ESCOLA DE PRIMAVERA DA MARATONA SBC DE PROGRAMAÇÃO

PROMOÇÃO:

CEFET-MG

UFMG

UFV

UNIFEI

USP Universidade de São Paulo

UFU

APOIO:

beecrowd

SBC Sociedade Brasileira de Computação

Grupo de Computação Competitiva

ÁRVORE GERADORA MÍNIMA

Por: Marcos Felipe Belisário Costa - UFU

CONTEÚDOS

01 - Problema motivador
02 - Definição do algoritmo
03 - Funcionamento do algoritmo
04 - Algoritmo
05 - Resolução do problema
06 - Considerações
07 - Outras aplicações
08 - Problemas

01 - PROBLEMA MOTIVADOR

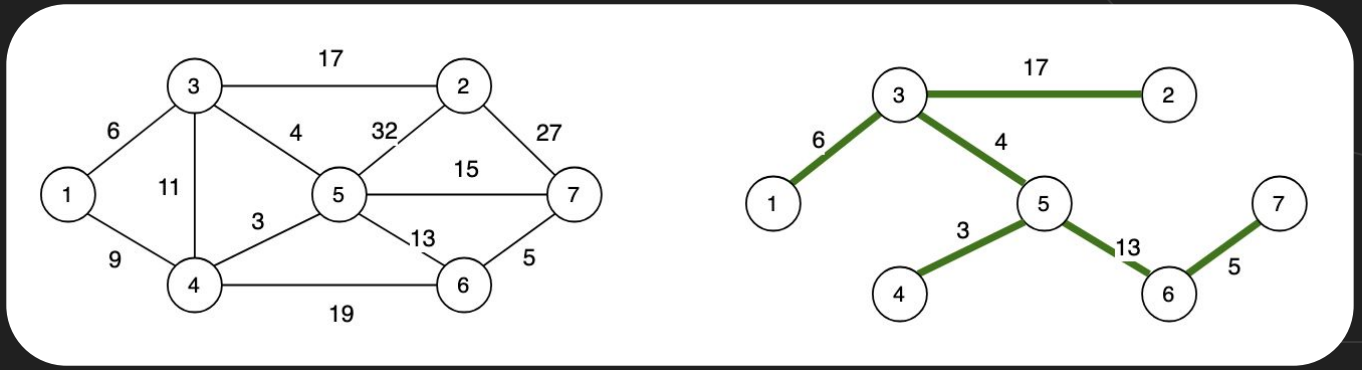
*Em uma empresa, roteadores transmitem dados entre si através de cabos de internet. Sabendo que existem **N** roteadores, é preciso haver uma rede de conexões que permita que os dados sejam trafegados de um roteador para qualquer outro, possivelmente passando por roteadores intermediários.*
Já existe uma rede de conexões atualmente, porém a empresa deseja cortar gastos removendo algumas das conexões existentes sem que a rede perca a conectividade.

Sabendo o custo de cada conexão, qual é o gasto mínimo que a empresa pode ter para manter a rede com as características desejadas após o corte de gastos?

01 - PROBLEMA MOTIVADOR

- O que o problema anterior realmente quer é o que chamamos de **Árvore Geradora Mínima**, ou *Minimum Spanning Tree* (MST)
- Uma **Árvore Geradora** é uma árvore que contém todos os **N** vértices do grafo original, porém com apenas as arestas necessárias para a formação de uma árvore (**N-1**).
- Uma **Árvore Geradora Mínima** é uma árvore geradora cuja soma dos custos das arestas é o menor possível dado um grafo original.

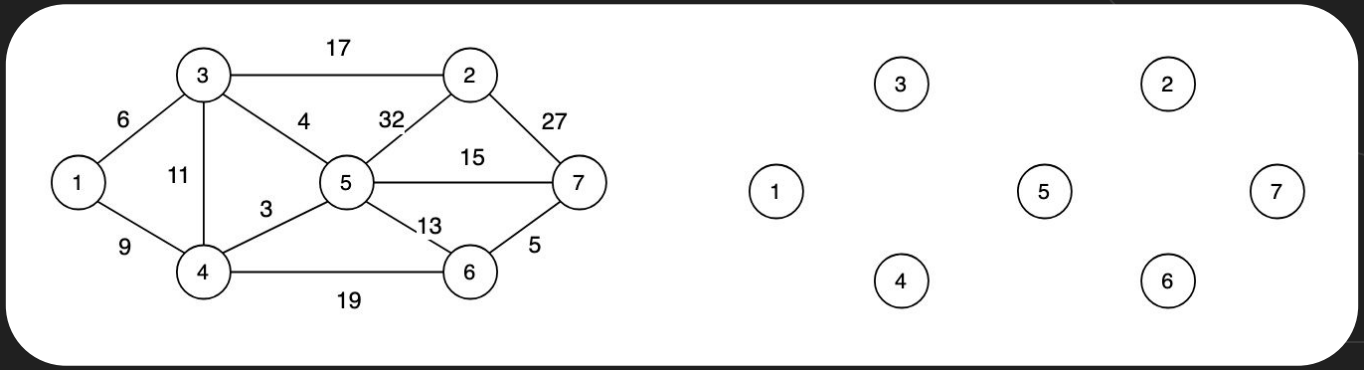
01 - PROBLEMA MOTIVADOR



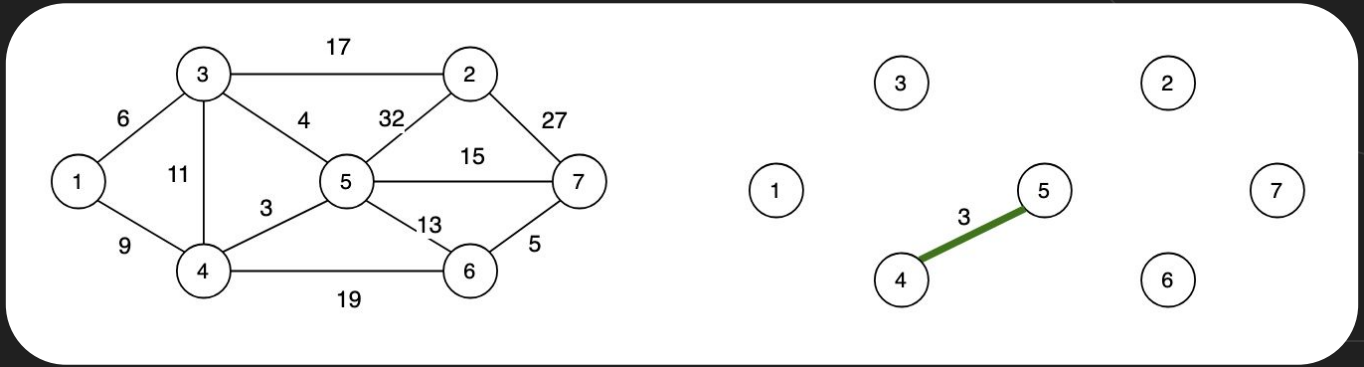
02 - DEFINIÇÃO DO ALGORITMO

- Existem alguns algoritmos que nos permitem encontrem uma árvore geradora mínima. Hoje falaremos sobre o **algoritmo de Kruskal**.
- O **Algoritmo de Kruskal** é um algoritmo guloso
- Permite encontrar o custo e uma árvore geradora mínima em tempo polinomial
- O algoritmo segue os seguintes passos:
    1. Ordenação das arestas em ordem crescente
    2. Trata todos os vértices como componentes unitárias (ignora todas as arestas)
    3. Tenta adicionar as arestas em ordem crescente desde que não formem um ciclo

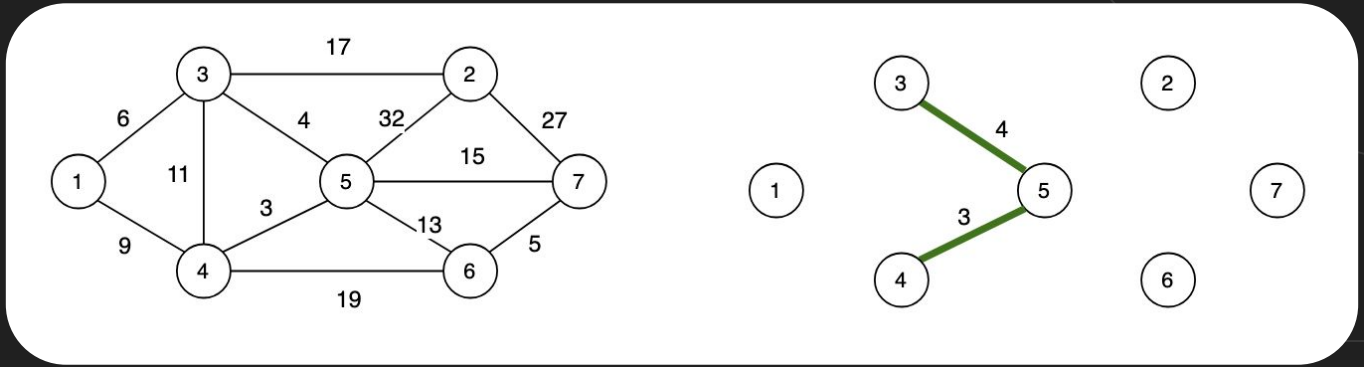
03 - FUNCIONAMENTO DO ALGORITMO



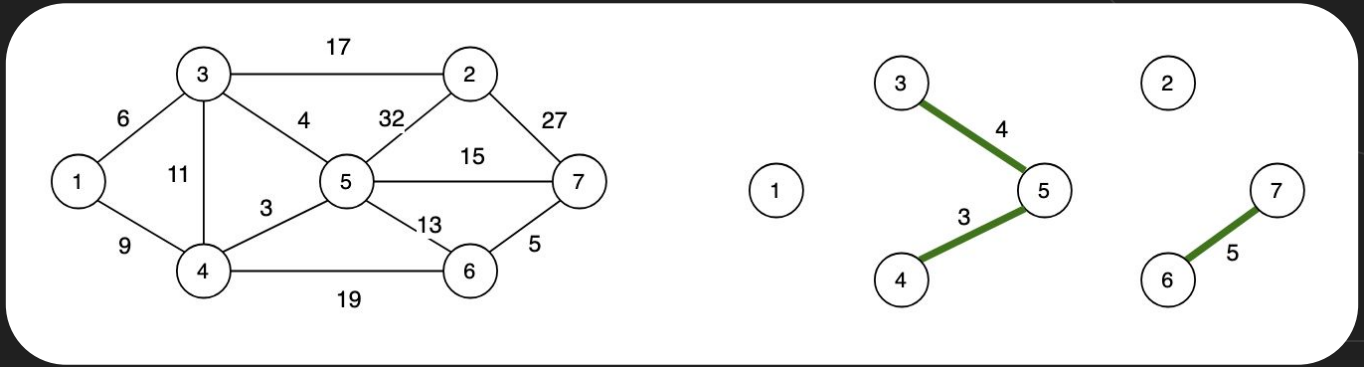
03 - FUNCIONAMENTO DO ALGORITMO



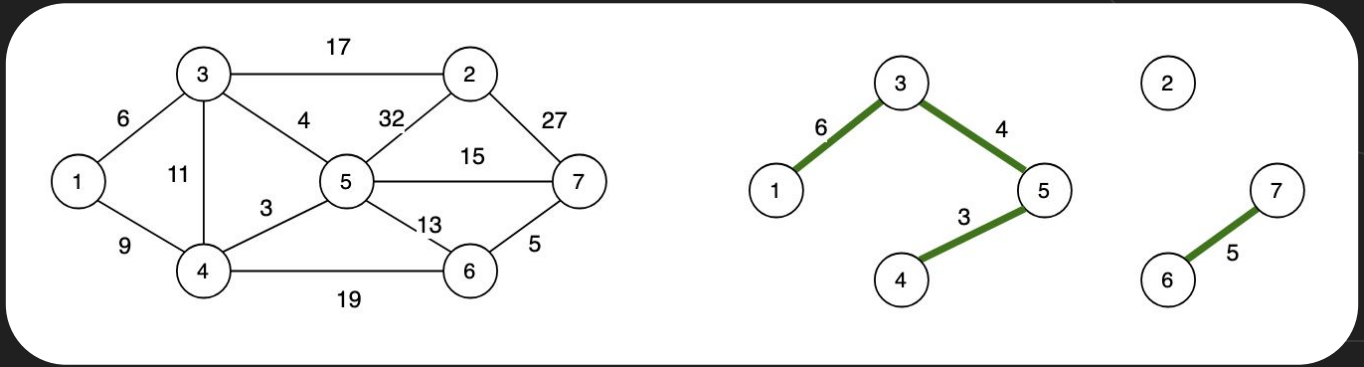
03 - FUNCIONAMENTO DO ALGORITMO



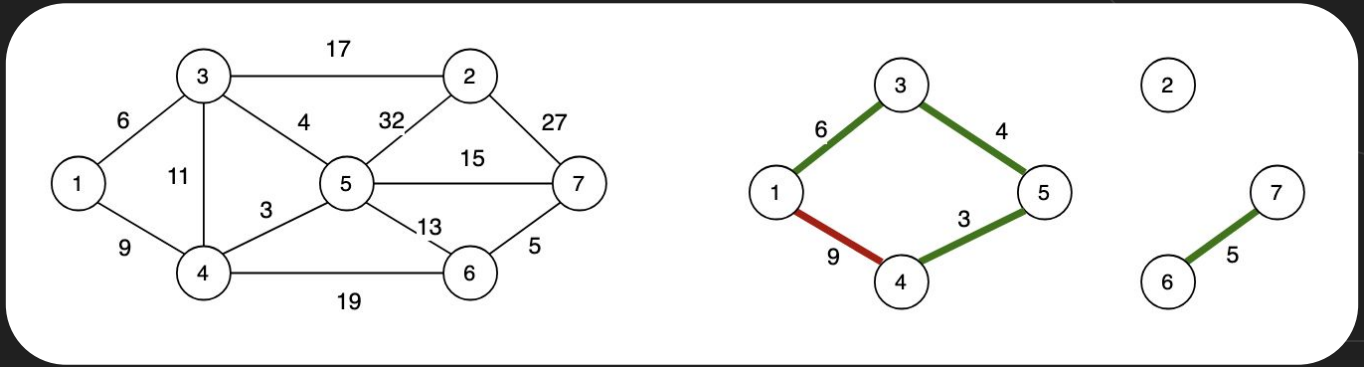
03 - FUNCIONAMENTO DO ALGORITMO



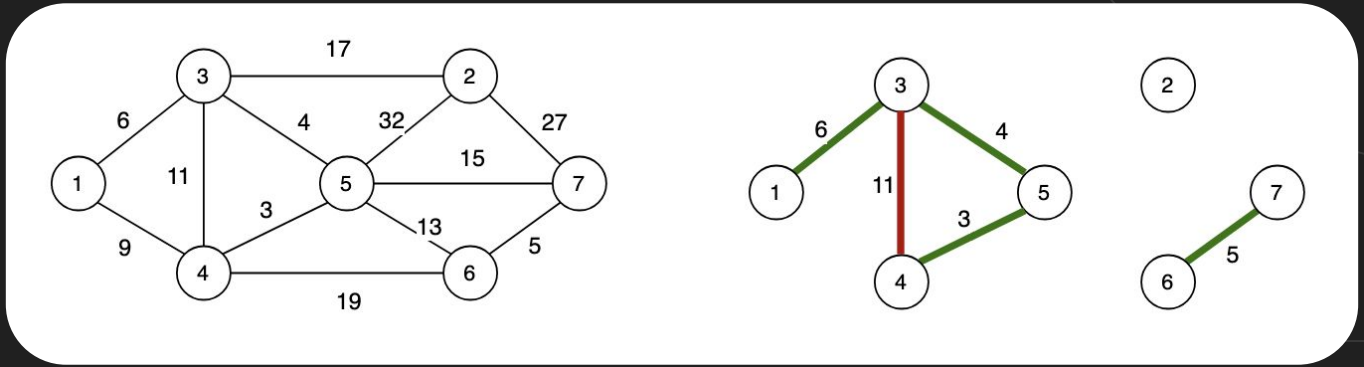
03 - FUNCIONAMENTO DO ALGORITMO



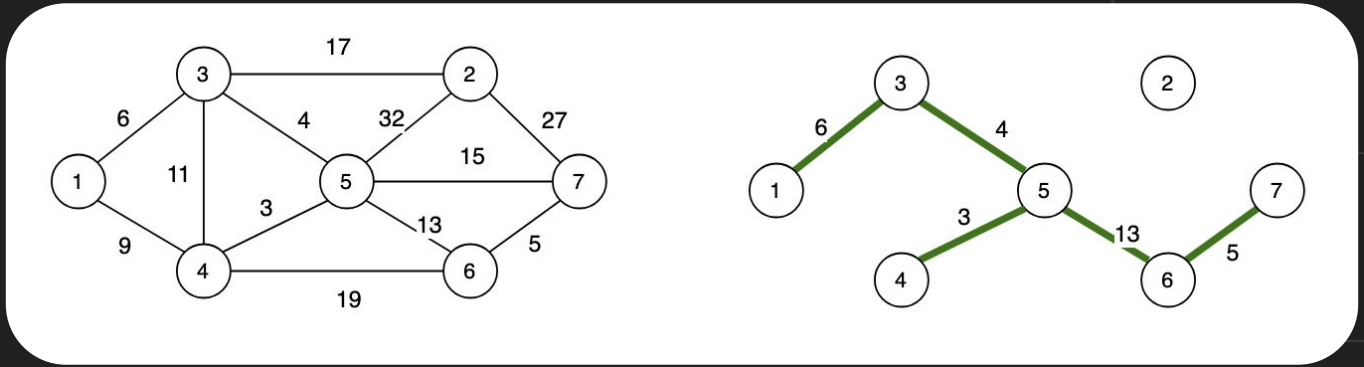
03 - FUNCIONAMENTO DO ALGORITMO



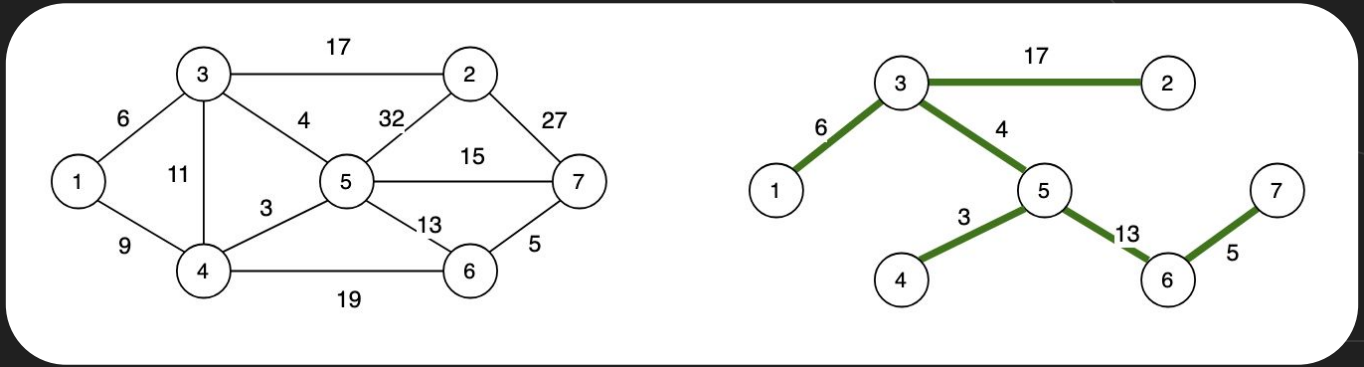
03 - FUNCIONAMENTO DO ALGORITMO



03 - FUNCIONAMENTO DO ALGORITMO



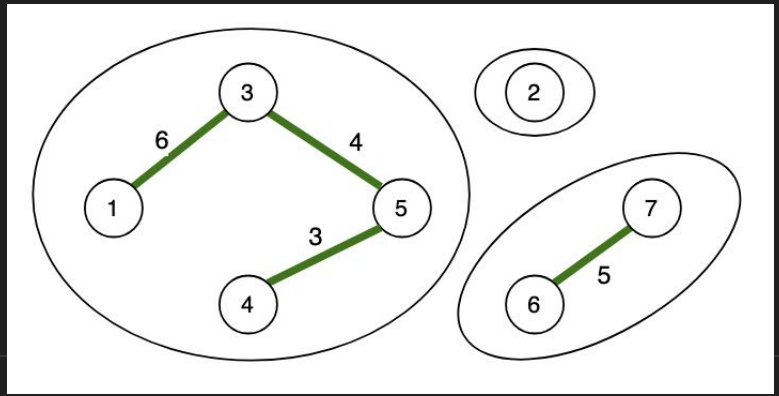
03 - FUNCIONAMENTO DO ALGORITMO



03 - FUNCIONAMENTO DO ALGORITMO

- Podemos facilmente ordenar as arestas do grafo baseado em ordem crescente de custo (utilizando a função **sort**)
- No entanto, o que podemos usar para saber se a adição de certa aresta causa um ciclo no grafo?
- **Resposta**: A estrutura de dados **Union-Find (DSU)** (ref)
  - O Union-Find tem duas operações básicas: Find e Union
  - A operação de Find retorna qual é a componente de um determinado vértice
  - A operação de Union faz com que dois vértices passem a participar da mesma componente.

03 - FUNCIONAMENTO DO ALGORITMO



04 - ALGORITMO

Union-Find

```
void InitializeUnionFind(int n) {
    for (int i=1; i<=n; i++) {
        comp[i] = i;
    }
}

int Find(int u) {
    if(comp[u] == u) return u;
    return comp[u] = Find(comp[u]);
}

void Union(int u, int v) {
    u = Find(u);
    v = Find(v);
    comp[v] =  u;
}
```

Árvore Geradora Mínima

```
int SolveMinimumSpanningTree(int n, int m) {
    InitializeUnionFind(n);
    sort(edges.begin(), edges.end());

    int totalCost = 0;
    for (int i=0; i<m; i++) {
        int u, v, c;
        c = edges[i].first;
        u = edges[i].second.first, v = edges[i].second.second;
        if(Find(u) != Find(v)) {
            totalCost += c;
            Union(u, v);
        }
    }
    return totalCost;
}
```

05 - RESOLUÇÃO DO PROBLEMA MOTIVADOR

- Link para o exercício 1774 - Roteadores
- Link para o código fonte da solução

06 - CONSIDERAÇÕES

- Pode existir múltiplas árvores geradoras mínimas, sendo que todas elas têm o mesmo custo total porém se utilizam de diferentes arestas
- Se o grafo original não for conexo, é impossível obter uma árvore geradora para o grafo inteiro.
    - No entanto, é possível obter a árvore geradora mínima de cada componente

07 - OUTRAS APLICAÇÕES

- Design de redes e similares;
- Subproblema para problemas que envolvem árvores, programação dinâmica e outros;
- Análise de agrupamentos;
- Segmentação na área de processamento de imagens;
- Dentre outros (ref).

08 - PROBLEMAS

[2404] Reduzindo Detalhes em um Mapa - https://judge.beecrowd.com/pt/problems/view/2404

[1764] Itinerário do Papai Noel - https://judge.beecrowd.com/pt/problems/view/1764

[2550] Novo Campus - https://judge.beecrowd.com/pt/problems/view/2550

[1552] Resgate em Queda Livre - https://judge.beecrowd.com/pt/problems/view/1552

[2459] Copa do Mundo - https://judge.beecrowd.com/pt/problems/view/2459

OBRIGADO PELA ATENÇÃO

Grupo de Computação Competitiva

GCC.CEFETMG

gcc-cefetmg.github.io/Site-GCC

@gcc.cefetmg