МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Интеллектуальные информационные технологии»

Лабораторная работа №3

По дисциплине: «***Алгоритмы и структуры данных***»

Тема: «***Нахождение минимального остовного дерева***

***связанного неориентированного графа***»

**Выполнил:** Антонюк Н.А.

**Группа:** ПО-11

**Проверила:** Глущенко Т.А.

Брест 2023

Алгоритм Прима

#include <iostream>

#include <vector>

#include <queue>

#include <utility>

#include <climits>

using namespace std;

#define V 8

int minKey(int key[], bool mstSet[]) {

int min = INT\_MAX, min\_index;

for (int v = 0; v < V; v++) {

if (mstSet[v] == false && key[v] < min) {

min = key[v];

min\_index = v;

}

}

return min\_index;

}

void printMST(int parent[], vector<vector<int>> graph) {

cout << "Edge \tWeight\n";

for (int i = 1; i < V; i++) {

cout << parent[i] << " - " << i << " \t" << graph[i][parent[i]] << " \n";

}

}

void primMST(vector<vector<int>> graph) {

int parent[V];

int key[V];

bool mstSet[V];

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

key[i] = INT\_MAX;

mstSet[i] = false;

}

key[0] = 0;

parent[0] = -1;

for (int count = 0; count < V - 1; count++) {

int u = minKey(key, mstSet);

mstSet[u] = true;

for (int v = 0; v < V; v++) {

if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v]) {

parent[v] = u;

key[v] = graph[u][v];

}

}

}

printMST(parent, graph);

}

int main() {

vector<vector<int>> graph = { {0, 4, 6, 0, 0, 0, 0, 0},

{4, 0, 0, 1, 3, 0, 0, 0},

{6, 0, 0, 1, 0, 3, 0, 0},

{0, 1, 1, 0, 3, 4, 0, 0},

{0, 3, 0, 3, 0, 0, 5, 2},

{0, 0, 3, 4, 0, 0, 0, 5},

{0, 0, 0, 0, 5, 0, 0, 1},

{0, 0, 0, 0, 2, 5, 1, 0} };

primMST(graph);

return 0;

}

Алгоритм Крускала

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

struct Edge {

int src, dest, weight;

};

vector<Edge> getEdgesFromGraph(const vector<vector<int>>& graph) {

vector<Edge> edges;

for (int i = 0; i < graph.size(); i++) {

for (int j = i; j < graph[i].size(); j++) {

if (graph[i][j] != 0) {

Edge e = { i, j, graph[i][j] };

edges.push\_back(e);

}

}

}

return edges;

}

int find(vector<int>& parent, int i) {

if (parent[i] == i) {

return i;

}

return find(parent, parent[i]);

}

void Union(vector<int>& parent, vector<int>& rank, int x, int y) {

int xroot = find(parent, x);

int yroot = find(parent, y);

if (rank[xroot] < rank[yroot]) {

parent[xroot] = yroot;

}

else if (rank[xroot] > rank[yroot]) {

parent[yroot] = xroot;

}

else {

parent[yroot] = xroot;

rank[xroot]++;

}

}

void KruskalMST(const vector<vector<int>>& graph) {

int V = graph.size();

vector<Edge> edges = getEdgesFromGraph(graph);

sort(edges.begin(), edges.end(), [](const Edge& a, const Edge& b) -> bool {

return a.weight < b.weight;

});

vector<Edge> result;

vector<int> parent(V);

vector<int> rank(V, 0);

for (int v = 0; v < V; v++) {

parent[v] = v;

}

int i = 0;

int e = 0;

while (e < V - 1 && i < edges.size()) {

Edge next\_edge = edges[i++];

int x = find(parent, next\_edge.src);

int y = find(parent, next\_edge.dest);

if (x != y) {

result.push\_back(next\_edge);

Union(parent, rank, x, y);

e++;

}

}

cout << "Following are the edges in the constructed MST\n";

for (i = 0; i < result.size(); i++) {

cout << result[i].src << " -- " << result[i].dest << " == " << result[i].weight << endl;

}

}

int main() {

vector<vector<int>> graph = { {0, 4, 6, 0, 0, 0, 0, 0},

{4, 0, 0, 1, 3, 0, 0, 0},

{6, 0, 0, 1, 0, 3, 0, 0},

{0, 1, 1, 0, 3, 4, 0, 0},

{0, 3, 0, 3, 0, 0, 5, 2},

{0, 0, 3, 4, 0, 0, 0, 5},

{0, 0, 0, 0, 5, 0, 0, 1},

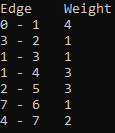
{0, 0, 0, 0, 2, 5, 1, 0} };

KruskalMST(graph);

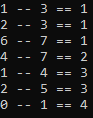
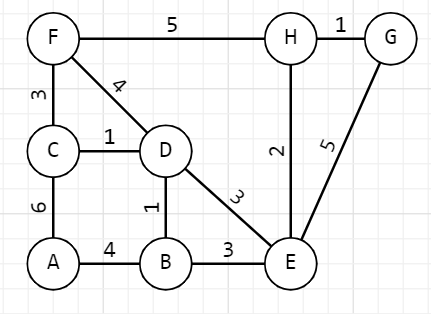
return 0;

}

Минимальное остовное дерево(Алгоритм Прима):



Минимальное остовное дерево(Алгоритм Крускаля):

class Solution {

public:

    int computeArea(int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2) {

                 int a1=abs(ax2-ax1) \* abs(ay2-ay1);

                 int a2=abs(bx2-bx1) \* abs(by2-by1);

                 int p=min(ax2,bx2);

                 int q=max(ax1,bx1);

                 int r= min(ay2,by2);

                 int s=max(ay1,by1);

                 int inter=(p-q)\*(r-s);

                cout<<a1<<" "<<a2<<" "<<inter<<endl;

            if((bx1)>(ax2)||(by1)>(ay2)||(bx2)<(ax1)||(by2)<(ay1))

            {

                return a1+a2;

            }

            else

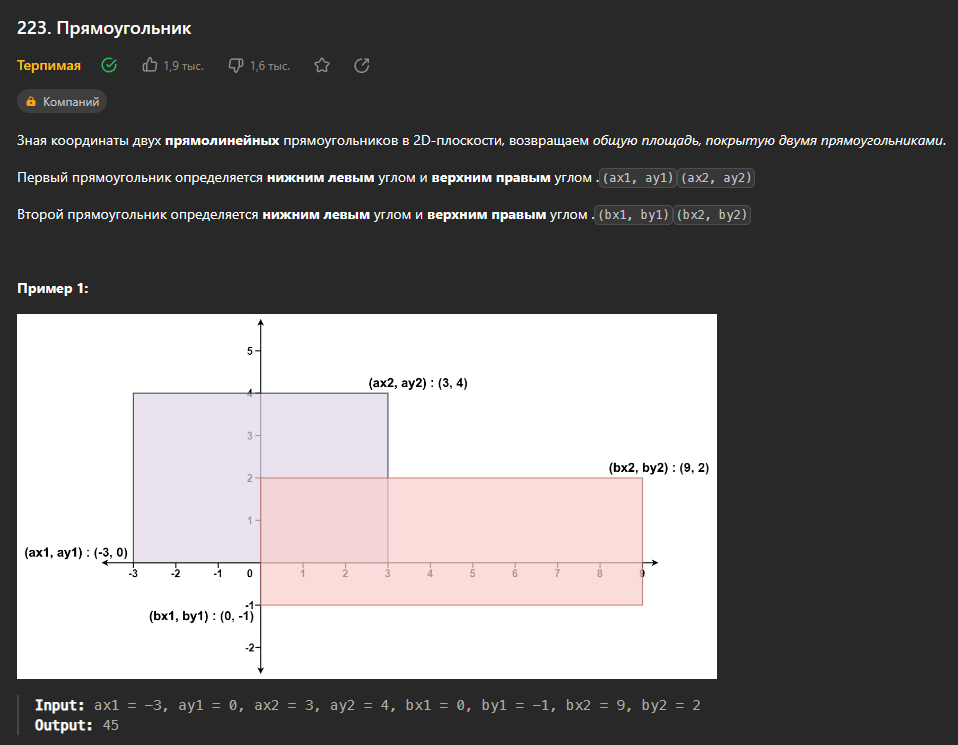
            {

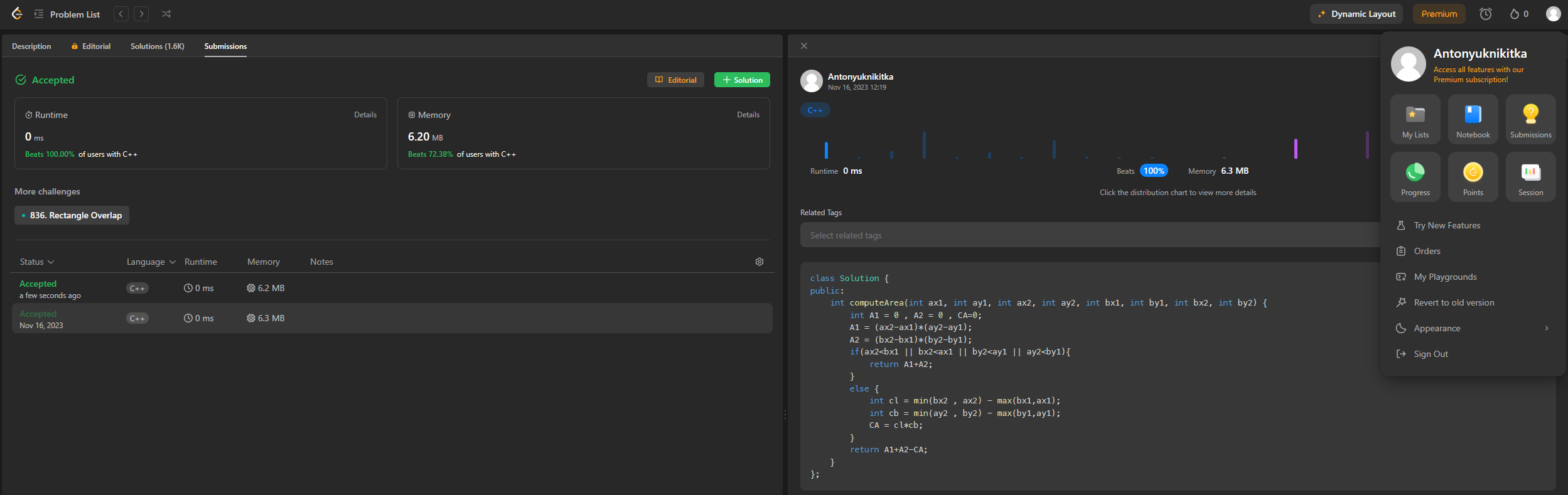
            return a1+a2-inter;

            }

    }

**Задача на LEETCODE:**





**Алгоритм задачи:**

class Solution {

public:

    int computeArea(int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2) {

                 int a1=abs(ax2-ax1) \* abs(ay2-ay1);

                 int a2=abs(bx2-bx1) \* abs(by2-by1);

                 int p=min(ax2,bx2);

                 int q=max(ax1,bx1);

                 int r= min(ay2,by2);

                 int s=max(ay1,by1);

                 int inter=(p-q)\*(r-s);

                cout<<a1<<" "<<a2<<" "<<inter<<endl;

            if((bx1)>(ax2)||(by1)>(ay2)||(bx2)<(ax1)||(by2)<(ay1))

            {

                return a1+a2;

            }

            else

            {

            return a1+a2-inter;

            }

    }

};

**Вопросы:**

1. *Для какого графа определяет число остовных деревев формула Кэли?*

Формула Кэли применяется к помеченным графам, в котором каждому ребру присвоено уникальное значение или метка

1. *Подсчитать по формуле Кэли и нарисовать число остовных деревьев для n = 3*

Для n=3, формула Кэли утверждает, что число остовных деревьев равно 33-1 = 32 = 9

1. *N=4*

Для n=4 число остовных деревьев будет 44-1 = 43 = 64

1. *Какое остовное дерево может находжиться алгоритмом дейкстры?*

Алгоритм Дейкстры находит кратчайшие пути от одной из вершин графа до всех остальных. Он не обязательно находит остовное дерево, но если все вершины соединены и алгоритм начинается из произвольной вершины, то результат будет остовным деревом

1. *Может ли быть несколько одинаковых минимальных остовных деревьев?*

Да, может быть несколько минимальных остовных деревьев в графе. Если у нескольких рёбер одинаковый вес, то существует несколько вариантов выбора при построении минимального остовного дерева

1. *Приведите оценку временной сложности обоих алгоритмов*

*Алгоритм Прима*: O((V+E)logV) где V - количество вершин, Е - количесто ребер

*Алгоритм Крускала*: O(E\*logE) или O(E\*logV), в зависимости от того, как реализована структура данных для наборы

1. *Являются ли алгоритмы жадными, если да, то почему?*

Алгоритмы Прима и Крускала являются жадными алгоритмами. Они принимают локальные оптимальные решения на каждом шаге в надежде достижения глобального оптимального результата

1. *Какие структуры данных позволяют оптимизировать алгоритмы*

* В алгоритме прима используется очередь с приоритетом чтобы эффективно находить ребро минимального веса

В алгоритме Крускала используется структура данных Disjoint Set (набор непересекающихся множеств) для эффективного определения циклов и объединения множества.