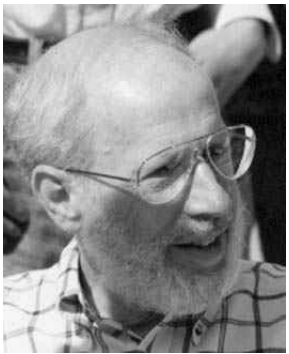
Grafos

Algoritmo de Kruskall

Prof. Edson Alves

Faculdade UnB Gama

Proponente



Joseph Bernard Kruskall, Jr.

(1962)

Características do algoritmo de Kruskall

Características do algoritmo de Kruskall

- O algoritmo de Kruskall encontra uma MST usando uma abordagem gulosa

Características do algoritmo de Kruskall

- O algoritmo de Kruskall encontra uma MST usando uma abordagem gulosa
- As arestas são ordenadas, ascendentemente, por peso

Características do algoritmo de Kruskall

- O algoritmo de Kruskall encontra uma MST usando uma abordagem gulosa
- As arestas são ordenadas, ascendentemente, por peso
- Inicialmente os vértices formam uma floresta de vértices isolados

Características do algoritmo de Kruskall

- O algoritmo de Kruskall encontra uma MST usando uma abordagem gulosa
- As arestas são ordenadas, ascendentemente, por peso
- Inicialmente os vértices formam uma floresta de vértices isolados
- Na ordem estipulada, cada aresta que una dois componentes conectados fará parte da MST e unirá estes componentes distintos

Características do algoritmo de Kruskall

- ⋆ **O algoritmo de Kruskall encontra uma MST usando uma abordagem gulosa**
- ⋆ **As arestas são ordenadas, ascendentemente, por peso**
- ⋆ **Inicialmente os vértices formam uma floresta de vértices isolados**
- ⋆ **Na ordem estipulada, cada aresta que una dois componentes conectados fará parte da MST e unirá estes componentes distintos**
- ⋆ **Complexidade:** $O(E \log V)$

Pseudocódigo

Pseudocódigo

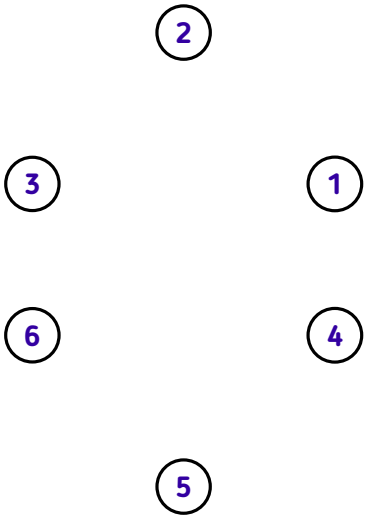
Entrada: um grafo ponderado $G(V, E)$
Saída: uma MST de G

1. Faça $M = \emptyset$ e seja $F(V, \emptyset)$ uma floresta de vértices isolados

2. Ordene E ascendentemente, por peso

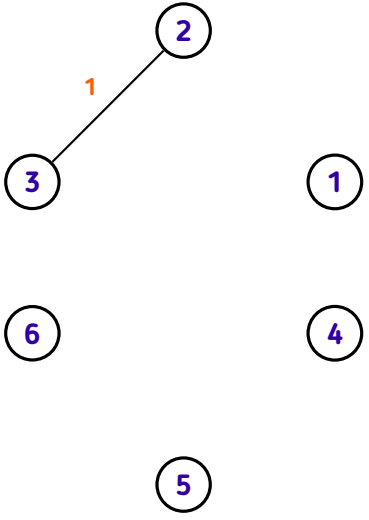
3. Para cada $(u, v, w) \in E$, se u e v estão em componentes distintos de F:
   (a) Una estes componentes em F
   (b) Inclua (u, v, w) no conjunto M

4. Retorne M



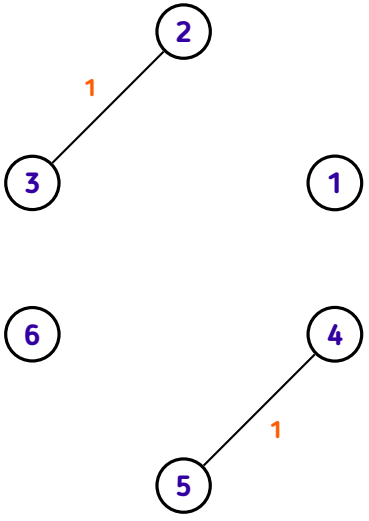
E

1 2 3
1 4 5
2 4 1
2 1 5
3 3 4
4 2 1
5 3 1
7 3 6
8 6 5



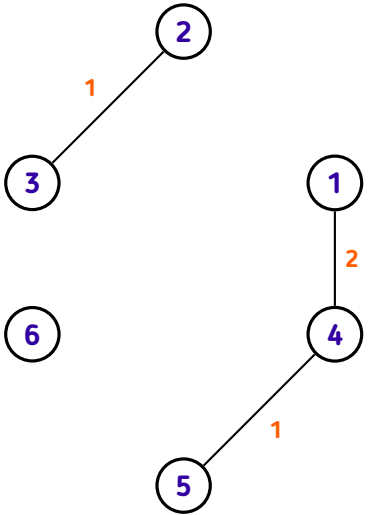
E

1 2 3 ✔
1 4 5
2 4 1
2 1 5
3 3 4
4 2 1
5 3 1
7 3 6
8 6 5



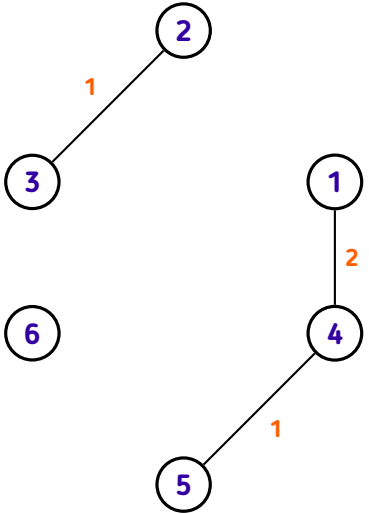
E

1 2 3 ✔
1 4 5 ✔
2 4 1
2 1 5
3 3 4
4 2 1
5 3 1
7 3 6
8 6 5



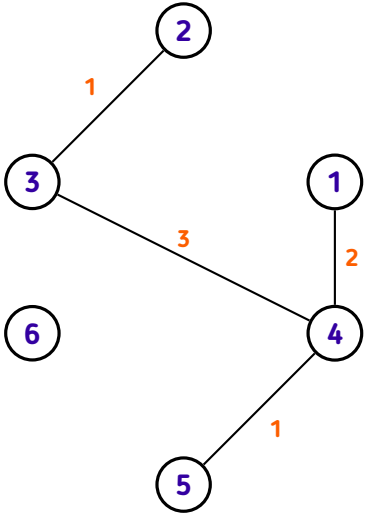
E

1 2 3 ✔
1 4 5 ✔
2 4 1 ✔
2 1 5
3 3 4
4 2 1
5 3 1
7 3 6
8 6 5



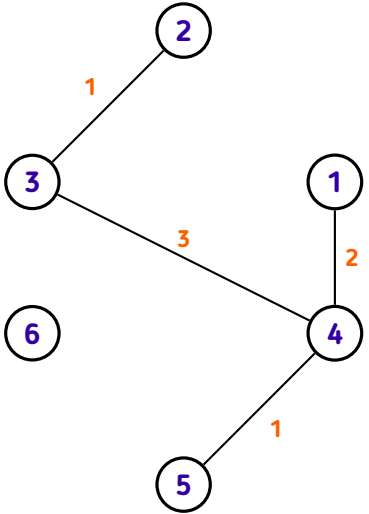
E

1 2 3 ✔
1 4 5 ✔
2 4 1 ✔
2 1 5 ✖
3 3 4
4 2 1
5 3 1
7 3 6
8 6 5



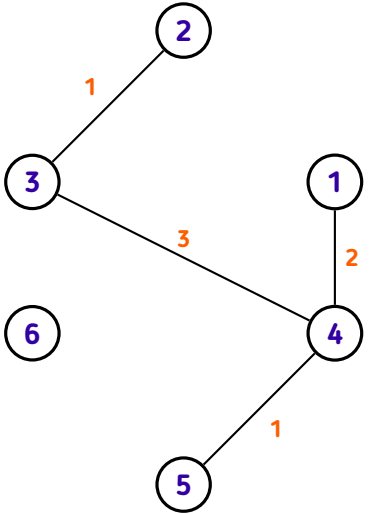
E

1 2 3 ✔
1 4 5 ✔
2 4 1 ✔
2 1 5 ✘
3 3 4 ✔
4 2 1
5 3 1
7 3 6
8 6 5



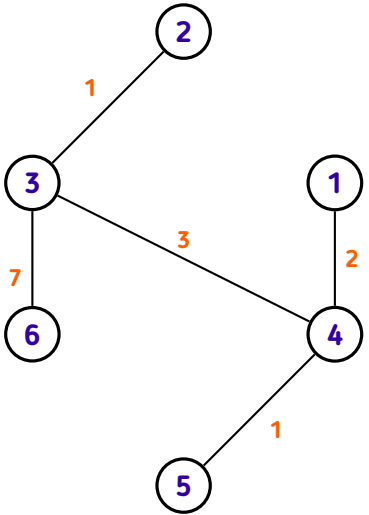
E

1 2 3 ✔
1 4 5 ✔
2 4 1 ✔
2 1 5 ✘
3 3 4 ✔
4 2 1 ✘
5 3 1
7 3 6
8 6 5



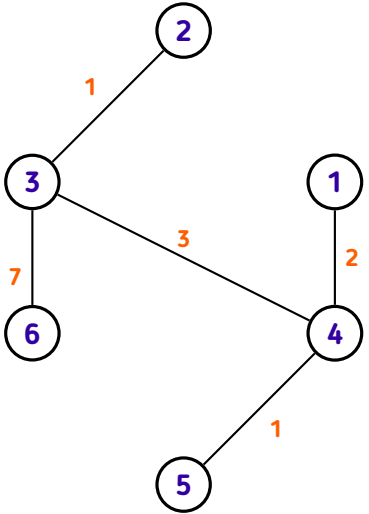
E

1 2 3 ✔
1 4 5 ✔
2 4 1 ✔
2 1 5 ✘
3 3 4 ✔
4 2 1 ✘
5 3 1 ✘
7 3 6
8 6 5



E

1 2 3 ✔
1 4 5 ✔
2 4 1 ✔
2 1 5 ✘
3 3 4 ✔
4 2 1 ✘
5 3 1 ✘
7 3 6 ✔
8 6 5



E

1 2 3 ✔
1 4 5 ✔
2 4 1 ✔
2 1 5 ✘
3 3 4 ✔
4 2 1 ✘
5 3 1 ✘
7 3 6 ✔
8 6 5 ✘

```cpp
int kruskal(int N, vector<edge>& es)
{
    sort(es.begin(), es.end());

    int cost = 0;
    UnionFind ufds(N);

    for (auto [w, u, v] : es)
    {
        if (not ufds.same_set(u, v))
        {
            cost += w;
            ufds.union_set(u, v);
        }
    }

    return cost;
}
```

Problemas sugeridos

1. **Codeforces Round #179 (Div. 1) – Problem B: Greg and Graph**
2. **LightOJ – Travel Company**
3. **OJ 104 – Arbitrage**
4. **OJ 10171 – Meeting Prof. Miguel...**

Referências

1. **CP-Algorithm.** *Minimum spanning tree – Kruskall's algorithm*, **acesso em 24/08/2021.**

2. **HALIM**, **Felix**; **HALIM**, **Steve**. *Competitive Programming 3*, **2010.**

3. **LAAKSONEN**, **Antti**. *Competitive Programmer's Handbook*, **2018.**

4. **Wikipédia**. *Joseph Kruskal*, **acesso em 24/08/2021.**

5. **Wikipédia**. *Kruskal's algorithm*, **acesso em 24/08/2021.**