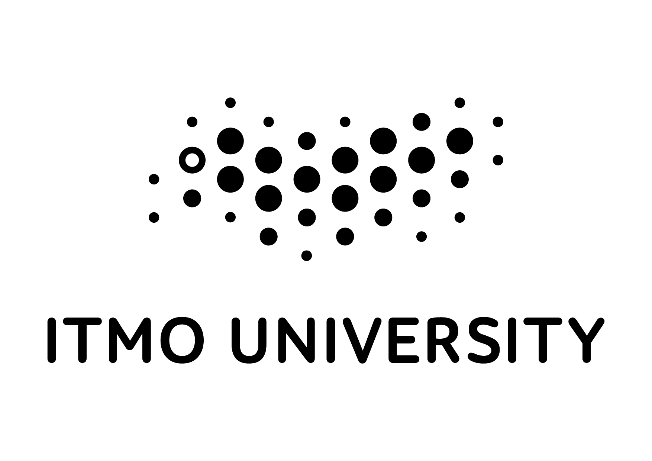
Санкт-Петербургский национальный исследовательский университет ИТМО

Факультет систем управления и робототехники



Алгоритмы и структуры данных.

Отчет №6.

Выполнил студент гр. R32362

Лаптев Максим Сергеевич

Санкт-Петербург 2023

**Содержание**

1. Цель
2. Задача №1628 – Белые полосы
3. Задача №1450 – Российские газопроводы
4. Задача №1160 – Network
5. Вывод

**Цель**

Решить данные задачи, написав код, работа которого удовлетворяет условиям

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Исходный код:

#include <iostream>

#include <set>

using namespace std;

int res = 0;

set<int> count(const set<int>& my\_set, int size){

int prev = 0;

set<int> single\_el;

for(int curr : my\_set){

if (curr - prev > 2) res = res + 1;

else if (curr - prev == 2) single\_el.insert(curr - 1);

prev = curr;

}

if (size - prev > 1) res = res + 1;

else if (size - prev == 1) single\_el.insert(size);

return single\_el;

}

int main() {

int m, n, k;

int x, y;

cin >> m >> n >> k;

set<int> rows[m], columns[n];

while(k > 0){

cin >> x >> y;

rows[x - 1].insert(y);

columns[y - 1].insert(x);

k--;

}

for(set<int>& row : rows) {

row = count(row, n);

}

for(set<int>& column : columns) {

column = count(column, m);

}

for (int i = 0; i < m; i++){

for (int el : rows[i]){

if (columns[el - 1].find(i + 1) != columns[el - 1].end()) res++;

}

}

cout << res;

return 0;

}

Будем сохранять входные данные в множества строк и колонок. Пройдемся по каждому из них, посчитаем количества белых отрезков и добавим их в переменную result. Отдельного рассмотрения заслуживает случай одноклеточного отрезка. В некоторых случаях он считается, в некоторых – нет. Посчитаем их все и рассмотрим в отдельности.

Язык программирования: C++

Результат



Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Исходный код:

#include <set>

#include <iostream>

#include <vector>

using namespace std;

int main() {

int n, m;

int s, f; // Start, Finish

int a, b, c; // I, J, cost

cin >> n >> m;

vector <vector <pair <int, int> > > g(n);

for (int i = 0; i < m; i++){

cin >> a >> b >> c;

a--; b--;

g[a].push\_back(make\_pair(b, c));

}

cin >> s >> f;

s--; f--;

vector<int> dist(n, -1);

dist[s] = 0;

set<pair<int, int> > unused;

unused.insert(make\_pair(0, s));

while (!unused.empty())

{

int i = unused.begin()->second;

unused.erase(unused.begin());

for (auto edge : g[i])

{

int j = edge.first;

int wt = edge.second;

if (dist[i] + wt > dist[j])

{

unused.erase(make\_pair(dist[j], j));

dist[j] = dist[i] + wt;

unused.insert(make\_pair(dist[j], j));

}

}

}

if (dist[f] == -1) cout << "No solution";

else cout << dist[f];

return 0;

}

Для решения использовался алгоритм Дейкстры, только с условием нахождения не минимального, а максимального пути. Это жадный алгоритм, главная суть которого заключается в этой строчке   
if (dist[i] + wt > dist[j])

Язык программирования: С++

Результат



Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

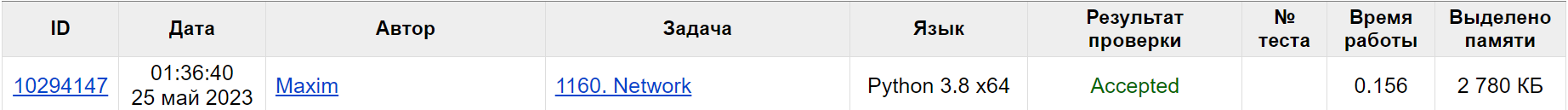
Исходный код:

n, m = map(int, input().split())  
lengths = []  
for i in range(m):  
 x, y, l = map(int, input().split())  
 lengths.append((x, y, l))  
lengths = sorted(lengths, key=lambda x: x[2])  
rebra = []  
was = set()  
max\_len = 0  
answer = set()  
   
for el in lengths:  
 flag = True  
 flag2 = False  
 for rebro in rebra:  
 if el[0] in rebro and el[1] in rebro:  
 flag = False  
 elif el[0] in rebro or el[1] in rebro:  
 flag2 = True  
 if flag:  
 if not flag2:  
 rebra.append({el[0], el[1]})  
 answer.add((el[0], el[1]))  
 max\_len = max(max\_len, el[2])  
 continue  
 to\_add = None  
 for rebro in rebra:  
 if el[0] in rebro:  
 rebro.add(el[1])  
 answer.add((el[0], el[1]))  
 max\_len = max(max\_len, el[2])  
 if to\_add is None:  
 to\_add = rebro  
 continue  
 rebra.remove(to\_add)  
 to\_add = to\_add.union(rebro)  
 rebra.append(to\_add)  
 rebra.remove(rebro)  
 break  
 if el[1] in rebro:  
 rebro.add(el[0])  
 answer.add((el[0], el[1]))  
 max\_len = max(max\_len, el[2])  
 if to\_add is None:  
 to\_add = rebro  
 continue  
 rebra.remove(to\_add)  
 to\_add = to\_add.union(rebro)  
 rebra.append(to\_add)  
 rebra.remove(rebro)  
 break  
print(max\_len)  
print(len(answer))  
for el in answer:  
 print(\*el)

Суть задачи заключалась в том, чтобы построить минимальное остовное дерево. Для этого я использовал алгоритм Крускала. Сначала все пути сортируются по длине. Далее по одному добавляются пути с наименьшей длиной и при этом чтобы они не создавали циклов. Пути добавляются в массив непересекающихся множеств. При этом если добавляется путь, первый конец которого в одном множестве, а второй – в другом, то эти множества объединяются.

Язык программирования: python

Результат:



**Вывод**

В результате проделанной работы были успешно решены 3 задачи, используя язык программирования python . Работа программ выдает правильный ответ, а также удовлетворяет условиям задачи (проходит по времени и памяти).