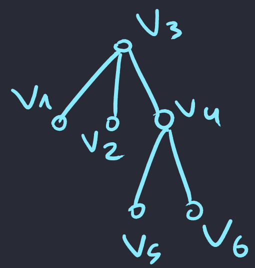
Matemática Discreta 25/10/24

Árvores

- Def: Dado um grafo T, se para cada par de vértices existir um único caminho, então T é uma árvore

- Raíz: Vértice designado como tal, é o "primeiro" vértice do grafo

Ex:



T com raíz v_3

- Nível: Dado um grafo T de raíz v_i, dado outro vértice qualquer v_j, o nível de v_j é o tamanho do caminho de v_i a v_j (não considera os possíveis pesos das arestas)

- Altura: Maior nível

- Seja T uma árvore com raíz v_0, e sejam x, y, z vértices de T e $(v_0, \ldots, v_n)$ um caminho em T:

1) v_{n-1} é pai de v_n

2) $v_0, \ldots, v_{n-1}$ são antecedentes de v_n

3) v_i é filho de v_{i-1}

4) Se x é antecessor de y, y é descendente

5) Se x e y são filhos de z, então x e y são irmãos

6) Se x não tem filhos, x é uma folha

- SUBÁRVORE

▷ A SUBÁRVORE COM RAÍZ X DE T É O SUBGRAFO DE T COM CONJUNTO DE VÉRTICES V' E ARESTAS E', ONDE V' TEM X E TODOS SEUS DESCENDENTES E E' TODAS AS ARESTAS ENVOLVIDAS NUM CAMINHO DE X A UM DESCENDENTE:

EX:

TEOREMA

SEJA T UMA ÁRVORE DE n VÉRTICES, ENTÃO:

a) T É ÁRVORE

$\Updownarrow$

b) T É CONEXO ACÍCLICO

$\Updownarrow$

c) T É CONEXO COM $n-1$ ARESTAS

$\Updownarrow$

d) T É ACÍCLICO E TEM $n-1$ ARESTAS

DEM

a) $\Rightarrow$ b)

SE T É ÁRVORE, QUALQUER PAR DE VÉRTICES ESTÁ CONECTADO POR UM CAMINHO, LOGO T É CONEXO.

SE T TEM UM CICLO

$$\{v_1, \dots, v_n, v_1\}$$

HÁ DOIS CAMINHOS DE v_1 A v_n:

$\{v_1 \dots v_n\}$ E $\{v_1, v_n\}$ $\Rightarrow$ T É ACÍCLICO

b) $\Rightarrow$ c)

Indução nos vértices (n)

$n=1$

$1-1=0$ arestas ✓

Supomos que c) vale para qualquer grafo conexo acíclico com k vértices ($k \leq n$). Seja T um grafo conexo acíclico com $n+1$ vértices, pegamos um vértice v e o maior caminho iniciado em v (p)

$\rightarrow \underline{\delta(w)=1}$

Removendo w, temos um grafo que satisfaz $n-1$ arestas, adicionando a aresta de w, temos um grafo com $n+1$ vértices e n arestas

c) $\Rightarrow$ d)

Aplicando a fórmula de Euler, temos:

$$V+F=E+2 \Leftrightarrow n+F=n-1+2 \Leftrightarrow F=1$$

Como F tem 1 face, essa face é a externa, logo não existem ciclos delimitando faces.

d) $\Rightarrow$ a)

Seja T com n vértices, $n-1$ arestas e acíclico, vamos mostrar que T é conexo para ser árvore.

Sejam T_i; $i \in \{1, \ldots, k\}$ as componentes conexas de T, cada T_i é um grafo conexo e acíclico, se n é a quantidade de vértices em T_i, por b)$\Rightarrow$c), T_i tem $n-1$ arestas,

SOMANDO, TEMOS:

$$\begin{cases} \sum_{i=1}^{k} n_i = n \\ \sum_{i=1}^{k} (n_i - 1) = n - 1 \end{cases} \Rightarrow \underbrace{\boxed{\sum_{i=1}^{k} n_i}}^{n} - \underbrace{\boxed{\sum_{i=1}^{k} 1}}^{k} = n - 1 \Rightarrow \boxed{k = 1}$$

TODAS AS COMPONENTES CONEXAS SÃO IGUAIS

◦ ÁRVORE GERADORA

▷ DADO UM GRAFO G, UMA ÁRVORE T É CHAMADA DE ÁRVORE GERADORA SE T É SUBGRAFO DE G E CONTÉM TODOS OS VÉRTICES DE G

TEOREMA

GRAFO G É CONEXO $\Leftrightarrow$ G TEM ÁRVORE GERADORA

$\Leftarrow$) SE G TEM ÁRVORE GERADORA, EXISTE AO MENOS 1 CAMINHO QUE CONECTA TODOS OS PONTOS, LOGO G É CONEXO

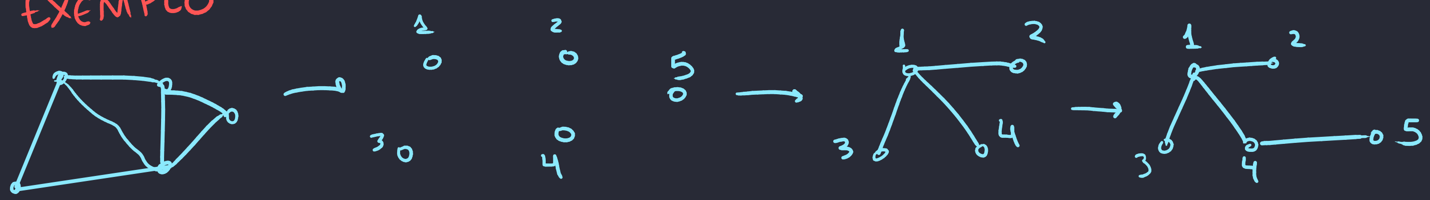
$\Rightarrow$) COMO $G(V,E)$ É CONEXO, ENTÃO $|E| > n-1$ ARESTAS, LOGO RETIRAMOS UMA ARESTA QUE RESULTA EM CICLO E REPETIMOS O PROCESSO NO SUBGRAFO RESULTANTE, OBTENDO ENTÃO UMA ÁRVORE.

ALGORITMO

(BUSCA EM LARGURA)(BREADTH-FIRST SEARCH)

VEM DA IDEIA DE UM GRAFO VAZIO E O CONJUNTO DE ARESTAS DO GRAFO:

EXEMPLO



<u>INPUT</u>: GRAFO CONEXO $G(V,E)$ $(V=\{v_1,\dots,v_n\})$

<u>OUTPUT</u>: $E' \subseteq E$ CONJUNTO DE ARESTAS DA ÁRVORE GERADORA

```
bfs(V,E){
    S=(v1)
    V'=(v1)
    E'= ∅
    while (V'≠ V){
        for x∈S in order{
            for y∈ V\V'{
                if(x,y)∈E{
                    add(x,y) to E'
                    add y to V'
                }
            }
        }
        S= children of S in order
    }
}
```

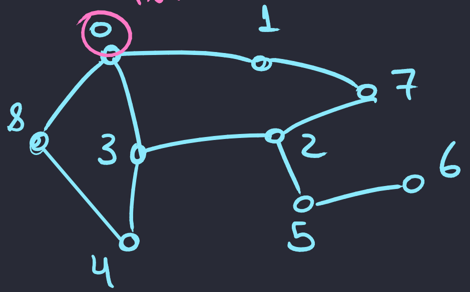
ALGORITMO

(BUSCA EM PROFUNDIDADE) (DEPTH-FIRST SEARCH)

PERCORRE A ÁRVORE ATÉ O MAIOR NÍVEL POSSÍVEL, TENDO UMA PILHA AUXILIAR QUE INDICA OS NÓS A SEREM ACESSADOS

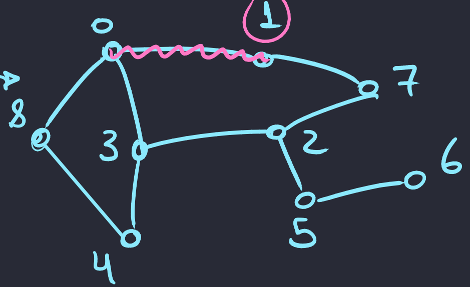
EXEMPLO

INICIAL

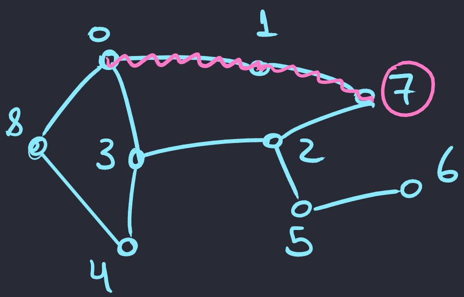


NÓS QUE PODEM SER ACESSADOS

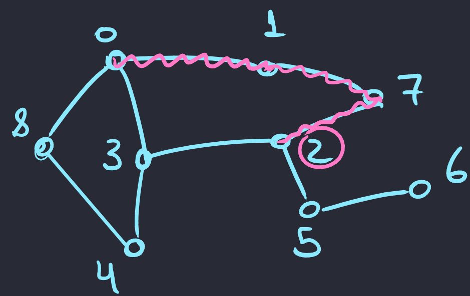
PILHA
1
3
8



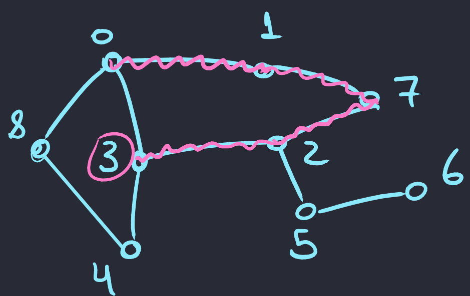
PILHA
7
3
8



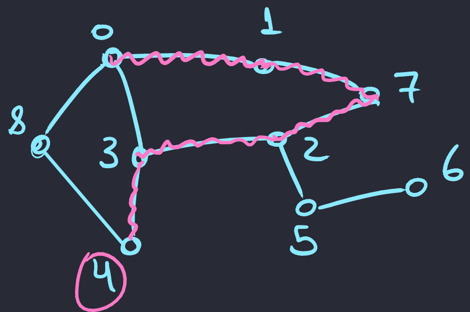
PILHA
2
3
8



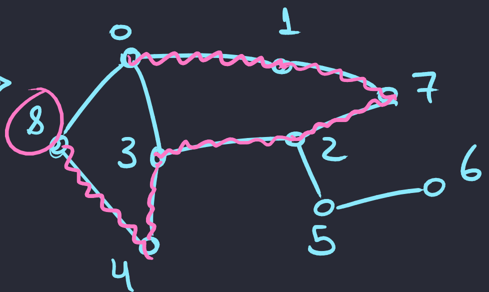
PILHA
3
5
3
8



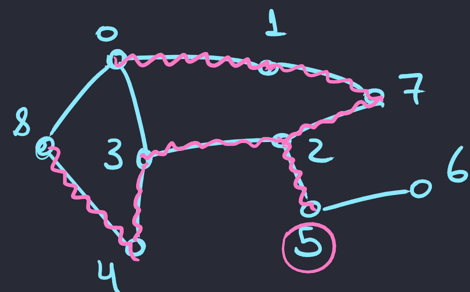
PILHA
4
5
3
8



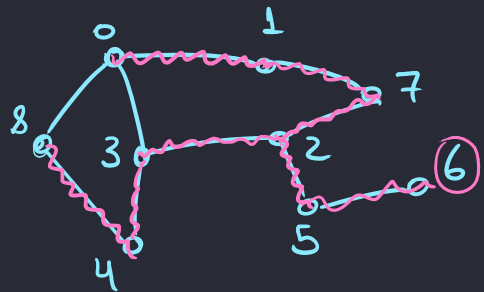
PILHA
8
5
3
8



PILHA
5
3
8

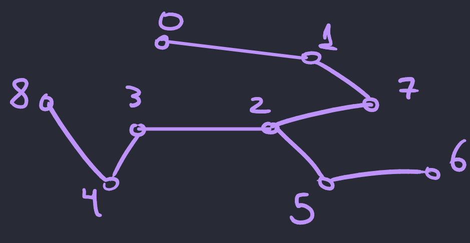


PILHA
6
3
8



PILHA
3
8

RESULTADO



INPUT: GRAFO CONEXO G COM VÉRTICES ORDENADOS

OUTPUT: ÁRVORE GERADORA

```
dfs(V, E){
    // V' OS VÉRTICES DA ÁRVORE GERADORA
    // E' AS ARESTAS //
    // v1 (V[0]) A RAÍZ DA ÁRVORE
    V' = [V[0]]
    E' = Ø
    w = V[0]
    while(true){
        while(∃{v, w} ∈ E → T+{v,w} has cicle){
            choose {w, vk} with minimum(k) and ...
                ... T + {w, vk} has not cicle
            add {w, vk} to E'
            add vk to V'
            w = vk
        }
        if (w == V[0]){
            return T
        }
        w = parent of w in T
    }
}
```

◦ Árvore Geradora Mínima

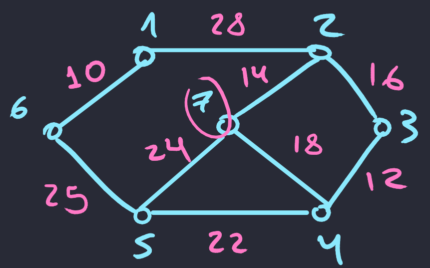
▷ Dado G um grafo com pesos, a árvore geradora mínima de G é a árvore geradora de G com a menor soma dos pesos

{Algoritmo

Algoritmo de Prim para árvores geradoras mínimas

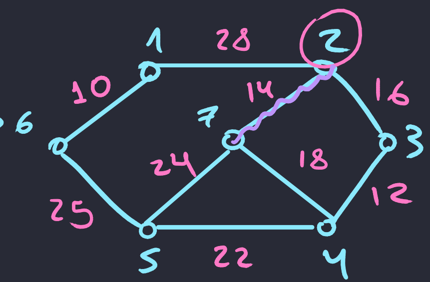
Vem da ideia de escolher um vértice, armazenar os pesos das arestas e escolher o menor, e repete o processo para cada aresta visitada

Exemplo

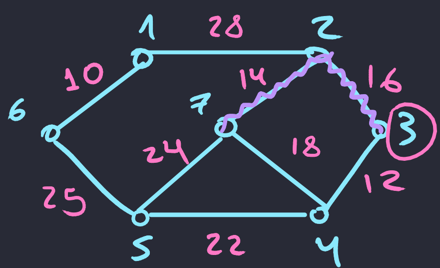


Pesos
14 menor
24
18

→

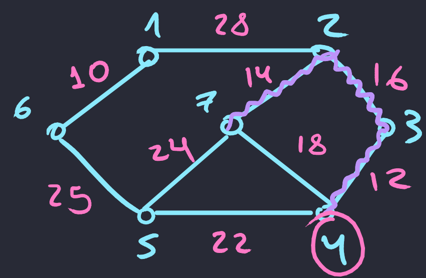


Pesos
24
18
16 menor
28

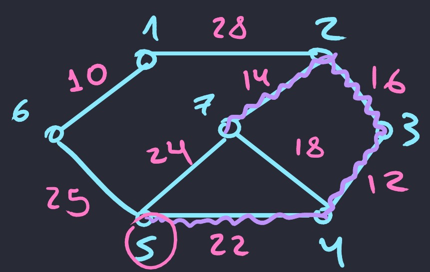


Pesos
12 menor
24
18
28

→

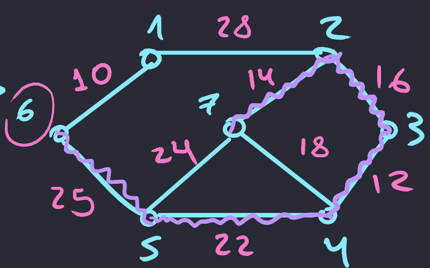


Pesos
22 menor
~~24~~
18 gera ciclo
28

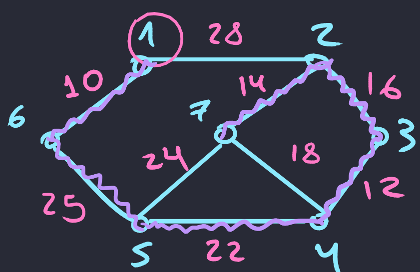


Pesos
25 menor
24 ciclo
28

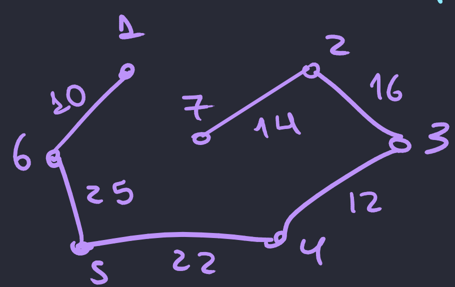
→



Pesos
10
28



output →



INPUT: Um grafo conexo com pesos e vértices $1, \ldots, n$ e vértice inicial s. Se $\{i,j\}$ é uma aresta, $w(i,j)$ é o peso dessa aresta, se $\{i,j\}$ não é uma aresta, $w(i,j) = \infty$.

OUTPUT: Conjunto de vértices da árvore geradora mínima (MST)

```
prim(w, n, s){
  // v(i)=1 se o vértice i foi adicionado na mst
  // v(i)=0 se não
  for i=1 to n{
      v(i)=0
  }
  v(s)=1 // adicionando s na mst
  E=∅ // Conjunto de arestas vazio
  for i=1 to n-1{
      min=∞
      for j=1 to n{
         if(v(j)==1){
            for k=1 to n{
               if(v(k)==0 and w(j,k)<min){
                  add_vertex = k
                  e={j,k}
                  min= w(j,k)
               }
```

```
            }
          }
        }
        v(add_vertex) = 1
        E = E ∪ {e}
    }
    return E
}
```

TEOREMA

O ALGORITMO DE PRIM ACHA UMA ÁRVORE GERADORA MÍNIMA

DEM

Vamos definir como T_i como o grafo construído pelo algoritmo depois da i-ésima iteração com $i \in \{1, \dots, n-1\}$

Seja T_0 o grafo de inicialização, por construção o grafo T_i é uma árvore com $i-1$ arestas, vamos provar por indução em i que cada T_i está contido em um grafo minimal.

Se $i=0$, T_0 é um único vértice, logo está contido numa árvore minimal

Supondo que T_i vale a hipótese indutiva, e V_i o conjunto de vértices de T_i

Seja $\{j, k\}$ a aresta selecionada ao gerar T_{i+1}

Se $\{j,k\}$ está na árvore minimal T' então $T_{i+1} \subseteq T'$

Se $\{j,k\}$ não está contido na árvore minimal T', logo $T' \cup \{j,k\}$ contém um ciclo C. C não está contido em T_{i+1}. Necessariamente C tem outra aresta $\{x,j\} \neq \{j,k\}$ com $x \in V_i$, $j \notin V_i$. Dado que $\{x,j\}$ não foi adicionado na iteração $i+1$, temos $w(x,j) \geq w(j,k)$

Definimos $T'' = \{T' \setminus \{x,j\} \cup \{j,k\}\}$

T'' é uma árvore geradora de peso menor ou igual a T', então T'' tem peso mínimo, logo, $T_{i+1} \subseteq T''$.