UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

Algoritmos em Grafos

Aula 15: Árvore Geradora Mínima(Kruskal)

Professor Pablo Soares

2022.1

Sumário

1. Árvore Geradora Mínima(**última aula**);
   a. Motivação;
   b. Algoritmo Genérico;
   c. Algoritmo de Prim.
2. Arestas seguras(Outra Forma de Escolha)
   a. Conjuntos Disjuntos;
      i. Estrutura para conjuntos disjuntos;
      ii. Funções.
   b. Algoritmo de Kruskal
      i. Complexidade do tempo de execução;
      ii. Exemplo.

Árvore Geradora Mínima (Guloso)

- Algoritmo Guloso para encontrar
  - |V| - 1 arestas de ***T***

AGM-Genérico(G, w)

1. ***T*** ← ∅
2. ***enquanto*** ***T*** *não formar uma árvore geradora mínima*
3. *Encontre uma aresta (u, v) que é segura para* ***T***
4. ***T*** ← ***T*** ∪ *(u, v)*
5. ***fimenquanto***
6. ***Retorne*** ***T***

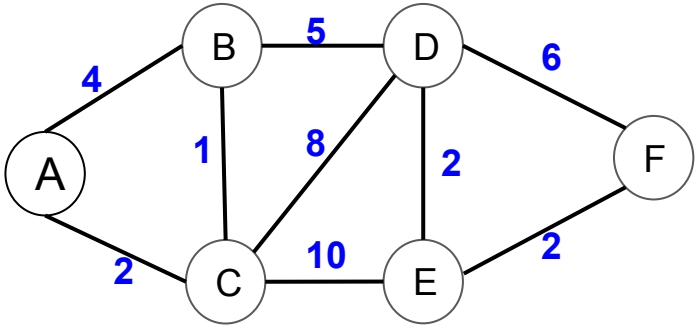
Fim.

Árvore Geradora Mínima (Arestas Seguras)

1. **Conjuntos Disjuntos**
   a. Não possuem elementos em comum
      i. Interseção é o conjunto vazio
2. Estrutura de dados para **Conjuntos Disjuntos**
   a. Árvore Direcionada
      i. cada nó representa um elemento do conjunto;
      ii. cada elemento possui um “pai”.
   b. Funções
      i. **Makeset**( ) → criar conjunto;
      ii. **Find**(x) →encontrar o pai;
      iii. **Union**(u, v) por *rank* →unir conjuntos.

Árvore Geradora Mínima (Funções)

1. Makeset(u) → criar conjunto



$G = (V, E)$

Makeset(u)

1. $\pi[u] \leftarrow u$
2. $rank[u] \leftarrow 0$

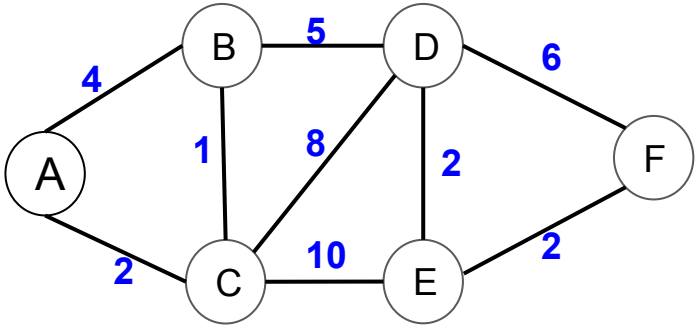
Fim.

Vértice	π	*rank*
A		
B		
C		
D		
E		
F		

A B C D E F

Árvore Geradora Mínima (Funções)

1. Makeset(u) → criar conjunto



$G = (V, E)$

Makeset(u)

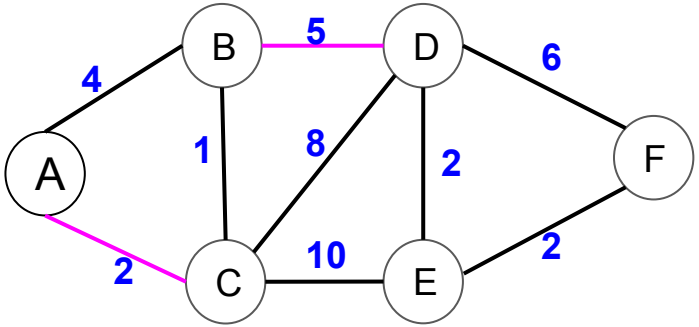
1. $\pi[u] \leftarrow u$
2. $rank[u] \leftarrow 0$

Fim.

Vértice	π	*rank*
A	A	0
B	B	0
C	C	0
D	D	0
E	E	0
F	F	0

Árvore Geradora Mínima (Funções)

2. Find(u) → encontrar o pai

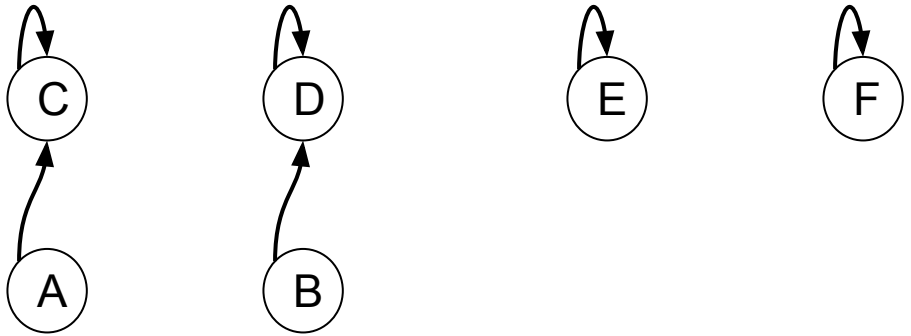


$G = (V, E)$

Find(x)
1. ***enquanto*** *x ≠ π[x]*
2. *x←π[x]*
3. ***fimenquanto***
4. *return x*

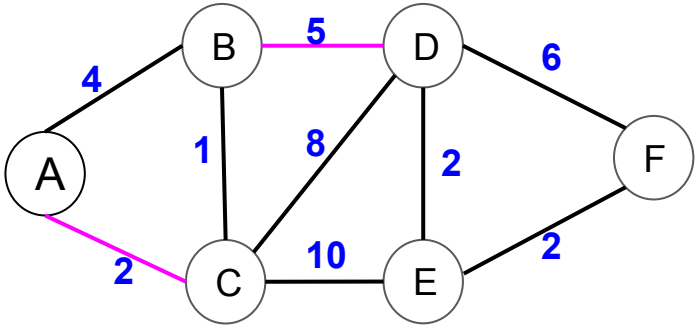
Fim.

Vértice	π	*rank*
A	C	0
B	D	0
C	C	1
D	D	1
E	E	0
F	F	0

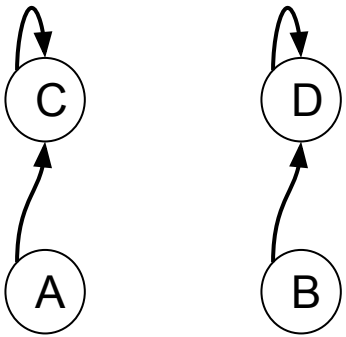


Árvore Geradora Mínima (Funções)

3. Union(u, v) por *rank* →unir conjuntos



$G = (V, E)$

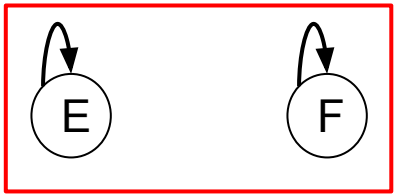


Union(u, v)

1. $P_u \leftarrow Find(u)$
2. $P_v \leftarrow Find(v)$
3. **se** $rank[P_u] > rank[P_v]$
4. $\pi[P_v] \leftarrow P_u$
5. **senão**
6. $\pi[P_u] \leftarrow P_v$
7. **se** $rank[P_u] = rank[P_v]$
8. $rank[P_v] \leftarrow rank[P_v] + 1$
9. **fimse**
10. **fimse**

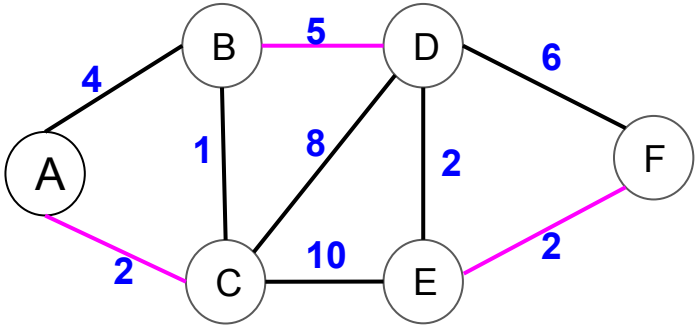
Fim.

Vértice	π	*rank*
A	C	0
B	D	0
C	C	1
D	D	1
E	E	0
F	F	0



Árvore Geradora Mínima (Funções)

3. Union(u, v) por *rank* →unir conjuntos



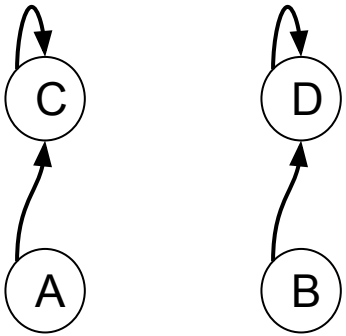
$G = (V, E)$

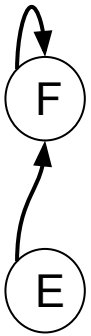
Union(u, v)

1. $P_u \leftarrow Find(u)$
2. $P_v \leftarrow Find(v)$
3. **se** $rank[P_u] > rank[P_v]$
4. $\pi[P_v] \leftarrow P_u$
5. **senão**
6. $\pi[P_u] \leftarrow P_v$
7. **se** $rank[P_u] = rank[P_v]$
8. $rank[P_v] \leftarrow rank[P_v] + 1$
9. **fimse**
10. **fimse**

Fim.

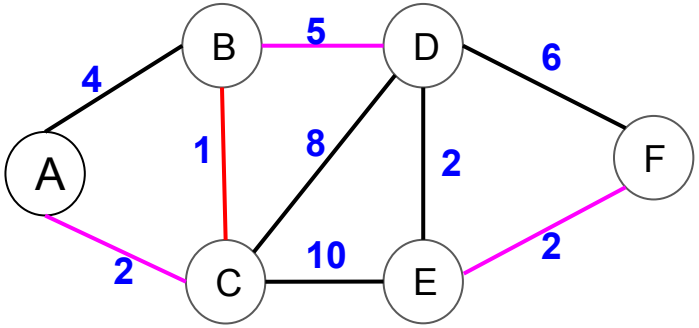
Vértice	π	rank
A	C	0
B	D	0
C	C	1
D	D	1
E	F	0
F	F	1



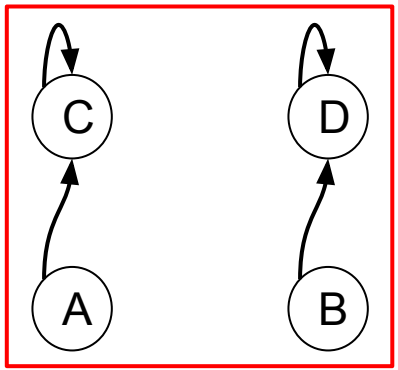


Árvore Geradora Mínima (Funções)

3. Union(u, v) por *rank* →unir conjuntos



$G = (V, E)$

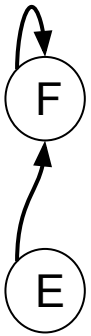


Union(u, v)

1. $P_u \leftarrow Find(u)$
2. $P_v \leftarrow Find(v)$
3. **se** $rank[P_u] > rank[P_v]$
4. $\pi[P_v] \leftarrow P_u$
5. **senão**
6. $\pi[P_u] \leftarrow P_v$
7. **se** $rank[P_u] = rank[P_v]$
8. $rank[P_v] \leftarrow rank[P_v] + 1$
9. **fimse**
10. **fimse**

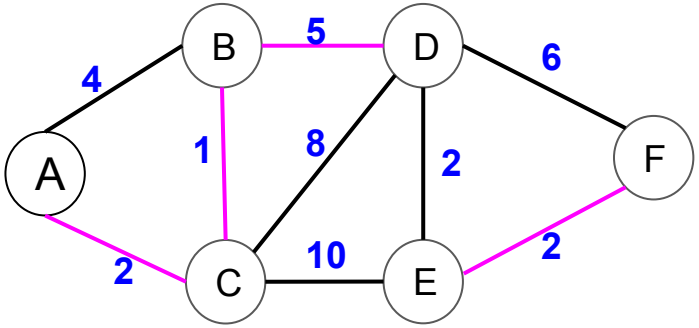
Fim.

Vértice	π	*rank*
A	C	0
B	D	0
C	C	1
D	D	1
E	F	0
F	F	1

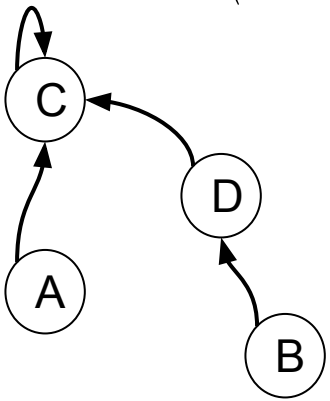


Árvore Geradora Mínima (Funções)

3. Union(u, v) por *rank* →unir conjuntos



$G = (V, E)$

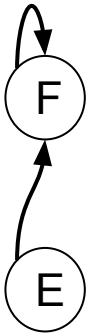


Union(u, v)

1. $P_u \leftarrow Find(u)$
2. $P_v \leftarrow Find(v)$
3. ***se*** $rank[P_u] > rank[P_v]$
4. $\pi[P_v] \leftarrow P_u$
5. ***senão***
6. $\pi[P_u] \leftarrow P_v$
7. ***se*** $rank[P_u] = rank[P_v]$
8. $rank[P_v] \leftarrow rank[P_v] + 1$
9. ***fimse***
10. ***fimse***

Fim.

Vértice	π	*rank*
A	C	0
B	D	0
C	C	2
D	C	1
E	F	0
F	F	1



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

Pseudocódigo do Algoritmo de Kruskal

Estruturas:

- ***rank*** → Altura na árvore
- ***π*** → *Pai (representante)*

Complexidade do Tempo de Execução

- $O(|V|+|E|)lg|V|$

```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

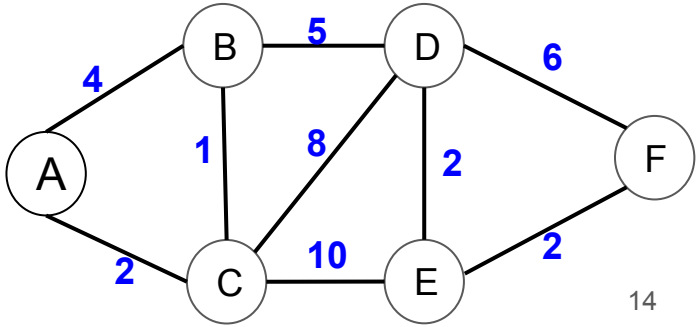
```
Find(x)
1.  enquanto x ≠ π[x]
2.      x←π[x]
3.  fimenquanto
4.  return x
Fim.
```

```
Union(u, v)
1.  P_u ← Find(u)
2.  P_v ← Find(v)
3.  se rank[P_u] > rank[P_v]
4.      π[P_v]←P_u
5.  senão
6.      π[P_u]←P_v
7.      se rank[P_u] = rank[P_v]
8.          rank[P_v] ← rank[P_v] + 1
9.      fimse
10. fimse
Fim.
```

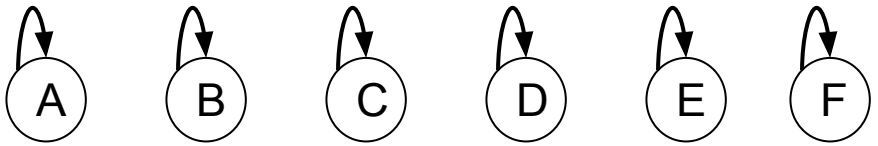
```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T  U {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

AB: **4**
AC: **2**
BC: **1**
BD: **5**
CD: **8**
CE: **10**
DE: **2**
DF: **6**
EF: **2**

Vértice	π	rank
A		
B		
C		
D		
E		
F		



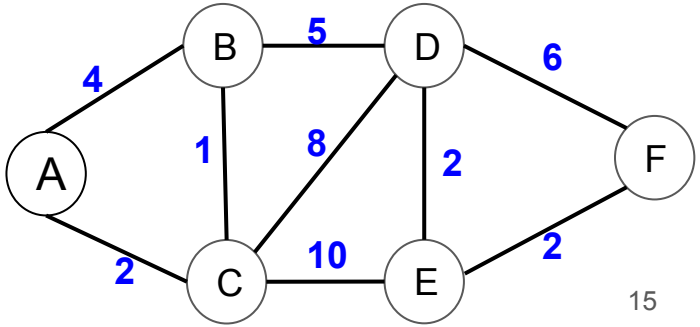
```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.     se Find(u) ≠ Find(v)
9.        T ← T  U {u, v}
10.       Union(u, v)
11.    fimse
12. fimpara
Fim.
```



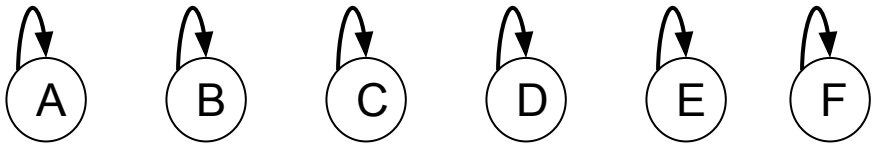
Inicialização do rank e π

AB: **4**
AC: **2**
BC: **1**
BD: **5**
CD: **8**
CE: **10**
DE: **2**
DF: **6**
EF: **2**

Vértice	π	rank
A	A	0
B	B	0
C	C	0
D	D	0
E	E	0
F	F	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T  U {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

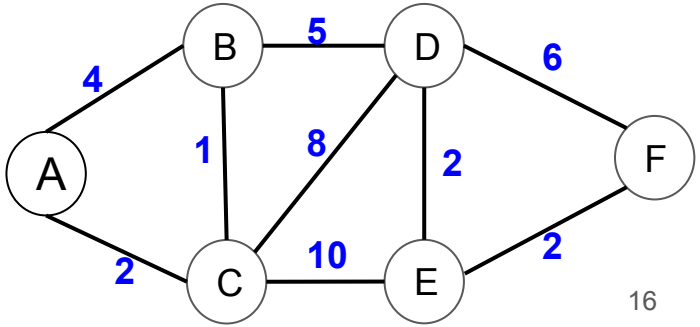


Ordenação das Arestas em ordem Crescente

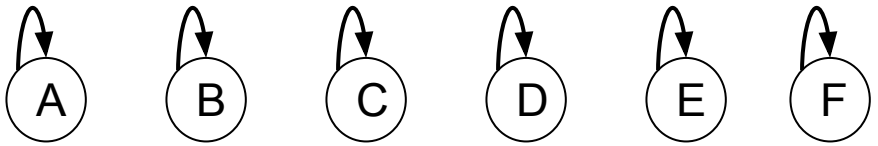
Ordenação

BC : **1**
AC : **2**
DE : **2**
EF : **2**
AB : **4**
BD : **5**
DF : **6**
CD : **8**
CE : **10**

Vértice	π	rank
A	A	0
B	B	0
C	C	0
D	D	0
E	E	0
F	F	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

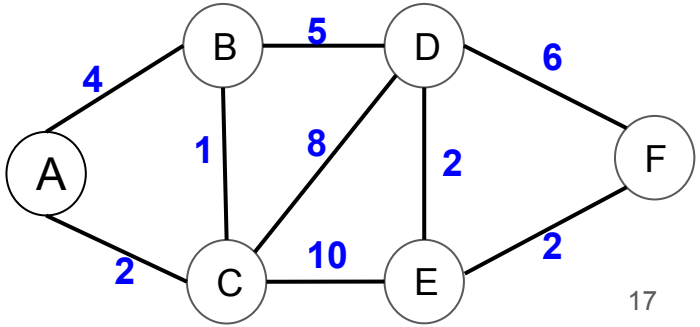


Ordenação

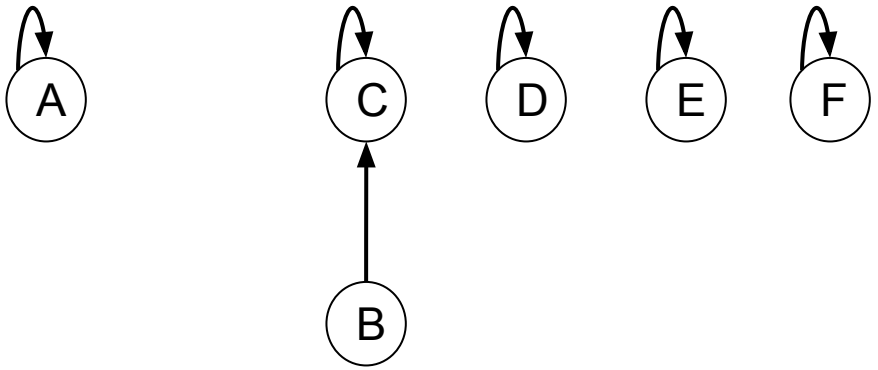
BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	A	0
B	B	0
C	C	0
D	D	0
E	E	0
F	F	0

Vértice → Representante
B → B
C → C
representantes distintos



```
Union(u, v)
1.  P_u ← Find(u)
2.  P_v ← Find(v)
3.  se rank[P_u] > rank[P_v]
4.      π[P_v]←P_u
5.  senão
6.      π[P_u]←P_v
7.      se rank[P_u] = rank[P_v]
8.          rank[P_v] ← rank[P_v] + 1
9.      fimse
10. fimse
Fim.
```

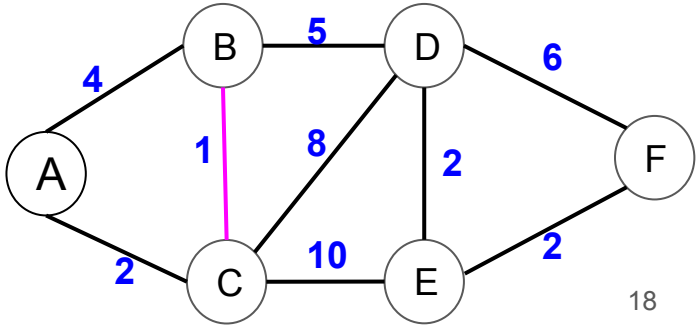


Representante do B: C
Rank do C: 1

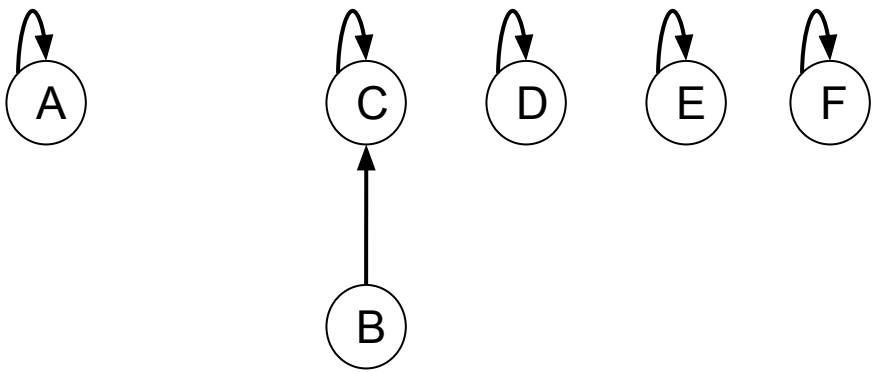
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	A	0
B	C	0
C	C	1
D	D	0
E	E	0
F	F	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

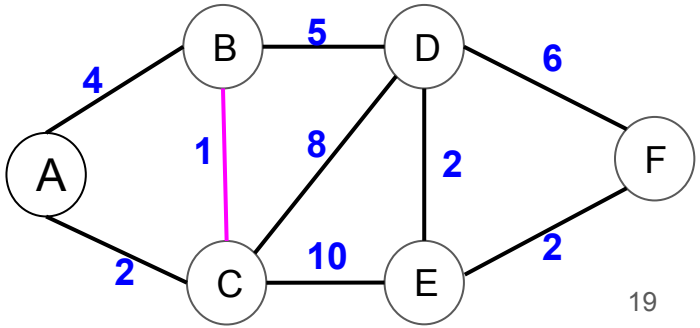


Vértice → Representante
A → A
C → C
representantes distintos

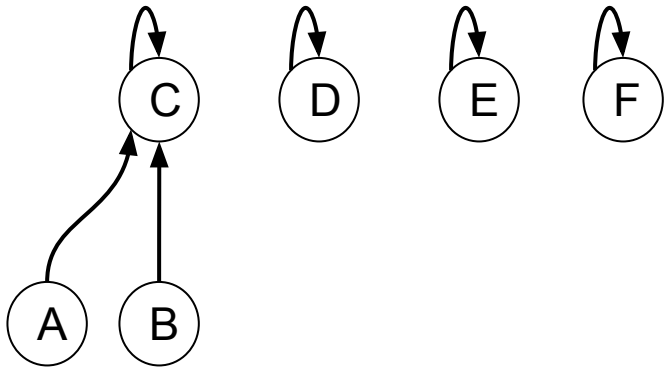
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	A	0
B	C	0
C	C	1
D	D	0
E	E	0
F	F	0



```
Union(u, v)
1.  P_u ← Find(u)
2.  P_v ← Find(v)
3.  se rank[P_u] > rank[P_v]
4.      π[P_v]←P_u
5.  senão
6.      π[P_u]←P_v
7.      se rank[P_u] = rank[P_v]
8.          rank[P_v] ← rank[P_v] + 1
9.      fimse
10. fimse
Fim.
```

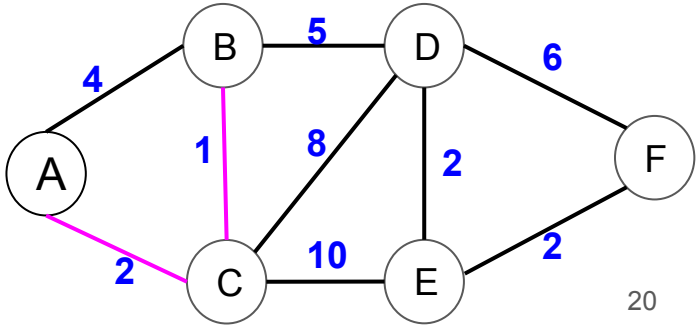


Representante do A: C

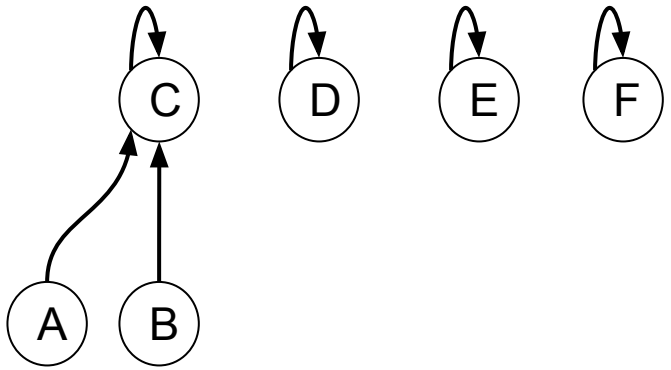
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	C	1
D	D	0
E	E	0
F	F	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

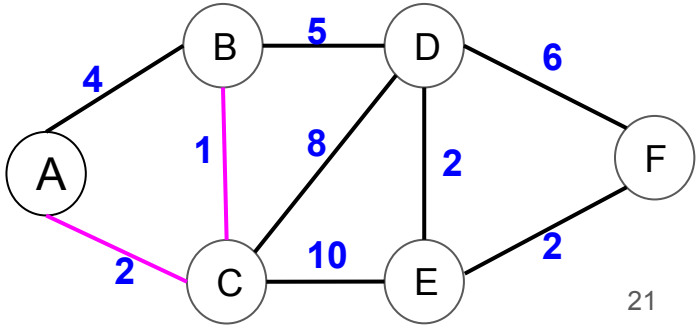


Vértice → Representante
D → D
E → E
representantes distintos

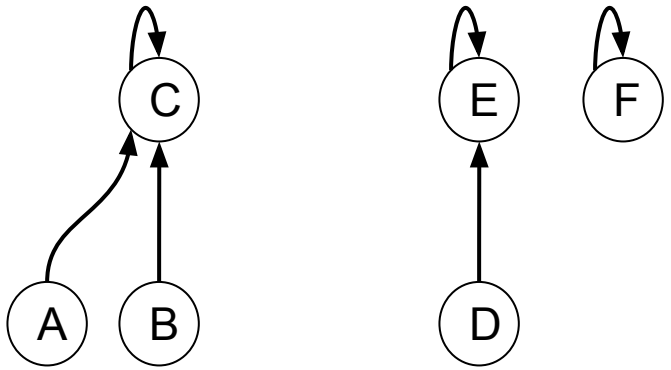
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	C	1
D	D	0
E	E	0
F	F	0



```
Union(u, v)
1.   P_u ← Find(u)
2.   P_v ← Find(v)
3.   se rank[P_u] > rank[P_v]
4.        π[P_v]←P_u
5.   senão
6.        π[P_u]←P_v
7.        se rank[P_u] = rank[P_v]
8.             rank[P_v] ← rank[P_v] + 1
9.        fimse
10.  fimse
Fim.
```

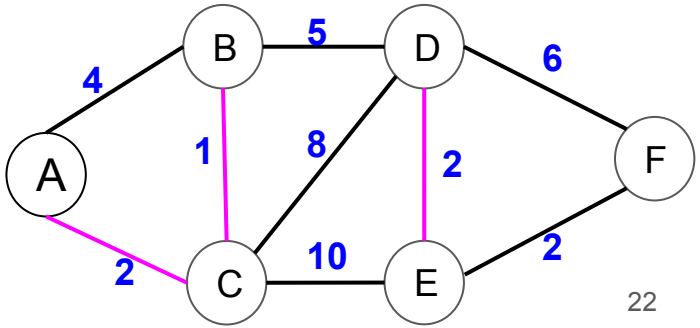


Representante do D: E
Rank do E: 1

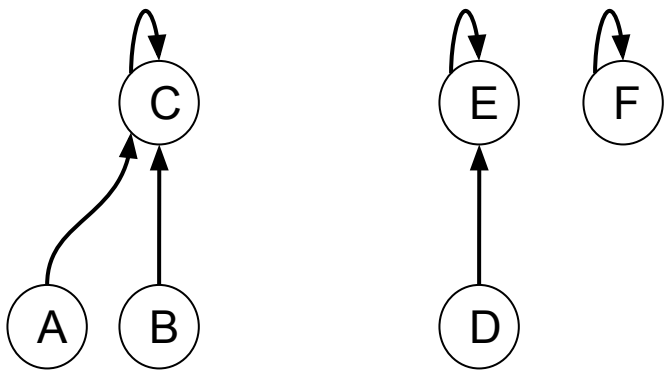
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	C	1
D	E	0
E	E	1
F	F	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

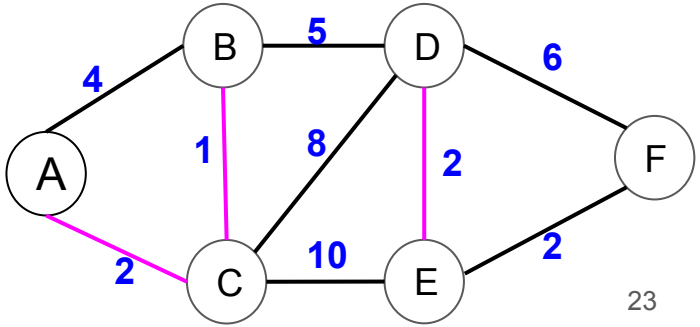


Vértice → Representante
E → E
F → F
representantes distintos

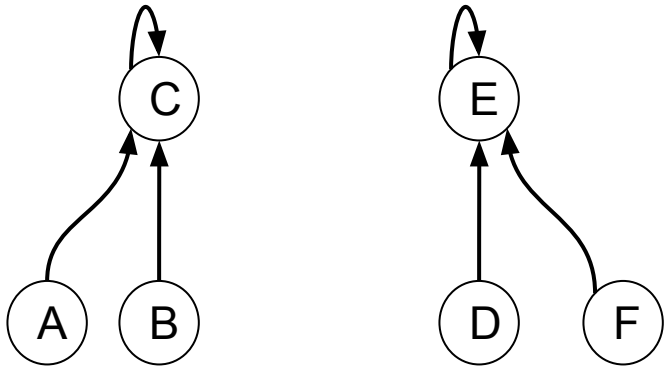
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	C	1
D	E	0
E	E	1
F	F	0



```
Union(u, v)
1.   P_u ← Find(u)
2.   P_v ← Find(v)
3.   se rank[P_u] > rank[P_v]
4.        π[P_v]←P_u
5.   senão
6.        π[P_u]←P_v
7.        se rank[P_u] = rank[P_v]
8.             rank[P_v] ← rank[P_v] + 1
9.        fimse
10.  fimse
Fim.
```

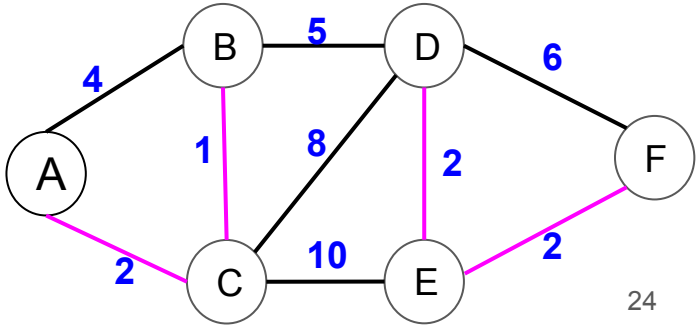


Representante do F: E

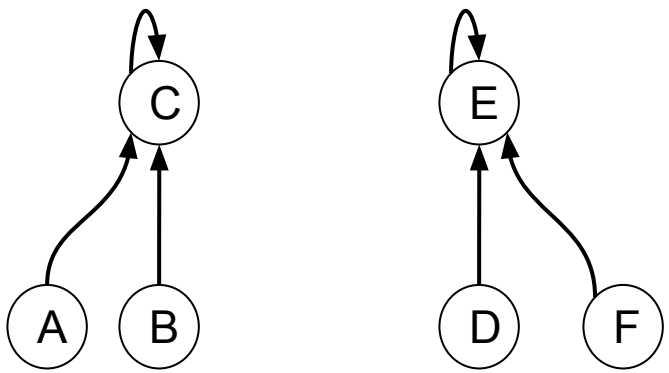
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	C	1
D	E	0
E	E	1
F	E	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

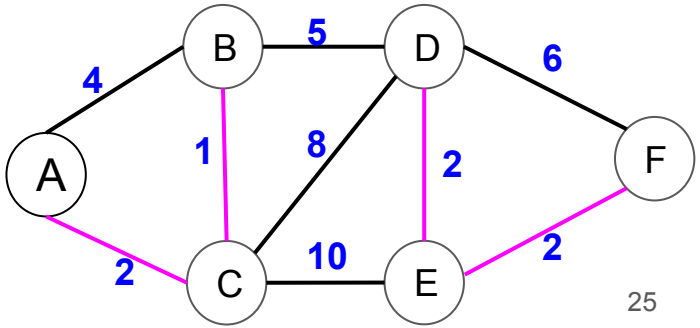


Vértice → Representante
A → C
B → C
o mesmo representante

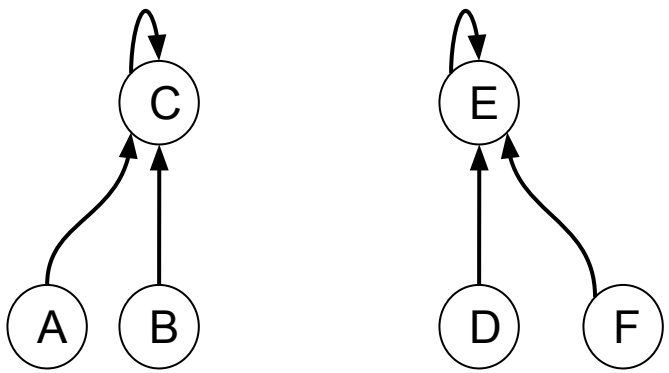
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	C	1
D	E	0
E	E	1
F	E	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

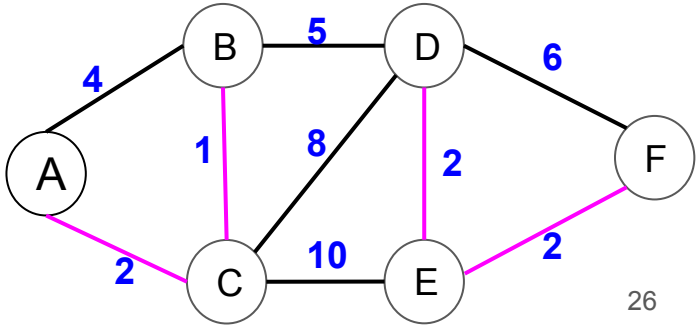


Vértice → Representante
B → C
D → E
representantes distintos

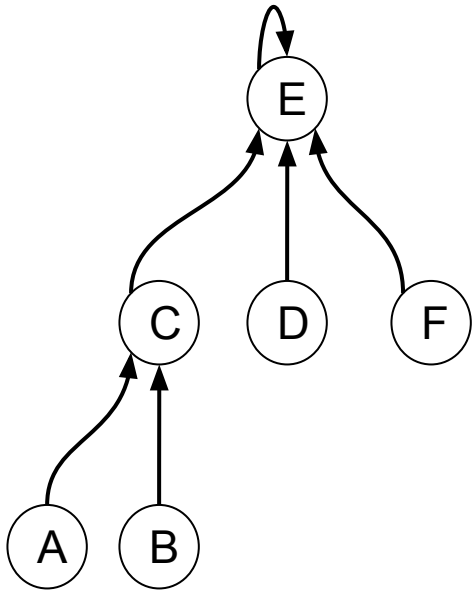
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	C	1
D	E	0
E	E	1
F	E	0



```
Union(u, v)
1.  P_u ← Find(u)
2.  P_v ← Find(v)
3.  se rank[P_u] > rank[P_v]
4.      π[P_v]←P_u
5.  senão
6.      π[P_u]←P_v
7.      se rank[P_u] = rank[P_v]
8.          rank[P_v] ← rank[P_v] + 1
9.      fimse
10. fimse
Fim.
```

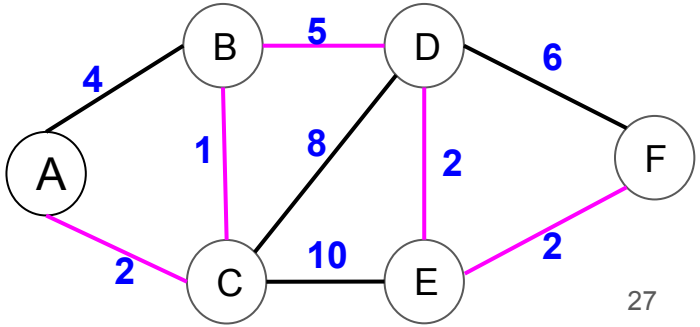


Representante do C: E
Rank do E: 2

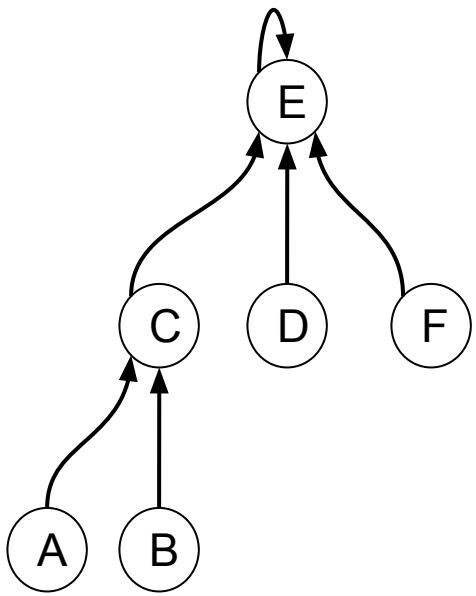
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	E	1
D	E	0
E	E	2
F	E	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

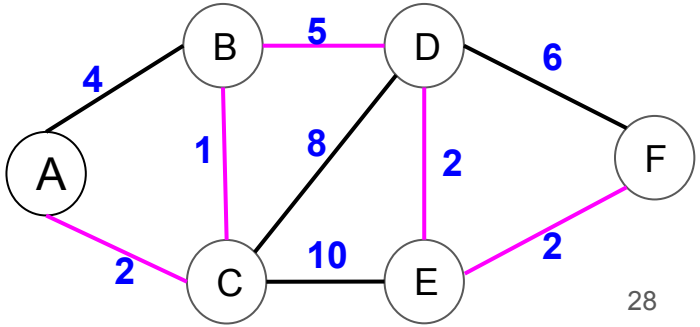


Vértice → Representante
D → E
F → E
o mesmo representante

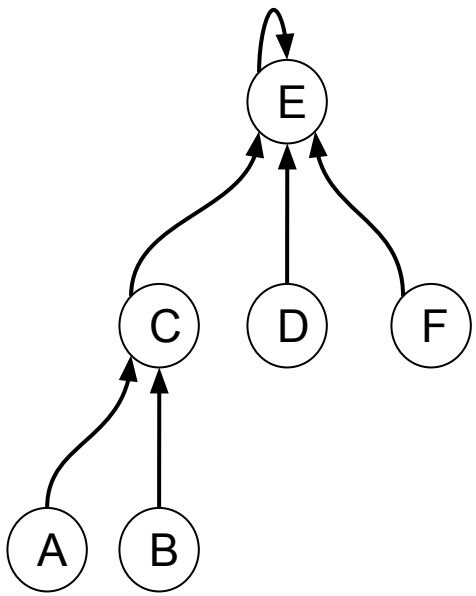
Ordenação

BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	E	1
D	E	0
E	E	2
F	E	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.      se Find(u) ≠ Find(v)
9.          T ← T ∪ {u, v}
10.         Union(u, v)
11.     fimse
12. fimpara
Fim.
```

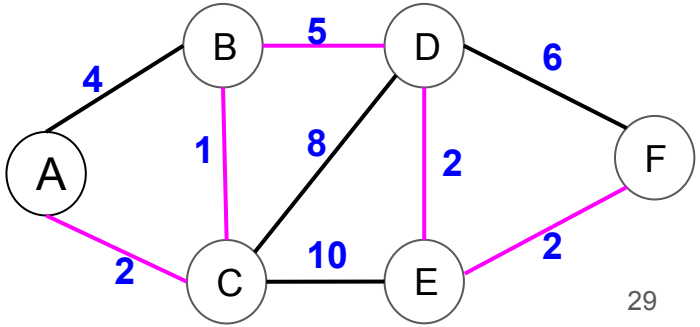


Vértice → Representante
C → E
D → E
o mesmo representante

Ordenação

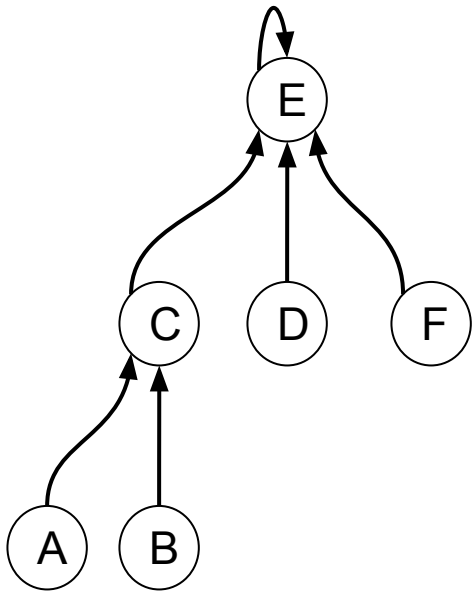
BC : 1
AC : 2
DE : 2
EF : 2
AB : 4
BD : 5
DF : 6
CD : 8
CE : 10

Vértice	π	rank
A	C	0
B	C	0
C	E	1
D	E	0
E	E	2
F	E	0



```
Kruskal(G, w)
1.  para cada vértice u ← V[G]
2.      π[u]←u
3.      rank[u]← 0
4.  fimpara
5.  T ← ∅
6.  OrdenarArestas(G), ordem crescente
7.  para cada aresta {u, v} ∈ E na ordem
8.     se Find(u) ≠ Find(v)
9.        T ← T  U {u, v}
10.       Union(u, v)
11.    fimse
12. fimpara
Fim.
```

Peso da AGM
12

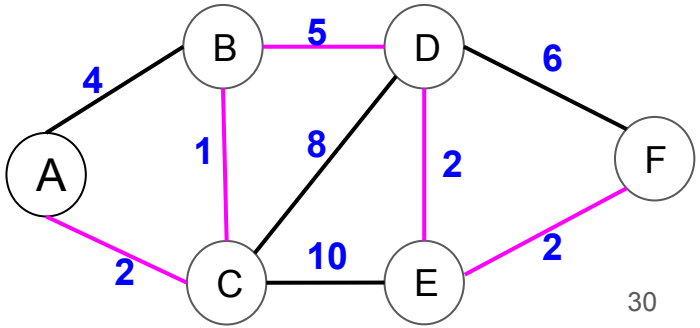


Vértice → Representante
C → E
E → E
o mesmo representante

Ordenação

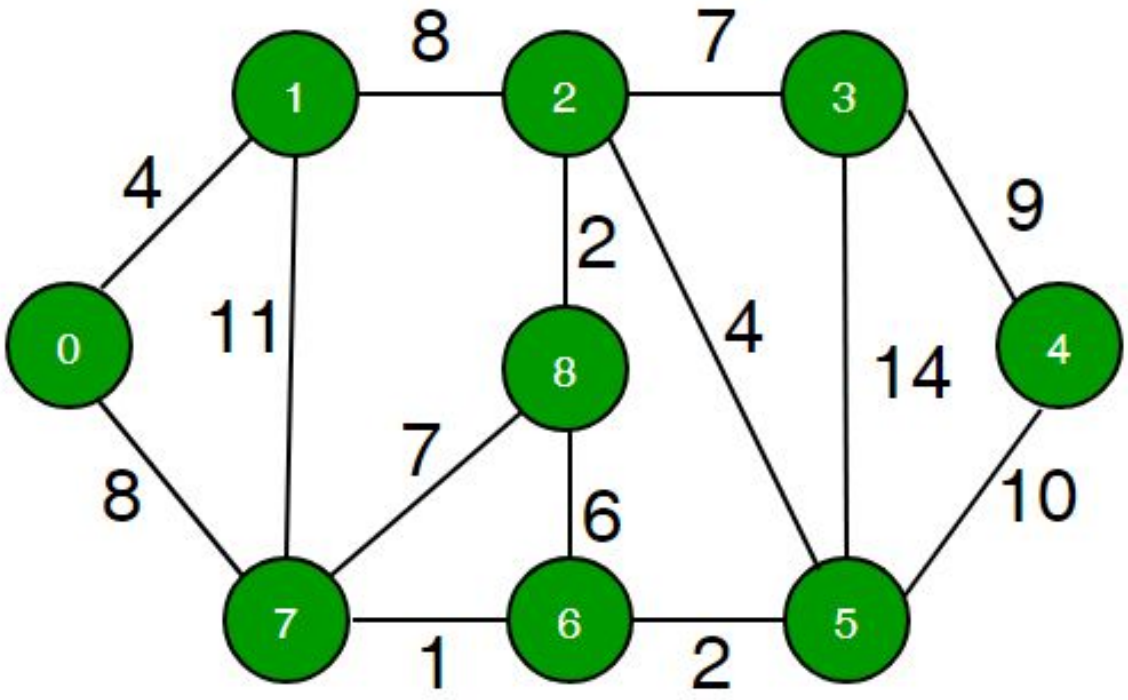
BC : **1**
AC : **2**
DE : **2**
EF : **2**
AB : **4**
BD : **5**
DF : **6**
CD : **8**
CE : **10**

Vértice	π	rank
A	C	0
B	C	0
C	E	1
D	E	0
E	E	2
F	E	0



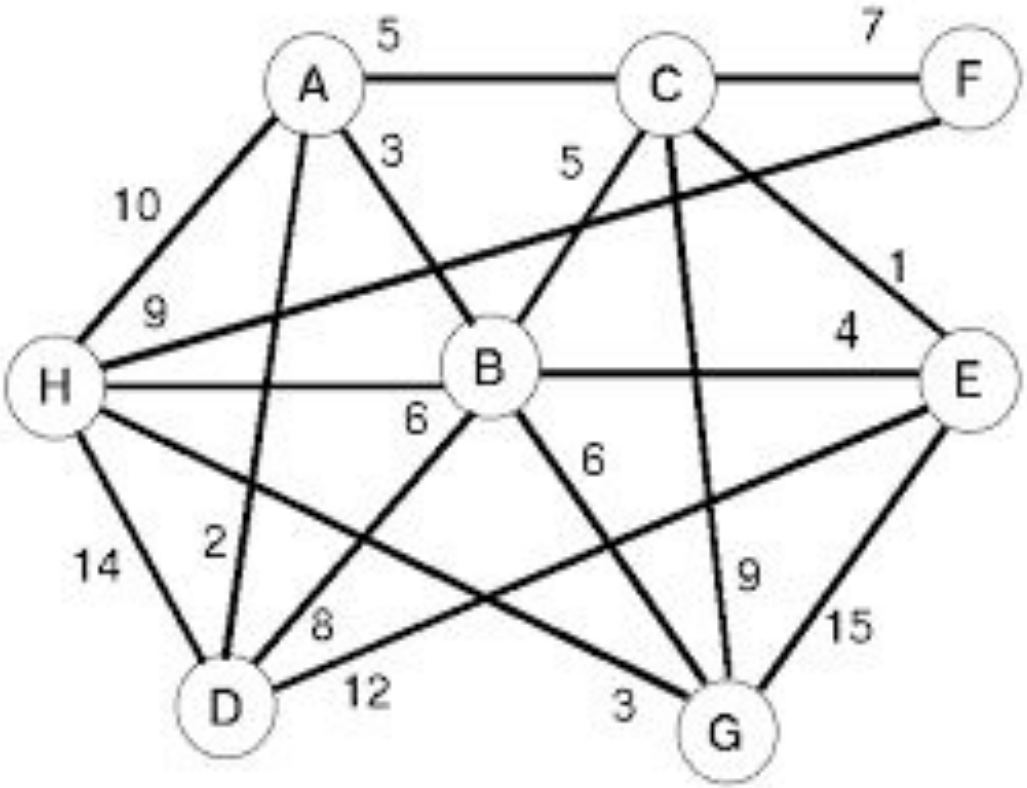
Exercício de Fixação

Encontre a árvore geradora mínima.



Exercício de Fixação

Encontre a árvore geradora mínima.



UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

Algoritmos em Grafos

Aula 15: Árvore Geradora Mínima(Kruskal)

Professor Pablo Soares

2022.1