Universidade Federal de Ouro Preto
Departamento de Computação - DECOM
BCC241 - Projeto e Análise de Algoritmos
Prof. Anderson Almeida Ferreira

Nome: Caio Silas de Araujo Amaro
Matrícula: 21.1.4111

- Algoritmo de Kruskal:

```
procedure kruskal(G, w)
Input:    A connected undirected graph G = (V, E) with edge weights w_e
Output:   A minimum spanning tree defined by the edges X

for all u ∈ V:
   makeset(u)

X = {}
Sort the edges E by weight
for all edges {u, v} ∈ E, in increasing order of weight:
   if find(u) ≠ find(v):
      add edge {u, v} to X
      union(u, v)
```

```
procedure union(x, y)
r_x = find(x)
r_y = find(y)
if r_x = r_y:  return
if rank(r_x) > rank(r_y):
   π(r_y) = r_x
else:
   π(r_x) = r_y
   if rank(r_x) = rank(r_y):  rank(r_y) = rank(r_y) + 1
```

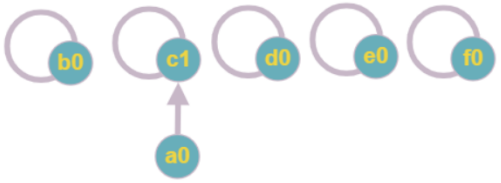
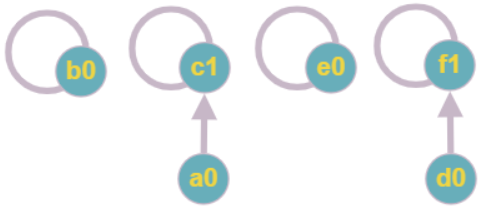
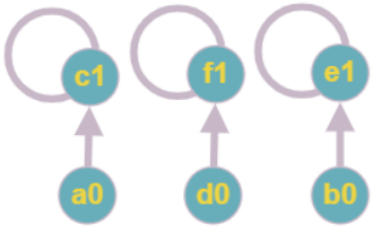
makeset -> representação de conjuntos disjuntos: (nó, rank, pai)

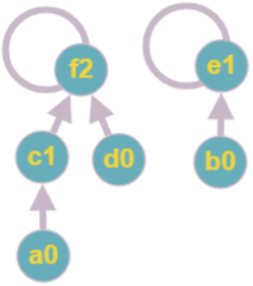
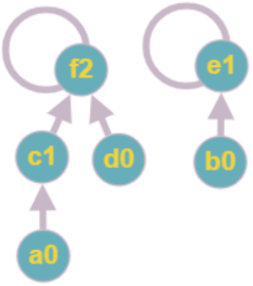
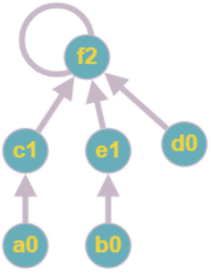
(a, 0, a), (b, 0, b), (c, 0, c), (d, 0, d), (e, 0, e), (f, 0, f)

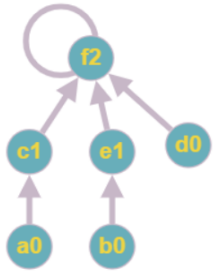
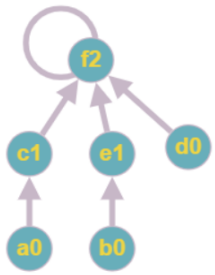
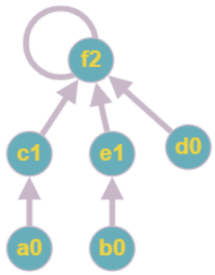
X = {}

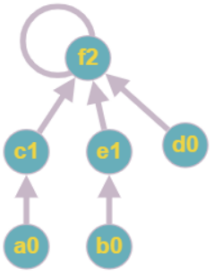
arestas ordenadas -> {{a,c}, {d,f}, {b,e}, {c,f}, {a,d}, {b,c}, {c,d}, {a,b}, {c,e}, {e,f}}

Aresta	Execução
{a,c}	representante de a = a representante de c = c representantes diferentes, faz a união: X <- X ∪ {a, c} rank(a) = rank(c)

	pai(a) <- c rank(c) <- rank(c) + 1 (a, 0, c), (b, 0, b), (c, 1, c), (d, 0, d), (e, 0, e), (f, 0, f)  X = {{a,c}}
{d,f}	representante de d = d representante de f = f representantes diferentes, faz a união: X <- X ∪ {d, f} rank(d) = rank(f) pai(d) <- f rank(f) <- rank(f) + 1 (a, 0, c), (b, 0, b), (c, 1, c), (d, 0, f), (e, 0, e), (f, 1, f)  X = {{a,c}, {d, f}}
{b,e}	representante de b = b representante de e = e representantes diferentes, faz a união: X <- X ∪ {b, e} rank(b) = rank(e) pai(b) <- e rank(e) <- rank(e) + 1 (a, 0, c), (b, 0, e), (c, 1, c), (d, 0, f), (e, 1, e), (f, 1, f)  X = {{a,c}, {d, f}, {b, e}}
{c,f}	representante de c = c representante de f = f representantes diferentes, faz a união: X <- X ∪ {c, f}

	rank(c) = rank(f) pai(c) <- f rank(f) <- rank(f) + 1 (a, 0, c), (b, 0, e), (c, 1, f), (d, 0, f), (e, 1, e), (f, 2, f)  X = {{a,c}, {d, f}, {b, e}, {c, f}}
{a,d}	representante de a = f representante de d = f representantes iguais, não faz a união (a, 0, c), (b, 0, e), (c, 1, f), (d, 0, f), (e, 1, e), (f, 2, f)  X = {{a,c}, {d, f}, {b, e}, {c, f}}
{b,c}	representante de b = e representante de c = f representantes diferentes, faz a união: X <- X ∪ {b, c} rank(e) < rank(f) pai(e) <- f (a, 0, c), (b, 0, e), (c, 1, f), (d, 0, f), (e, 1, f), (f, 2, f)  X = {{a,c}, {d, f}, {b, e}, {c, f}, {b, c}}

{c,d}	representante de c = f representante de d = f representantes iguais, não faz a união (a, 0, c), (b, 0, e), (c, 1, f), (d, 0, f), (e, 1, f), (f, 2, f)  X = {{a,c}, {d, f}, {b, e}, {c, f}, {b, c}}
{a,b}	representante de a = f representante de b = f representantes iguais, não faz a união (a, 0, c), (b, 0, e), (c, 1, f), (d, 0, f), (e, 1, f), (f, 2, f)  X = {{a,c}, {d, f}, {b, e}, {c, f}, {b, c}}
{c,e}	representante de c = f representante de e = f representantes iguais, não faz a união (a, 0, c), (b, 0, e), (c, 1, f), (d, 0, f), (e, 1, f), (f, 2, f)  X = {{a,c}, {d, f}, {b, e}, {c, f}, {b, c}}
{e,f}	representante de e = f representante de f = f

	representantes iguais, não faz a união (a, 0, c), (b, 0, e), (c, 1, f), (d, 0, f), (e, 1, f), (f, 2, f)  X = {{a,c}, {d, f}, {b, e}, {c, f}, {b, c}}

- Algoritmo de Prim

```
procedure prim(G, w)
Input:    A connected undirected graph G = (V, E) with edge weights w_e
Output:   A minimum spanning tree defined by the array prev

for all u ∈ V:
   cost(u) = ∞
   prev(u) = nil
Pick any initial node u_0
cost(u_0) = 0

H = makequeue(V)      (priority queue, using cost-values as keys)
while H is not empty:
   v = deletemin(H)
   for each {v, z} ∈ E:
      if cost(z) > w(v, z):
         cost(z) = w(v, z)
         prev(z) = v
         decreasekey(H, z)
```

Set S	a	b	c	d	e	f
{}	0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
{a}		6/a	1/a	5/a	∞/nil	∞/nil
{a, c}		5/c		5/a	6/c	4/c
{a, c, f}		5/c		2/f	6/c	
{a, c, f, d}		5/c			6/c	
{a, c, f, d, b}					3/b	
{a, c, f, d, b, e}						