# Московский авиационный институт (национальный исследовательский университет)

## Факультет информационных технологий и прикладной математики

## Кафедра вычислительной математики и программирования


**Курсовой проект по курсу «Дискретный анализ»**


Студент: И. П. Попов
Преподаватель: С. А. Сорокин
Группа: М8О-306Б

Дата:
Оценка:
Подпись:

**Москва, 2022**

# Курсовой проект

**Задача:** Реализуйте алгоритм A* для графа на решетке.

**Формат ввода**
В первой строке вам даны два числа n и m ($1 \leqslant n \leqslant 10^4$, $1 \leqslant m \leqslant 10^5$) — количество вершин и рёбер в графе.

В следующих n строках вам даны пары чисел x y($-10^9 \leqslant x, y \leqslant 10^9$), описывающие положение вершин графа в двумерном пространстве.
В следующих m строках даны пары чисел в отрезке от 1 до n, описывающие рёбра графа.

Далее дано число q ($1 \leqslant q \leqslant 300$) и в следующих q строках даны запросы в виде пар чисел a b ($1 \leqslant a, b \leqslant n$) на поиск кратчайшего пути между двумя вершнами.

**Формат вывода**

В ответ на каждый запрос выведите единственное число — длину кратчайшего пути между заданными вершинами с абсолютной либо относительной точностью $10^{-6}$, если пути между вершинами не существует выведите –1.

# 1 Описание

**Поиск A*** (произносится «А звезда» или «А стар», от англ. A star) — в информатике и математике, алгоритм поиска по первому наилучшему совпадению на графе, который находит маршрут с наименьшей стоимостью от одной вершины (начальной) к другой (целевой, конечной).

Порядок обхода вершин определяется эвристической функцией «расстояние + стоимость» (обычно обозначаемой как f(x)). Эта функция — сумма двух других: функции стоимости достижения рассматриваемой вершины (x) из начальной (обычно обозначается как g(x) и может быть как эвристической, так и нет), и функции эвристической оценки расстояния от рассматриваемой вершины к конечной (обозначается как h(x)).

Функция h(x) должна быть допустимой эвристической оценкой, то есть не должна переоценивать расстояния к целевой вершине. Например, для задачи маршрутизации h(x) может представлять собой расстояние до цели по прямой линии, так как это физически наименьшее возможное расстояние между двумя точками.

Этот алгоритм был впервые описан в 1968 году Питером Хартом, Нильсом Нильсоном и Бертрамом Рафаэлем. Это по сути было расширение алгоритма Дейкстры, созданного в 1959 году. Новый алгоритм достигал более высокой производительности (по времени) с помощью эвристики. В их работе он упоминается как «алгоритм A». Но так как он вычисляет лучший маршрут для заданной эвристики, он был назван A*.

## 2 Исходный код

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

#define endl '\n';

struct coords {
    double x;
    double y;
};

inline double euclid_dist(ll u, ll v, const vector<coords>& vert) {
    coords v1 = vert[u];
    coords v2 = vert[v];

    return sqrt((v1.x - v2.x) * (v1.x - v2.x) + (v1.y - v2.y) * (v1.y - v2.y));
}

struct node {
    ll ind;
    double path;

    node(const ll _ind, const double _path) {
        ind = _ind;
        path = _path;
    }

    friend bool operator< (const node& v1, const node& v2) {
        if (v1.path != v2.path)
            return v1.path > v2.path;
        return v1.ind < v2.ind;
    }
};

double solve(const ll start, const ll finish, const vector<coords> &vert, const vector
    <vector<ll>> &graph) {
    vector<double> res(vert.size(), DBL_MAX);//     ,
    vector<double> dist(vert.size(), DBL_MAX);
    priority_queue<node> pq;

    res[start] = 0; //
    dist[start] = res[start] + euclid_dist(start, finish, vert);
    pq.push(node(start, dist[start]));

    while (!pq.empty()) {
        node tmp = pq.top();
```

```
47          pq.pop();
48
49          if (tmp.ind == finish)//
50              break;
51
52          if (tmp.path > dist[tmp.ind])
53              continue;
54
55          for (auto next : graph[tmp.ind]){//
56
57              if (res[next] == DBL_MAX || (res[tmp.ind] + euclid_dist(tmp.ind, next, vert
                    )) < res[next]){
58                  res[next] = res[tmp.ind] + euclid_dist(tmp.ind, next, vert);
59                  dist[next] = res[next] + euclid_dist(next, finish, vert);
60
61                  pq.push(node(next, dist[next]));
62              }
63          }
64      }
65      return res[finish];
66  }
67
68  int main() {
69      ios_base::sync_with_stdio(false);
70      cin.tie(nullptr); cout.tie(nullptr);
71
72      ll n, m;
73      cin >> n >> m;
74      vector<coords> vert(n);
75      vector<vector<ll>> graph(n);
76
77      for (ll i = 0; i < n; ++i)
78          cin >> vert[i].x >> vert[i].y;
79
80      for (ll i = 0; i < m; ++i) {
81          ll u, v;
82          cin >> u >> v;
83          u--; v--;
84          graph[u].push_back(v);
85          graph[v].push_back(u);
86      }
87
88      ll q;
89      cin >> q;
90
91      while(q--) {
92          ll u, v;
93          cin >> u >> v;
94          u--; v--;
```

```
95          double ans = solve(u, v, vert, graph);
96          if (ans != DBL_MAX){
97              cout << fixed << setprecision(6) << ans << endl;
98          }
99          else{
100             cout << -1 << endl;
101         }
102     }
103     return 0;
104 }
```

# 3 Консоль

console input:

```
4 5
0 0
1 1
-1 1
0 2
1 2
1 3
2 4
3 4
1 4
1
1 4
```

console output:

```
2.000000
```

# 4 Оценка сложности

Временная сложность алгоритма A* зависит от эвристики. В худшем случае, число вершин, исследуемых алгоритмом, растёт экспоненциально по сравнению с длиной оптимального пути, но сложность становится полиномиальной, когда эвристика удовлетворяет следующему условию:

$|h(x) - h^*(x)| \leq O(\log h^*(x))|h(x) - h^*(x)| \leq O(\log h^*(x))$;
где h* — оптимальная эвристика, то есть точная оценка расстояния из вершины x к цели. Другими словами, ошибка h(x) не должна расти быстрее, чем логарифм от оптимальной эвристики.

Но ещё большую проблему, чем временная сложность, представляют собой потребляемые алгоритмом ресурсы памяти. В худшем случае ему приходится помнить экспоненциальное количество узлов.

# 5 Выводы

Алгоритм A* является потомком алгорима Дейкстры, который в силу своей жадности обладает таким недостатком, как выбор неоптимального пути до финишной вершины.

Алгоритм A* и допустим, и обходит при этом минимальное количество вершин, благодаря тому, что он работает с «оптимистичной» оценкой пути через вершину. Оптимистичной в том смысле, что, если он пойдёт через эту вершину, алгоритм уверен, что реальная стоимость результата будет равна этой оценке, но никак не меньше.

Когда A* завершает поиск, он, согласно определению, нашёл путь, истинная стоимость которого меньше, чем оценка стоимости любого пути через любой открытый узел. Но поскольку эти оценки являются оптимистичными, соответствующие узлы можно без сомнений отбросить. Иначе говоря, A* никогда не упустит возможности минимизировать длину пути, и потому является допустимым.

Недостатком алгоритма A* является необходимость придумать эвристику. Однако, во многих задачах эта эвристика находится достаточно легко и естественно.

# Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))