

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего
образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа № 7 "Задачи 1650, 1450, 1806"

по дисциплине Алгоритмы и структуры данных

Выполнила: студентка гр. **R3238**
поток **2.1**

Нечаева А. А.

Преподаватель: *Тропченко Андрей Александрович*

Санкт-Петербург, 2024

1 Цель

Разработать и реализовать алгоритмы для решения задач 1650, 1450 и 1806.

2 Задача 1650

1650. Миллиардеры

Ограничение времени: 3.0 секунды
Ограничение памяти: 64 МБ

Возможно, вы знаете, что из всех городов мира больше всего миллиардеров живёт в Москве. Но, поскольку работа миллиардера подразумевает частые перемещения по всему свету, в определённые дни какой-то другой город может занимать первую строчку в таком рейтинге. Ваши приятели из ФСБ, ФБР, МІ5 и Шин Бет скинули вам списки перемещений всех миллиардеров за последнее время. Ваш работодатель просит посчитать, сколько дней в течение этого периода каждый из городов мира был первым по общей сумме денег миллиардеров, находящихся в нём.

Исходные данные

В первой строке записано целое число n — количество миллиардеров ($1 \leq n \leq 10000$). В каждой из следующих n строк записаны данные на определённого человека: его имя, название города, где он находился в первый день данного периода, и размер состояния. В следующей строке записаны целые числа m и k — количество дней, о которых есть данные, и количество зарегистрированных перемещений миллиардеров соответственно ($1 \leq m \leq 50000$; $0 \leq k \leq 50000$). В следующих k строках записан список перемещений в формате: номер дня (от 1 до $m - 1$), имя человека, название города назначения. Вы можете считать, что миллиардеры путешествуют не чаще одного раза в день и что они отбывают поздно вечером и прибывают в город назначения рано утром следующего дня. Список упорядочен по возрастанию номера дня. Все имена и названия городов состоят не более чем из 20 латинских букв, регистр букв имеет значение. Состояния миллиардеров лежат в пределах от 1 до 100 миллиардов.

Результат

В каждой строке должно содержаться название города и, через пробел, количество дней, в течение которых этот город лидировал по общему состоянию миллиардеров, находящихся в нём. Если таких дней не было, пропустите этот город. Города должны быть отсортированы по алфавиту (используйте обычный порядок символов: ABC...Zabc...z).

Пример

исходные данные	результат
5 Abramovich London 15000000000 Deripaska Moscow 10000000000 Potanin Moscow 5000000000 Berezovsky London 2500000000 Khodorkovsky Chita 1000000000 25 9 1 Abramovich Anadyr 5 Potanin Courchevel 10 Abramovich Moscow 11 Abramovich London 11 Deripaska StPetersburg 15 Potanin Norilsk 20 Berezovsky Tbilisi 21 Potanin StPetersburg 22 Berezovsky London	Anadyr 5 London 14 Moscow 1

Автор задачи: Павел Атнашев

Источник задачи: NEERC 2008, Четвертьфинал Восточного подрегиона

Рис. 1. Условие задачи 1650.

2.1 Краткое описание алгоритма

1. Входные данные: в первой строке записано целое число n — количество миллиардеров ($1 \leq n \leq 10000$). В каждой из следующих n строк записаны данные на определённого человека: его имя, название города, где он находился в первый день данного периода, и размер состояния.

В следующей строке записаны целые числа m и k — количество дней, о которых есть данные, и количество зарегистрированных перемещений миллиардеров соответственно ($1 \leq m \leq 50000$; $0 \leq k \leq 50000$). В следующих k строках записан список перемещений в формате: номер дня (от 1 до $m - 1$), имя человека, название города назначения.

Вы можете считать, что миллиардеры путешествуют не чаще одного раза в день и что они отбывают поздно вечером и прибывают в город назначения рано утром следующего дня. Список упорядочен по возрастанию номера дня. Все имена и названия городов состоят не более чем из 20 латинских букв, регистр букв имеет значение. Состояния миллиардеров лежат в пределах от 1 до 100 миллиардов

2. Создадим структуру, в которой будут находиться отсортированные по неубыванию множество городов, в котором будем хранить актуальное состояние каждого из городов, в случае перелета миллиардера сумма города изменяется, соответственно заново сортируется структура;

3. создадим еще одну структуру данных, в которой будем хранить имя миллиардера и информацию о городе, в котором он находится. Для этого будем использовать *словарь* (или *map*): ключ — имя миллиардера, значение — структура, в которой хранится информация о количестве денег и о длительности их нахождения в нем;

4. весь алгоритм работает за $O(n \cdot \ln n)$

5. Выходные данные: в каждой строке должно содержаться название города и, через пробел, количество дней, в течение которых этот город лидировал по общему состоянию миллиардеров, находящихся в нём. Если таких дней не было, пропустите этот город. Города должны быть отсортированы по алфавиту (используйте обычный порядок символов: ABC...Zabc...z).

2.2 Листинг

Листинг 1. Исходный код для 1650

```
1 #include <iostream>
2 #include <map>
3 #include <set>
4
5
6 // structure for city parameters
7 struct City {
8     std::string _name;
9     long long _money;
10    int _days;
11 } pCity[60000];
12
13 //structure for billionaires' parameters
14 struct Billionaire {
15     long long _money;
16     City *_loc;
17 } pBillionaire[10000];
18
19
20 int main() {
21
22     int n, m, k;
23     std::cin >> n;
24
25     int _city_number = 0;
26
27     std::map<std::string, City *> _cities;
28     std::map<std::string, Billionaire *> _billionaires;
29     std::set<std::pair<long long, City *>, std::greater<>>
        score;
30
31     for (int i = 0; i < n; i++) {
32         std::string name_person;
33         std::string name_city;
34         long long money;
35         std::cin >> name_person >> name_city >> money;
36
37         if (!_cities[name_city]) {
```

```
39         pCity[_city_number]._name = name_city;
40         pCity[_city_number]._money = money;
41         _cities[name_city] = &pCity[_city_number++];
42
43     } else _cities[name_city]->_money += money;
44
45
46     pBillionaire[i]._money = money;
47     pBillionaire[i]._loc = _cities[name_city];
48     _billionaires[name_person] = &pBillionaire[i];
49
50 }
51
52 for (auto &item : _cities) {
53     score.insert({item.second->_money, item.second});
54 }
55
56 int today = 0;
57
58 std::cin >> m >> k;
59
60 for (int i = 0; i < k; i++) {
61     int day;
62     std::string name_person;
63     std::string name_city;
64     std::cin >> day >> name_person >> name_city;
65
66     int count = day - today;
67     today = day;
68
69     auto a = score.begin();
70     auto b = a++;
71
72     if (a->first < b->first || a == score.end()) {
73         b->second->_days += count;
74     }
75
76     City *to_city = _cities[name_city];
77     Billionaire *who = _billionaires[name_person];
78
79     if (to_city == nullptr) {
80
81         pCity[_city_number]._name = name_city;
```

```
82         _cities[name_city] = &pCity[_city_number++];
83         to_city = _cities[name_city];
84     }
85
86     score.erase({who->_loc->_money, who->_loc});
87     score.erase({to_city->_money, to_city});
88
89     who->_loc->_money -= who->_money;
90
91     score.insert({who->_loc->_money, who->_loc});
92
93     who->_loc = to_city;
94     to_city->_money += who->_money;
95
96     score.insert({to_city->_money, to_city});
97 }
98
99 int count = m - today;
100
101 auto a = score.begin();
102 auto b = a++;
103
104 if (a->first < b->first || a == score.end()) {
105     b->second->_days += count;
106 }
107
108 for (auto &item : _cities) {
109     if (item.second->_days > 0) {
110         std::cout << item.first << " " << item.second->
            _days << std::endl;
111     }
112 }
113
114 return 0;
115 }
```

2.3 Результат

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10648572	05:18:05 10 май 2024	mistDragon	1650. Миллиардеры	Clang++ 17 x64	Accepted		0.562	12 396 KB

Рис. 2. Результат отправки задачи 1650.

3 Задача 1450

1450. Российские газопроводы

Ограничение времени: 0.5 секунды
Ограничение памяти: 64 МБ

Вступление

Большими неприятностями обернулся прошедший год для Государства Российского. То неурожай, то птичий грипп, то вечные споры хозяйствующих субъектов... А тут ещё и Президент задумал, наконец, собрать средства на покупку новой балалайки и ручного медведя для своего двоюродного племянника. Все эти факторы (в особенности, конечно, последний) сильно ударили по экономике государства. Посоветавшись со своими друзьями в валенках и ушанках, Президент решил воспользоваться традиционным методом укрепления национального бюджета - увеличением налога на транспортировку газа.

Задача

Сеть российских газопроводов представляет собой N перекачивающих станций, некоторые из которых соединены газопроводами. Для каждого из M газопроводов известны номера станций $A[i]$ и $B[i]$, которые он соединяет, и его прибыльность $C[i]$, т.е. то количество долларов, которое будет ежегодно приносить в виде налогов перекачка газа по этому газопроводу. Каждая пара станций соединена не более чем одним газопроводом.

Сеть была построена советскими инженерами, которые точно знали, что газ поставляется из месторождений Украины в Сибирь, а не наоборот. Поэтому все газопроводы являются однопроводными, т.е. для каждого газопровода перекачка газа возможна только в направлении из станции с номером $A[i]$ на станцию с номером $B[i]$. Более того, для любых двух станций X и Y верно, что если возможна перекачка газа из X на Y (возможно, через промежуточные станции), то обратная перекачка из Y на X невозможна. Известно, что газ поступает на начальную станцию с номером S и отгружается потребителям на конечной станции с номером F .

Президент потребовал от Правительства указать маршрут (т.е. линейную последовательность попарно соединённых газопроводами станций) перекачки газа из начальной станции на конечную, причём прибыльность этого маршрута должна быть максимальной. Под прибыльностью маршрута понимается суммарная прибыльность входящих в него газопроводов.

К сожалению, Президент не учёл того факта, что многие газопроводы изначальной сети уже давно прекратили существование, в результате чего может оказаться, что перекачка газа из начальной станции на конечную вообще невозможна...

Исходные данные

Первая строка содержит целые числа N ($2 \leq N \leq 500$) и M ($0 \leq M \leq 124750$). Каждая из следующих M строк содержит целые числа $A[i]$, $B[i]$ ($1 \leq A[i], B[i] \leq N$) и $C[i]$ ($1 \leq C[i] \leq 10000$) для соответствующего газопровода. Последняя строка содержит целые числа S и F ($1 \leq S, F \leq N$; $S \neq F$).

Результат

Если искомый маршрут существует, выведите его прибыльность. Иначе выведите "No solution".

Пример

исходные данные	результат
6 7 6 5 10 1 4 11 1 2 4 3 1 5 2 4 5 6 3 1 6 1 3 6 4	17

Замечания

В примере искомым маршрутом является маршрут $6 \rightarrow 3 \rightarrow 1 \rightarrow 4$.

Автор задачи: Дмитрий Ковалёв, Илья Гребнев, Никита Рыбак
Источник задачи: Timus Top Coders: Second Challenge

Рис. 3. Условие задачи 1450.

3.1 Краткое описание алгоритма

В основе решения лежит применение алгоритма **Форда-Беллмана** – алгоритм нахождения кратчайшего пути из заданной вершины s до всех остальных вершин взвешенного графа $G = (V, E)$. Если в графе G есть циклу с отрицательным суммарным весом, притом достижимые из s , тогда кратчайших путей не существует.

1. Входные данные: первая строка содержит целые числа N ($2 \leq N \leq 500$) и M ($0 \leq M \leq 124750$). Каждая из следующих M строк содержит целые числа $A[i]$, $B[i]$ ($1 \leq A[i], B[i] \leq N$) и $C[i]$ ($1 \leq C[i] \leq 10000$) для соответствующего газопровода. Последняя строка содержит целые числа S и F ($1 \leq S, F \leq N; S \neq F$).

2. создадим отдельный массив, в котором будет фиксироваться максимальная газопроводность на каждом шаге;

3. на каждом шаге рассмотрим все пути из каждой посещенной вершины, в случае нахождения большего значения максимальной газопроводности в вершину v , обновляем значение по этому индексу в массиве из прошлого пункта;

4. таким образом, в массиве из пункта 2 будут находиться максимальные значения газопроводности, -1 означает, что такого пути нет.

5. Выходные данные: если искомый маршрут существует, выведите его прибыльность. Иначе выведите "No solution".

3.2 Листинг

Листинг 2. Исходный код для 1450

```
1 #include <iostream>
2 #include <vector>
3
4
5 // structure to keep the edge
6 struct Way {
7     int a, b, w;
8 };
9
```

```
10
11 int main() {
12
13     std::ios_base::sync_with_stdio(0);
14     std::cin.tie(0);
15     std::cout.tie(0);
16
17     int n, m, s, f;
18     std::vector<Way> ways;
19     std::vector<int> gas_transfer(500, -1);
20
21     std::cin >> n >> m;
22
23     for (int i = 0; i < m; ++i) {
24         int a, b, w;
25         std::cin >> a >> b >> w;
26         ways.push_back({a - 1, b - 1, w});
27     }
28
29     std::cin >> s >> f;
30     —s;
31     —f;
32
33     gas_transfer[s] = 0;
34
35     for (int i = 0; i < n - 1; ++i) {
36         for (int j = 0; j < m; ++j) {
37
38             if (gas_transfer[ways[j].b] < gas_transfer[ways[j]
39                 .a] + ways[j].w && gas_transfer[ways[j].a] !=
40                 -1)
41                 gas_transfer[ways[j].b] = gas_transfer[ways[j]
42                     .a] + ways[j].w;
43         }
44     }
45
46     if (gas_transfer[f] != -1) std::cout << gas_transfer[f];
47     else std::cout << "No solution";
48
49     return 0;
50 }
```

3.3 Результат

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10648582	05:52:12 10 май 2024	mislDragon	1450. Российские газопроводы	Clang++ 17 x64	Accepted		0.171	2.792 KB

Рис. 4. Результат отправки задачи 1450.

4 Задача 1806

1806. Мобильные телеграфы

Ограничение времени: 3.0 секунды

Ограничение памяти: 256 МБ

Каждому бойцу 25-й стрелковой дивизии выдали новейшее средство связи — мобильный телеграф. С его помощью можно отправлять телеграммы командованию и боевым товарищам прямо на поле битвы. К сожалению, конструкция телеграфов ещё далека от совершенства — передавать сообщения можно только между некоторыми парами телеграфов.

Каждому устройству присвоен уникальный номер — строка из десяти десятичных цифр. С телеграфа a можно отправить сообщение на телеграф b только в том случае, если из номера a можно получить номер b , изменив в нём ровно одну цифру либо поменяв в нём две цифры местами. Время передачи сообщения с телеграфа a на телеграф b зависит от длины наибольшего общего префикса их номеров — чем больше его длина, тем быстрее передаётся сообщение.

Во время очередного сражения Анка из своей хорошо замаскированной позиции увидела небольшую группу белых, пытающуюся обойти обороняющихся красноармейцев с тыла. Какое минимальное время понадобится на доставку этой информации от Анки до Чапаева по телеграфу, возможно, с помощью других красноармейцев?

Исходные данные

В первой строке записано целое число n ($2 \leq n \leq 50000$) — количество бойцов в дивизии. Во второй строке через пробел в порядке невозрастания записаны десять целых чисел в пределах от 1 до 10000 — время передачи сообщения с одного телеграфа на другой при длине общего префикса их номеров, равной нулю, единице, двум, ..., девяти. Далее идут n строк, содержащие номера телеграфов, выданных бойцам дивизии. Номер телеграфа Анки указан первым, а номер телеграфа Чапаева — последним. Все номера телеграфов попарно различны.

Результат

Если передать Чапаеву сообщение нельзя, выведите в единственной строке «-1». В противном случае в первой строке выведите минимальное время, за которое можно доставить сообщение. Во второй строке выведите количество бойцов, которые участвуют в его доставке, а в третьей строке выведите через пробел их номера в порядке от Анки к Чапаеву. Бойцы 25-й дивизии занумерованы числами от 1 до n в том порядке, в котором описаны номера их мобильных телеграфов на входе. Если существует несколько способов передать сообщение за минимальное время, выведите любой из них.

Примеры

исходные данные	результат
5 100 10 10 10 1 1 1 1 1 1 9123493342 3123493942 9223433942 3223493942 9223433945	211 5 1 2 4 3 5
2 1 1 1 1 1 1 1 1 1 1 0123493342 0223433945	-1

Автор задачи: Павел Атнашев

Источник задачи: NEERC 2010, Четвертьфинал Восточного подрегиона

Рис. 5. Условие задачи 1806.

4.1 Краткое описание алгоритма

В основе решения задачи лежит **алгоритм Дейкстры** – алгоритм нахождения кратчайших путей от одной из вершин графа до всех остальных.

Каждой вершине из множества вершин V сопоставим метку — минимальное известное расстояние от этой вершины до стартовой вершины a . Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от a до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина u , имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u , назовём соседями этой вершины. Для каждого соседа вершины u , кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Работа алгоритма завершается, когда все вершины посещены.

1. Входные данные: в первой строке записано целое число n ($2 \leq n \leq 50000$) — количество бойцов в дивизии. Во второй строке через пробел в порядке невозрастания записаны десять целых чисел в пределах от 1 до 10000 — время передачи сообщения с одного телеграфа на другой при длине общего префикса их номеров, равной нулю, единице, двум, ..., девяти. Далее идут n строк, содержащие номера телеграфов, выданных бойцам дивизии. Номер телеграфа Анки указан первым, а номер телеграфа Чапаева — последним. Все номера телеграфов попарно различны.

2. Строим граф;

3. Находим кратчайшее расстояние, с помощью алгоритма Дейкстры

4. Выходные данные: если передать Чапаеву сообщение нельзя, выведите в единственной строке «-1». В противном случае в первой строке выведите минимальное время, за которое можно доставить сообщение. Во

второй строке выведите количество бойцов, которые поучаствуют в его доставке, а в третьей строке выведите через пробел их номера в порядке от Анки к Чапаеву. Бойцы 25-й дивизии занумерованы числами от 1 до n в том порядке, в котором описаны номера их мобильных телеграфов на входе. Если существует несколько способов передать сообщение за минимальное время, выведите любой из них.

4.2 Листинг

Листинг 3. Исходный код для 1806

```
1 #include <iostream>
2 #include <vector>
3 #include <unordered_map>
4 #include <queue>
5
6
7 const int inf = 1e9 + 7;
8 const int width = 10;
9 const int max_n = 50000;
10
11 long long pw[width]; // 1, 10, 100 ...
12 int cost[width]; // The costs of the prefix matching
13
14 struct Node {
15     std::vector<std::pair<int, Node*>> v; // Neighbors <cost,
        Node*>
16     Node* parent{}; // Of the shortest path
17     int d{}; // Shortest path cost
18     bool visited{};
19 } nodes[max_n];
20
21 std::unordered_map<long long, Node*> m;
22
23 int getDigit(long long num, int i) {
24     return (int) (num / pw[i] % 10);
25 }
26
27 long long setDigit(long long num, int i, int d) {
28     return num - ((long long) getDigit(num, i)) * pw[i] + d *
        pw[i];
29 }
```

```
30
31 int matchPrefix(long long num1, long long num2) {
32     int matched = 0;
33
34     for (int i = width - 1; i >= 0; i--) {
35         if (getDigit(num1, i) == getDigit(num2, i)) {
36             matched++;
37         } else {
38             break;
39         }
40     }
41
42     return matched;
43 }
44
45 // Adds this station into the graph at node #id
46 void add(long long num, int id) {
47     std::vector<std::pair<int, Node*>> v;
48
49     // Replace digit
50     for (int i = 0; i < width; i++) {
51         for (int d = 0; d < 10; d++) {
52             auto num2 = setDigit(num, i, d);
53
54             if (m.find(num2) != m.end()) {
55                 int c = cost[matchPrefix(num, num2)];
56                 v.emplace_back( c, (*m.find(num2)).second );
57             }
58         }
59     }
60
61     // Switch two digits
62     for (int i = 0; i < width; i++) {
63         for (int j = i + 1; j < width; j++) {
64             int di = getDigit(num, i);
65             int dj = getDigit(num, j);
66
67             auto num2 = setDigit(setDigit(num, j, di), i, dj);
68
69             if (m.find(num2) != m.end()) {
70                 int c = cost[matchPrefix(num, num2)];
71                 v.emplace_back( c, (*m.find(num2)).second );
72             }
73         }
74     }
75 }
```

```
73     }
74 }
75
76 m[num] = &nodes[id];
77
78 for(auto p : v){
79     p.second->v.emplace_back( p.first , &nodes[id] );
80     nodes[id].v.emplace_back( p.first , p.second );
81 }
82 }
83
84 // Dijkstras algorithm
85 void dijkstra(Node* start , int n) {
86
87     using pin = std::pair<int , Node*>;
88     std::priority_queue<pin , std::vector<pin> , std::greater<>>
89         q;
90
91     for (int i = 0; i < n; ++i) nodes[i].d = inf , nodes[i].
92         visited = false;
93
94     start->d = 0;
95
96     q.push( { 0 , start } );
97
98     while(!q.empty()) {
99
100         auto p = q.top();
101         q.pop();
102         auto node = p.second;
103
104         if (node->visited) continue;
105
106         node->visited = true;
107
108         for (auto it = node->v.begin(); it < node->v.end(); it
109             ++){
110             auto u = (*it).second;
111             int cur_cost = (*it).first;
112
113             if (!u->visited && u->d > node->d + cur_cost) {
114                 u->parent = node;
115                 u->d = node->d + cur_cost;
116             }
117         }
118     }
119 }
```



```
113         q.push( { u->d, u } );
114     }
115 }
116 }
117 }
118
119 int main() {
120
121     int n;
122     std::cin >> n;
123
124     long long b = 1;
125
126     for (long long & i : pw) i = b, b *= 10;
127
128     for (int & i : cost) std::cin >> i;
129
130     for (int i = 0; i < n; i++) {
131         long long num;
132         std::cin >> num;
133         add(num, i);
134     }
135
136     std::vector<Node*> result;
137     dijkstra(&nodes[0], n);
138
139     if (!nodes[n - 1].visited) {
140         std::cout << -1;
141         return 0;
142     }
143
144     std::cout << nodes[n - 1].d << "\n";
145
146     for (Node* node = &nodes[n - 1]; node; node = node->parent
147         ) result.push_back(node);
148
149     std::cout << result.size() << "\n";
150
151     for (auto item = result.rbegin(); item < result.rend(); ++
152         item)
153         std::cout << 1 + (*item->nodes) << " ";
```

4.3 Результат

Результаты проверки решений

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10652415	03:32:24 13 май 2024	mistDragon	1806. Мобильные телефоны	Clang++ 17 x64	Accepted		0.921	69 968 KB

Рис. 6. Результат отправки задачи 1806.

5 Вывод по работе

В ходе выполнения данной лабораторной работы были реализованы алгоритмы для решения задач 1650, 1450 и 1806.

В задаче 1450 был применен алгоритм **Форда-Беллмана**, в 1806 – алгоритм **Дейкстры**.