

Grafos 2

Projeto de Algoritmos Turma 01 - 2023/1

Isabella Carneiro 18/0019066

Rafael Fernandes Amancio 190036940



Exercícios

- Roteadores - Nível 3
- Mania de par - Nível 6
- Inversão - Nível 7
- LeetCode - Exercício 1584

Beecrowd

1774

Roteadores

Roteadores

Por Carlos Andrade, UFMS  Brazil

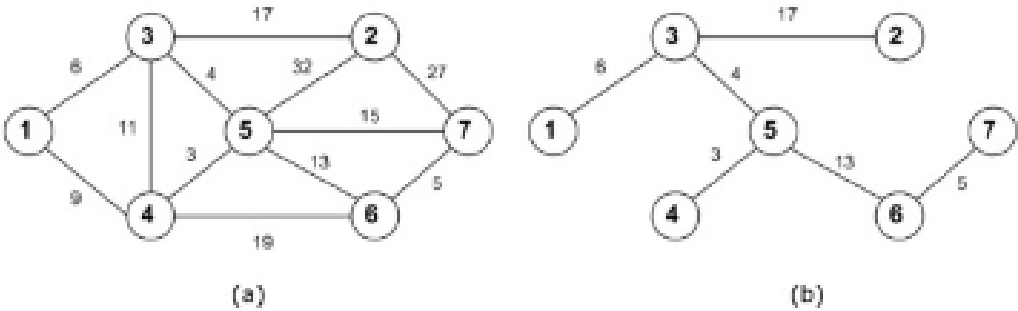
Timelimit: 4

Bruno é o responsável por configurar os roteadores de uma empresa. Os roteadores transmitem os dados entre si através dos cabos de internet. Os dados transmitidos podem trafegar por uma ou mais rotas para serem entregues ao destinatário.

O preço dos cabos de rede utilizados nos roteadores da empresa pode chegar a ser muito caro, e a empresa precisa cortar gastos. Pensando nisso a empresa decidiu fazer algumas alterações na infra-estrutura de redes.

Bruno deve modificar a infra-estrutura da rede da empresa de forma com que todos os roteadores consigam transmitir dados entre si e exista somente uma rota entre cada par de roteadores, economizando o máximo possível de cabos de internet.

A sua tarefa é descobrir qual será o custo total com cabos que a empresa terá após as modificações feitas por Bruno. A figura abaixo mostra (a) a infraestrutura de redes atual; e (b) a infraestrutura de redes após as modificação feitas.



Entrada

A primeira linha é composta por dois inteiros R ($3 \leq R \leq 60$) e C ($R \leq C \leq 200$) representado respectivamente a quantidade de roteadores e a quantidade de cabos de internet utilizados atualmente.

Seguem C linhas, cada uma contendo três inteiros V ($1 \leq V \leq R$), W ($1 \leq W \leq R$) e P ($1 \leq P \leq 10000$), sendo V e W um par de roteadores que estão conectados por um cabo de internet e P o preço do cabo de internet utilizado.

Saída

Seu programa deve imprimir um único valor inteiro que representa o custo total que a empresa gastará com cabos após as modificações.

Exemplo de Entrada	Exemplo de Saída
7 12 1 3 6 1 4 9 2 3 17 2 5 32 2 7 27 3 4 11 3 5 4 4 5 3 4 6 19 5 6 13 5 7 15 6 7 5	48





CÓDIGO FONTE	
1	#include <iostream>
2	#include <algorithm>
3	
4	struct Edge
5	{
6	int v;
7	int w;
8	int p;
9	};
10	
11	Edge g[205];
12	int p[65];
13	
14	bool compare(const Edge &a, const Edge &b)
15	{
16	return a.p < b.p;
17	}
18	
19	int parent(int i)
20	{
21	if (i == p[i])
22	return i;
23	return parent(p[i]);
24	}
25	
26	int kruskal(int c)
27	{
28	int i, mst, v, w;
29	
30	for (i = 0, mst = 0; i < c; i++)
31	{
32	v = parent(g[i].v);
33	w = parent(g[i].w);
34	
35	if (v != w)
36	{
37	p[v] = p[w];
38	mst += g[i].p;
39	}
40	}
41	
42	return mst;
43	}
44	
45	int main()
46	{
47	int r, c, i;
48	
49	std::cin >> r >> c;
50	
51	for (i = 0; i < c; i++)
52	{
53	std::cin >> g[i].v >> g[i].w >> g[i].p;
54	}
55	
56	std::sort(g, g + c, compare);
57	for (i = 1; i <= r; i++)
58	p[i] = i;
59	
60	std::cout << kruskal(c) << std::endl;
61	
62	return 0;
63	}


Beecrowd


1774


Roteadores



beecrowd



AO VIVO
O que os outros estão resolvendo.



LISTAR
Liste todas as suas submissões.


TENTADO
Problemas ainda não resolvidos.


FAQS
Precisa de ajuda?


RESPOSTAS
O que isso significa?


FÓRUM
Busque por ajuda no Fórum.


CÓDIGO FONTE
EDITAR & ENVIAR

VISUALIZE O CÓDIGO FONTE DE SUAS SUBMISSÕES, JUNTO COM ALCUNS DETALHES EXTRAS.

SUBMISSÃO # 33550170

PROBLEMA:

1774 - Roteadores

RESPOSTA:

Accepted

LINGUAGEM:

C++17 (g++ 7.3.0, -std=c++17 -O2 -lm) [+0s]

TEMPO:

0.000s

TAMANHO:

758 Bytes

MEMÓRIA:

-

SUBMISSÃO:

13/05/2023 11:01:03

CÓDIGO FONTE

```
1 #include <iostream>
2 #include <algorithm>
3
4 struct Edge
5 {
6     int v;
7     int w;
8     int p;
9 };
10
11 Edge g[205];
12 int p[65];
13
14 bool compare(const Edge &a, const Edge &b)
15 {
16     return a.p < b.p;
17 }
18
19 int parent(int i)
20 {
21     if (i == p[i])
22         return i;
23     return parent(p[i]);
24 }
25
26 int kruskal(int c)
27 {
28     int i, mst, v, w;
29
30     for (i = 0, mst = 0; i < c; i++)
31     {
32         v = parent(g[i].v);
```

Beecrowd

1931

Mania de par

beecrowd | 1931

Mania de Par

Por Vinícius Fernandes dos Santos, Centro Federal de Educação Tecnológica de Minas Gerais. 🇧🇷 Brazil

Timelimit: 1

Patrícia é uma ótima desenvolvedora de software. No entanto, como quase toda pessoa brilhante, ela tem algumas manias estranhas, e uma delas é que tudo que ela faz tem que ser em número par. Muitas vezes essa mania não atrapalha, apesar de causar estranhamento nos outros. Alguns exemplos: ela tem que fazer diariamente um número par de refeições; no café da manhã toma duas xícaras de café, duas torradas e duas fatias de queijo; sempre que vai ao cinema compra dois bilhetes de entrada (felizmente sempre tem um amigo ou amiga lhe acompanhando); e toma dois banhos por dia (ou quatro, ou seis...).

Mas algumas vezes essa mania de Patrícia atrapalha. Por exemplo, ninguém gosta de viajar de carro com ela, pois se no trajeto ela tem que pagar pedágios, o número de pedágios que ela paga tem que ser par.

Patrícia mora em um país em que todas as estradas são bidirecionais e têm exatamente um pedágio. Ela precisa ir visitar um cliente em uma outra cidade, e deseja calcular o mínimo valor total de pedágios que ela tem que pagar, para ir da sua cidade à cidade do cliente, obedecendo à sua estranha mania de que o número de pedágios pagos tem que ser par.

Entrada

A entrada consiste de diversas linhas. A primeira linha contém 2 inteiros **C** e **V**, o número total de cidades e o número de estradas ($2 \leq C \leq 10^4$ e $0 \leq V \leq 50000$). As cidades são identificadas por inteiros de 1 a **C**. Cada estrada liga duas cidades distintas, e há no máximo uma estrada entre cada par de cidades. Cada uma das **V** linhas seguintes contém três inteiros **C**₁, **C**₂ e **G**, indicando que o valor do pedágio da estrada que liga as cidades **C**₁ e **C**₂ é **G** ($1 \leq C_1, C_2 \leq C$ e $1 \leq G \leq 10^4$). Patrícia está atualmente na cidade 1 e a cidade do cliente é **C**.

Saída

Uma única linha deve ser impressa, contendo um único inteiro, o custo total de pedágios para Patrícia ir da cidade 1 à cidade **C**, pagando um número par de pedágios, ou, se isso não for possível, o valor -1.

Exemplo de Entrada	Exemplo de Saída
4 4 1 2 2 2 3 1 2 4 10 3 4 6	12
5 6 1 2 3 2 3 5 3 5 2 5 1 8 2 4 1 4 5 4	-1


Beecrowd 1931 Mania de par

```
2 #include <vector>
3 #include <queue>
4
5 using namespace std;
6
7 #define INF 9999999
8
9 class Graph
10 {
11 public:
12     int numVertices;
13     vector<vector<pair<int, int>>> adjacencyList;
14
15     Graph(int numVertices)
16     {
17         this->numVertices = numVertices;
18         adjacencyList.resize(numVertices);
19     }
20
21     void addEdge(int u, int v, int weight)
22     {
23         adjacencyList[u].push_back(make_pair(v, weight));
24     }
25
26     int dijkstra(int source, int target)
27     {
28         vector<int> distances(numVertices, INF);
29         vector<int> visited(numVertices, 0);
30
31         priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
32
33         distances[source] = 0;
34
35         pq.push(make_pair(distances[source], source));
36
37         while (!pq.empty())
38         {
39             pair<int, int> p = pq.top();
40             int u = p.second;
41             pq.pop();
42
43             if (!visited[u])
44             {
45                 visited[u] = 1;
46                 for (auto it = adjacencyList[u].begin(); it != adjacencyList[u].end(); it++)
47                 {
48                     int v = it->first;
49                     int weight = it->second;
50
51                     if (distances[v] > distances[u] + weight)
52                     {
53                         distances[v] = distances[u] + weight;
54                         pq.push(make_pair(distances[v], v));
55                     }
56                 }
57             }
58         }
59
60         return distances[target];
61     }
62 };
63
```


```
64 int main()
65 {
66     int numCities, numRoads, city1, city2, weight;
67
68     cin >> numCities >> numRoads;
69
70     Graph graph(numCities), graph2(numCities);
71
72     for (int i = 0; i < numRoads; i++)
73     {
74         cin >> city1 >> city2 >> weight;
75         graph.addEdge(--city1, --city2, weight);
76         graph.addEdge(city2, city1, weight);
77     }
78
79     for (int i = 0; i < numCities; i++)
80     {
81         for (auto it1 = graph.adjacencyList[i].begin(); it1 != graph.adjacencyList[i].end(); it1++)
82         {
83             city1 = it1->first;
84             weight = it1->second;
85
86             if (!graph.adjacencyList[city1].empty())
87             {
88                 for (auto it2 = graph.adjacencyList[city1].begin(); it2 != graph.adjacencyList[city1].end(); it2++)
89                 {
90                     if (i != it2->first)
91                         graph2.addEdge(i, it2->first, weight + it2->second);
92                 }
93             }
94         }
95     }
96
97     int minDistance = graph2.dijkstra(0, numCities - 1);
98
99     cout << (minDistance == INF ? -1 : minDistance) << endl;
100
101     return 0;
102 }
```




Beecrowd
1931
Mania de par




beecrowd




AO VIVO
O que os outros estão resolvendo.




LISTAR
Liste todas as suas submissões.




TENTADO
Problemas ainda não resolvidos.




FAQS
Precisa de ajuda?



RESPOSTAS
O que isso significa?



FÓRUM
Busque por ajuda no Fórum.



CÓDIGO FONTE

EDITAR & ENVIAR

VISUALIZE O CÓDIGO FONTE DE SUAS SUBMISSÕES, JUNTO COM ALGUNS DETALHES EXTRAS.

SUBMISSÃO # 33554010

PROBLEMA:	1931 - Mania de Par
RESPOSTA:	Accepted
LINGUAGEM:	C++17 (g++ 7.3.0, -std=c++17 -O2 -lm) [+0s]
TEMPO:	0.135s
TAMANHO:	2,31 KB
MEMÓRIA:	-
SUBMISSÃO:	13/05/2023 14:49:14

CÓDIGO FONTE

```
1 #include <iostream>
2 #include <vector>
3 #include <queue>
4
5 using namespace std;
6
7 #define INF 9999999
8
9 class Graph
10 {
11 public:
12     int numVertices;
13     vector<vector<pair<int, int>>> adjacencyList;
14
15     Graph(int numVertices)
16     {
17         this->numVertices = numVertices;
18         adjacencyList.resize(numVertices);
19     }
20
21     void addEdge(int u, int v, int weight)
22     {
23         adjacencyList[u].push_back(make_pair(v, weight));
```


Beecrowd 1550 Inversão



BEE 1550

GRAPOS | NÍVEL 7 | + 7.2 PONTOS | TEMPO LIMITE BASE: 3 SEGUNDOS | LIMITE DE MEMÓRIA: 200 MB

beecrowd | 1550

Inversão

Por Gabriel Dalalio, ITA 🇧🇷 Brazil

Timelimit: 3

Pedro é um garoto curioso que gostava de eletrônica. Certo dia, o menino estava mexendo no laboratório de sua escola e encontrou uma caixa cheia de pequenos aparelhos eletrônicos feitos por outros alunos em anos anteriores.

Dentro dessa caixa havia um aparelho que possuía apenas um visor e dois botões. Esse visor apresentava um número inteiro. Mexendo nos botões, Pedro descobriu para que servia cada um deles. O primeiro botão adicionava uma unidade ao número no visor. O segundo botão invertia os dígitos do número, por exemplo, 123 invertido resulta em 321 e 150 invertido resulta em 51 (ignora-se os zeros a esquerda).

Inicialmente, o visor apresentava o número A . Após a descoberta da função dos botões, Pedro quer saber como fazer o número do visor mudar de A para um número maior igual a B . O seu trabalho nesse problema é ajudar Pedro a descobrir qual é o número mínimo de apertos de botão para que o número no visor passe a ser igual a B .

Entrada

A entrada é iniciada por um inteiro T , $0 < T \leq 500$, que indica a quantidade de casos de teste a ser processados. Segue-se T linhas cada uma contendo dois inteiros A e B , $0 < A < B < 10000$, indicando respectivamente o número inicial no visor e o número que deve ser mostrado no visor depois de apertar os botões.

Saída

Para cada caso de teste, o programa deve imprimir um inteiro indicando o número mínimo de apertos de botão para que o número do visor passe de A para B .

Exemplo de Entrada	Exemplo de Saída
4	8
1 9	4
100 301	3
808 909	3
133 233	

Aquecimento para a OBI 2014

Beecrowd

1550


Inversão


```
13 int inc(int a)
14 {
15     return a + 1;
16 }
17
18 int inv(int a)
19 {
20     int r = 0;
21     while (a)
22     {
23         r *= 10;
24         r += a % 10;
25         a /= 10;
26     }
27     return r;
28 }
29
30 void init()
31 {
32     v.assign(MAX, vector<int>());
33     for (int i = 0; i < MAX; i++)
34     {
35         v[i].push_back(inc(i));
36         v[i].push_back(inv(i));
37     }
38 }
39
40 void dijkstra(int a)
41 {
42     d.assign(MAX, INF);
43     d[a] = 0;
44
45     priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
46     pq.push(make_pair(0, a));
47
48     while (!pq.empty())
49     {
50         pair<int, int> u = pq.top();
51         pq.pop();
52         int w = u.second;
53         if (w < MAX)
54         {
55             for (int i = 0; i < v[w].size(); i++)
56             {
57                 int vx = v[w][i];
58                 if (vx < MAX - 1 && d[vx] > u.first + 1)
59                 {
60                     d[vx] = u.first + 1;
61                     pq.push(make_pair(d[vx], vx));
62                 }
63             }
64         }
65     }
66 }
67
68 int main(void)
69 {
70     int t, a, b;
71
72     init();
73
74     cin >> t;
75
76     while (t--)
77     {
78         cin >> a >> b;
79         d.clear();
80         dijkstra(a);
81         cout << d[b] << endl;
82     }
83
84     v.clear();
85
86     return 0;
```


Beecrowd


1550


Inversão


**beecrowd**


**AO VIVO**
O que os outros estão resolvendo.


**LISTAR**
Liste todas as suas submissões.

**TENTADO**
Problemas ainda não resolvidos.

**FAQS**
Precisa de ajuda?

**RESPOSTAS**
O que isso significa?

**FÓRUM**
Busque por ajuda no Fórum.

**CÓDIGO FONTE**

EDITAR & ENVIAR

VISUALIZE O CÓDIGO FONTE DE SUAS SUBMISSÕES, JUNTO COM ALGUNS DETALHES EXTRAS.

SUBMISSÃO # 33550170

PROBLEMA:

1774 - Roteadores

RESPOSTA:

Accepted

LINGUAGEM:

C++17 (g++ 7.3.0, -std=c++17 -O2 -lm) [+0s]

TEMPO:

0.000s

TAMANHO:

758 Bytes

MEMÓRIA:

-

SUBMISSÃO:

13/05/2023 11:01:03

CÓDIGO FONTE

```
1 #include <iostream>
2 #include <algorithm>
3
4 struct Edge
5 {
6     int v;
7     int w;
8     int p;
9 };
10
11 Edge g[205];
12 int p[65];
13
14 bool compare(const Edge &a, const Edge &b)
15 {
16     return a.p < b.p;
17 }
18
19 int parent(int i)
20 {
21     if (i == p[i])
22         return i;
23     return parent(p[i]);
```

LeetCode

1584

LeetCode

Description

Editorial

Solutions (1.3K)

Submissions

1584. Min Cost to Connect All Points

Medium

3.6K

84

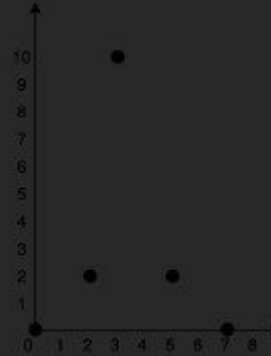
Companies

You are given an array `points` representing integer coordinates of some points on a 2D-plane, where `points[i] = [xi, yi]`.

The cost of connecting two points `[xi, yi]` and `[xj, yj]` is the **Manhattan distance** between them: `|xi - xj| + |yi - yj|`, where `|val|` denotes the absolute value of `val`.

Return the *minimum cost* to make all points connected. All points are connected if there is **exactly one** simple path between any two points.

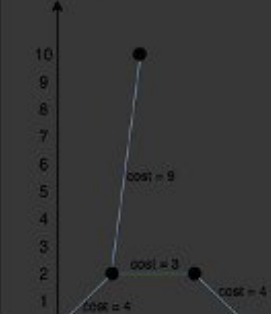
Example 1:



Input: `points = [[0,0],[2,2],[3,10],[5,2],[7,0]]`

Output: `20`

Explanation:



Pressione **F11** para sair do modo tela cheia

```
1 // Função para calcular o valor absoluto de um número
2 int mod(int x){
3     if(x < 0) return -1 * x; // Retorna o valor absoluto de x se x for negativo
4     return x; // Retorna x se x for positivo ou zero
5 }
6
7 // Função para calcular a distância de Manhattan entre dois pontos
8 int manhattandistance(int** points, int i, int j, int* pointsColSize){
9     int* p1 = *(points + i); // Obtém o ponteiro para a linha i da matriz de pontos
10    int* p2 = *(points + j); // Obtém o ponteiro para a linha j da matriz de pontos
11    int distancia = 0; // Inicializa a distância como zero
12    for(int k = 0; k < (*pointsColSize); k++){ // Percorre as colunas das linhas i e j
13        distancia = distancia + mod(*(p1 + k) - *(p2 + k)); // Calcula a distância de Manhattan entre os pontos p1 e p2
14    }
15    return distancia; // Retorna a distância de Manhattan entre os pontos p1 e p2
16 }
17
18 // Função para calcular o custo mínimo de conexão entre todos os pontos
19 int minCostConnectPoints(int** points, int pointsSize, int* pointsColSize){
20     int distancia = 0; // 0 custo final
21     int distMinima; // Distância mínima temporária
22     int dist; // Distância temporária
23     int S[pointsSize]; // 0 conjunto S (contém os vértices da MST)
24     int top = 0; // Índice do topo do conjunto S
25     int present[pointsSize]; // Array que marca a presença dos vértices contidos em S
26     int v2; // Vértice, usado com distMinima para adicionar a S
27     int edges[pointsSize]; // A distância mínima para vértices (fora de S) a partir de vértices dentro de S
28
29     // Inicialização dos arrays
30     for(int j = 0; j < pointsSize; j++){
```

Testcase

Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

points =

`[[0,0],[2,2],[3,10],[5,2],[7,0]]`

Output

`20`

Expected

Console

Run

Submit

Obrigada!

