Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4
по “Алгоритмам и структурам данных”
Тимус

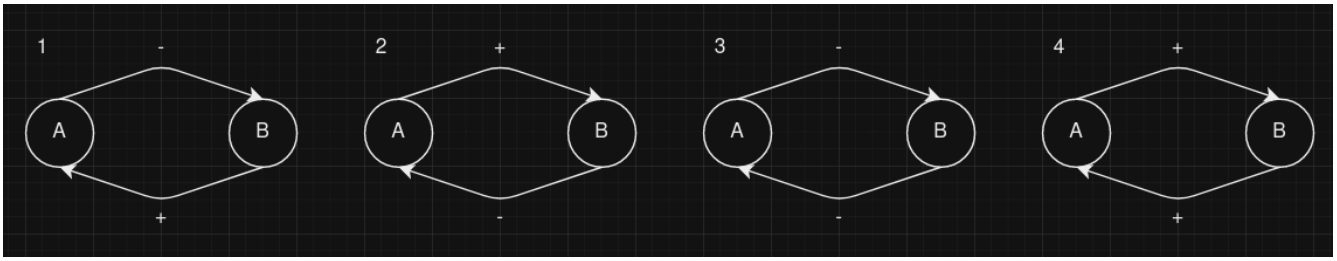
Выполнил:
Студент группы P3206
Сорокин А.Н.
Преподаватели:
Косяков М.С.
Тараканов Д.С.

Санкт-Петербург
2024

Задача №7 “Currency Exchange”
Решение за O(VE).

Условие задачи достаточно прямолинейное – определить, можно ли получить прибыль обменом валют. Однако данные в этой задаче можно представить в качестве не просто взвешенного ориентированного графа, а в качестве **знакового графа**.

Действительно, представим обычный обмен валют в виде такого графа:



Имеем 4 ситуации. Ситуации 1 и 2 не дают ничего – сбалансированный случай, когда мы получаем (приблизительно) столько же единиц валюты A, сколько у нас было до начала махинаций с обменом. Ситуация 3 – стандартная, отрицательный цикл, когда мы обменом уходим в убыток (в реальной жизни работает только так :D). Такая ситуация в рамках данной задачи нас не интересует. Ситуация 4 – то, что нам нужно, когда мы обнаружили положительный цикл и стали богаче (после обмена получили больше валюты A, чем у нас было).

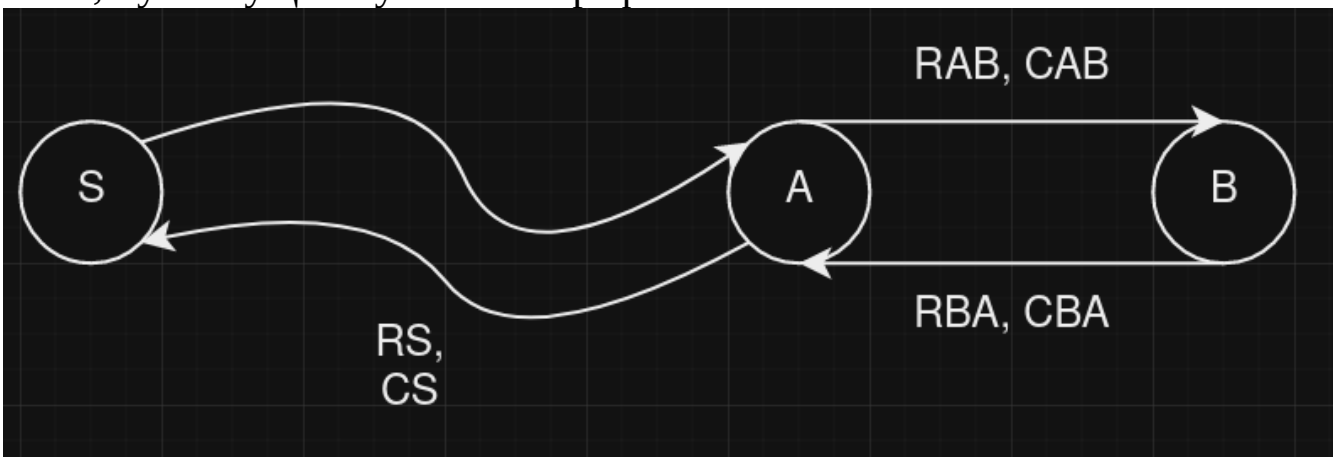
Возможно, интерпретировать задачу как знаковый граф – не совсем удачная идея, так как у нас есть числовые коэффициенты, ну да ладно.

Проведем некоторые наблюдения и обнаружим, что для получения положительного ответа на вопрос задачи – можно ли получить прибыль обменом валют – нам нет необходимости проходить всем путем из валюты S в валюту S: *нам достаточно найти любой положительный цикл.*

У меня нет строгого доказательства, поэтому постараюсь объяснить “на пальцах”:

В рамках объяснения будем положительным циклом также называть цикл (A, B, -), (B, A, +), где + перевешивает -, гарантируя прибыль при обмене $A \to B \to A$.

Итак, пусть существует такой граф:



где S – целевая (стартовая) валюта, A и B – валюты, образующие положительный цикл, RS и CS – курс и комиссия перевода в валюту S, RAB, CAB, RBA, CBA – коэффициенты для валют A и B соответственно.

Имеем V_A единиц валюты A. Пусть RS – наименьшее допустимое значение курса обмена, CS – наибольшее допустимое значение комиссии. Поскольку A и B образуют положительный цикл, то $V_A < ((V_A - CAB) * RAB - CBA) * RBA$, то есть при обмене $A \to B \to A$ мы получим $V_A' > V_A$.

Проведем еще одну операцию обмена: $V_A'' = ((V_A' - CAB) * RAB - CBA) * RBA$ и, поскольку цикл положительный, заметим, что $V_A'' > V_A'$. Таким образом, совершив n операций обмена $A \to B \to A$, где $n \to \infty$, мы получим $V_A^{(n)} \gg V_A$.

У нас было V_S единиц валюты S, после перехода в валюту A мы получили V_A, и очевидно, что без помощи пожительного цикла $A \to B \to A$ при переходе из валюты A в валюту S мы получим $V_{S \leftarrow A}$ такое, что $V_{S \leftarrow A} \ll V_S$, что даст сильный убыток. Однако, если совершить n операций обмена в положительном цикле, где $n \to \infty$, мы придем к этому:

при $f(x)=((x-CAB)*RAB-CBA)*RBA \rightarrow (\lim_{n\to\infty} f^n(V_A)-CS)*RS>V_S$.

Следовательно, *если существует положительный цикл, то обменом валют можно получить прибыль*.

Для поиска положительного цикла воспользуемся **алгоритмом Беллмана-Форда**, немного измененнным под условие нашей задачи – вместо поиска кратчайшего пути во взвешенном графе будем пытаться найти источник прибыли. Данный алгоритм позволит нам запоминать состояния денежных единиц и справляться с отрицательными циклами – то, что не может, например, алгоритм Дейкстра.

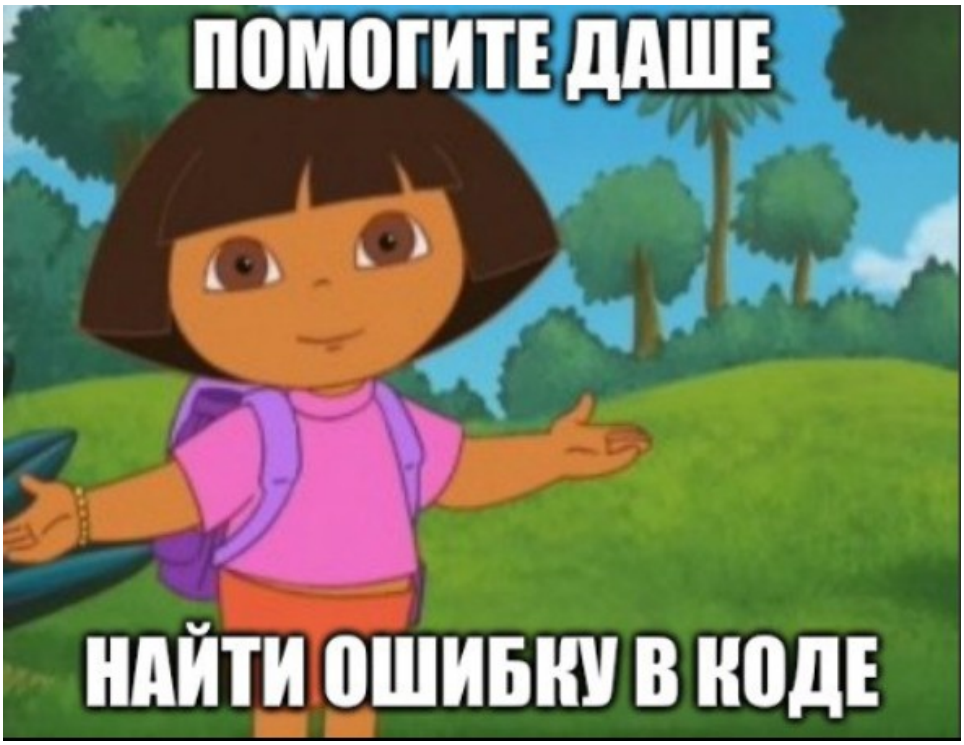
Код решения:

```
#include <fstream>
#include <iostream>
#include <vector>
using namespace std;

struct Edge {
  int A;
  int B;
  double R;
  double C;
};
const double EPS = 0.00000001;
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  int N, M, S;
  double V;
  cin >> N >> M >> S >> V;
  vector<Edge> graph;
  vector<double> capital(101);
  capital[S] = V;

  for (int i = 0; i < M; ++i) {
```

```
    int A, B;
    double RAB, CAB, RBA, CBA;
    cin >> A >> B >> RAB >> CAB >> RBA >> CBA;
    graph.push_back({A, B, RAB, CAB});
    graph.push_back({B, A, RBA, CBA});
  }
  // bellman-ford algorithm
  for (int i = 0; i < N - 1; ++i) {
    for (int j = 0; j < graph.size(); ++j) {
      if (capital[graph[j].B] -
              (capital[graph[j].A] - graph[j].C) * graph[j].R <
          EPS) {
        capital[graph[j].B] = (capital[graph[j].A] - graph[j].C) * graph[j].R;
      }
    }
  }

  for (int i = 0; i < graph.size(); ++i) {
    // if somewhere by exchanging currency you get a better value, means you
can
    // get richer
    if ((capital[graph[i].A] - graph[i].C) * graph[i].R - capital[graph[i].B] >
        EPS) {
      cout << "YES";
      return 0;
    }
  }
  cout << "NO";
  return 0;
}
```

Это была классная задача, *надеюсь, следующая задача не создаст много проблем…*

Задача №8 “Мобильные телеграфы”



Последние попытки

Автор: vtalgo24 . Sorokin . 367548 • Задача: Мобильные телеграфы

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10656148	23:37:05 16 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Accepted		0.781	4 976 КБ
10656142	23:17:46 16 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	3	0.015	416 КБ
10656140	23:16:14 16 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	3	0.001	396 КБ
10656139	23:13:35 16 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	3	0.015	396 КБ
10656137	23:10:53 16 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	3	0.001	392 КБ
10655474	11:23:20 16 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	2	0.015	324 КБ
10653853	17:18:46 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	2	0.015	376 КБ
10653848	17:17:20 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	380 КБ
10653844	17:14:56 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	400 КБ
10653837	17:13:31 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	380 КБ
10653831	17:11:49 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	404 КБ
10653821	17:05:10 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	384 КБ
10653815	16:58:15 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	2	0.001	332 КБ
10653814	16:55:40 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Compilation error			
10653809	16:52:46 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	380 КБ
10653806	16:50:02 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	384 КБ
10653795	16:40:51 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	380 КБ
10653751	16:07:56 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	388 КБ
10653748	16:03:55 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.015	380 КБ
10653704	14:40:52 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	384 КБ
10653699	14:37:05 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	384 КБ
10653692	14:32:01 14 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	380 КБ
10652047	21:02:59 12 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.015	388 КБ
10652044	21:01:54 12 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Visual C++ 2022 x64	Wrong answer	1	0.001	272 КБ
10652031	20:48:44 12 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	G++ 13.2 x64	Wrong answer	1	0.015	420 КБ
10652028	20:48:26 12 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	GCC 13.2 x64	Compilation error			
10652027	20:47:24 12 май 2024	vtalgo24 . Sorokin . 367548	1806. Мобильные телеграфы	Clang++ 17 x64	Wrong answer	1	0.001	388 КБ

Решение за O((V + E) log V).

~~Если не считать приколы C++~~, задача предельно простая. Имеем набор телеграфов с уникальными номерами, требуется определить наиболее дешевый путь для сообщения от 1 телеграфа к последнему (или вывести -1, если такого пути нет).

Данную задачу можно решить используя **алгоритм Дейкстры**, так как он достаточно хорошо справляется с поиском кратчайшего пути. Главной проблемой данной задачи является сам граф – задан он в неявном виде, и поэтому для каждой вершины необходимо искать вершины, смежные с данной, и для каждого ребра считать вес. Однако, следует заметить, что номера телеграфов имеют фиксированную длину, равную 10, отсюда можно смело заключить, что поиск соседей будет работать за O(1) :)

Главное, что необходимо учитывать – для операций с номерами телеграфов необходимо использовать числа, а не строки, так как иначе время работы и, тем более, затраты по памяти могут возрасти.

То, почему алгоритм Дейкстры работает, объясняется в решении задачи “Цивилизация”. Ребра графа в данной задаче всегда имеют положительный вес, граф имеет конечное число вершин.

Следовательно, решение такое: создаем priority_queue, начиная поиск с первого телеграфа: в цикле ищем соседей, считаем длину пути, как приходим к конечному телеграфу, останавливаемся. Далее через bAcKtRaCkInG восстанавливаем путь от начальной вершины к конечной (если конечная вершина была достигнута, иначе выводим -1 и выходим).

Код решения:
```
#include <cmath>
#include <fstream>
#include <iostream>
#include <queue>
```

```
#include <set>
#include <string>
#include <unordered_map>

using namespace std;

struct Vertex {
  long long cost;
  int id;
};
bool operator<(const Vertex &a, const Vertex &b) { return a.cost > b.cost; }

int getDif(long long a, long long b) {
  int c = 0;
  while (a != b) {
    a /= 10;
    b /= 10;
    ++c;
  }
  return 10 - c;
}

int main() {
  // ifstream infile("input.txt");
  int n;
  cin >> n;
  vector<long long> difcount(10);
  for (int i = 0; i < 10; ++i) {
    cin >> difcount[i];
  }

  vector<long long> telegraphs(n);
  unordered_map<long long, int> telegraphToIndex;
  for (int i = 0; i < n; ++i) {
    long long t;
    cin >> t;
```

```
  telegraphs[i] = t;
  telegraphToIndex[t] = i;
}

priority_queue<Vertex> pq;
pq.push({0, 0});

vector<Vertex> visited(n, {-1, -1});
for (int i = 0; i < n; ++i) {
  visited[i] = {-1, -1};
}
visited[0] = {0, 0};
while (!pq.empty()) {
  auto curr = pq.top();
  pq.pop();
  if (curr.id == n - 1)
    break;
  long long old_val = telegraphs[curr.id];
  // find linked telegraphs by iterating
  multiset<Vertex> linkedTelegraphs;
  for (int i = 0; i < 10; ++i) {
    long long p1 = pow(10, i);
    long long r1 = (old_val / p1) % 10;
    for (int j = 0; j < 10; ++j) {
      long long new_val = old_val - r1 * p1 + j * p1;
      if (old_val == new_val)
        continue;
      auto next = telegraphToIndex.find(new_val);
      if (next != telegraphToIndex.end()) {
        linkedTelegraphs.insert(
            {difcount[getDif(old_val, new_val)], next->second});
      }
    }
    for (int j = i + 1; j < 10; ++j) {
      long long p2 = pow(10, j);
      long long r2 = (old_val / p2) % 10;
```

```
      long long new_val = old_val - r1 * p1 + r2 * p1 - r2 * p2 + r1 * p2;
      if (old_val == new_val)
        continue;
      auto next = telegraphToIndex.find(new_val);
      if (next != telegraphToIndex.end()) {
        linkedTelegraphs.insert(
            {difcount[getDif(old_val, new_val)], next->second});
      }
    }
  }
  // iterate through linked telegraphs
  for (auto next : linkedTelegraphs) {
    long long new_cost = curr.cost + next.cost;
    if (visited[next.id].cost == -1 || visited[next.id].cost > new_cost) {
      visited[next.id].cost = new_cost;
      visited[next.id].id = curr.id;
      pq.push({new_cost, next.id});
    }
  }
}

if (visited[n - 1].cost == -1) {
  cout << -1 << endl;
} else {
  cout << visited[n - 1].cost << endl;
  int c = 1;
  for (int i = n - 1; i != 0; i = visited[i].id) {
    c++;
  }
  cout << c << endl;

  vector<int> res;
  for (int i = n - 1; i != 0; i = visited[i].id) {
    res.push_back(i + 1);
  }
  res.push_back(1);
```

```
    for (int i = c - 1; i >= 0; --i) {
      cout << res[i] << " ";
    }
    cout << endl;
  }

  return 0;
}
```