start, p_1, ..., p_k = from)$. Перед первой фазой кратчайший путь до вершины $p0 = start$ найден корректно. Во время первой фазы ребро (p_0, p_1) было просмотрено, следовательно, расстояние до вершины p1 было корректно посчитано после первой фазы.
Повторяя эти утверждения k раз, получаем, что после k-ой фазы расстояние до вершины $p_k = from$ посчитано корректно, что и требовалось доказать.
Последнее, что надо заметить — это то, что любой кратчайший путь не может иметь более $n - 1$ ребра. Следовательно, алгоритму достаточно произвести только $n - 1$ фазу. После этого ни одна релаксация гарантированно не может завершиться улучшением расстояния до какой-то вершины.

Существуют и другие алгоритмы поиска кратчайшего пути от одной вершины графа до другой. Например, алгоритм Дейкстры. Этот алгоритм является жадным и имеет сложность $O(VlogV)$, когда алгоритм Форда-Беллмана имеет сложность $O(V * E)$. Однако, первый не умеет работать с ребрами, имеющими отрицательный вес, поэтому для моей задачи он не применим.