Descrição

O Algoritmo de busca por Árvore Geradora (BFS: Breath-first search)

Input

G grafo conexo com n vértices $v_1, \ldots, v_n$

Output

uma árvore geradora T

Pseudocódigo

```
bfs(V,E){
        // V = vértices ordenadas v_1,...v_n de G
        // E = arestas
        // V' = vértices da árvore geradora T
        // E' = arestas da árvore geradora T
        // v_1 = raiz de T
        // S = lista ordenada

        S = (v_1)
        V' = {v_1}
        E' = NULL

        While(true){
                for each X ∈ S,in order,
                        for each y ∈ V\V', in order,
                                f(x,y) ∈ E' ∪ {(x,y)}, V' = V' ∪ {V}
                if no edges are added
                        return T
                S = children of S ordered
        }
}
```

Inicialização

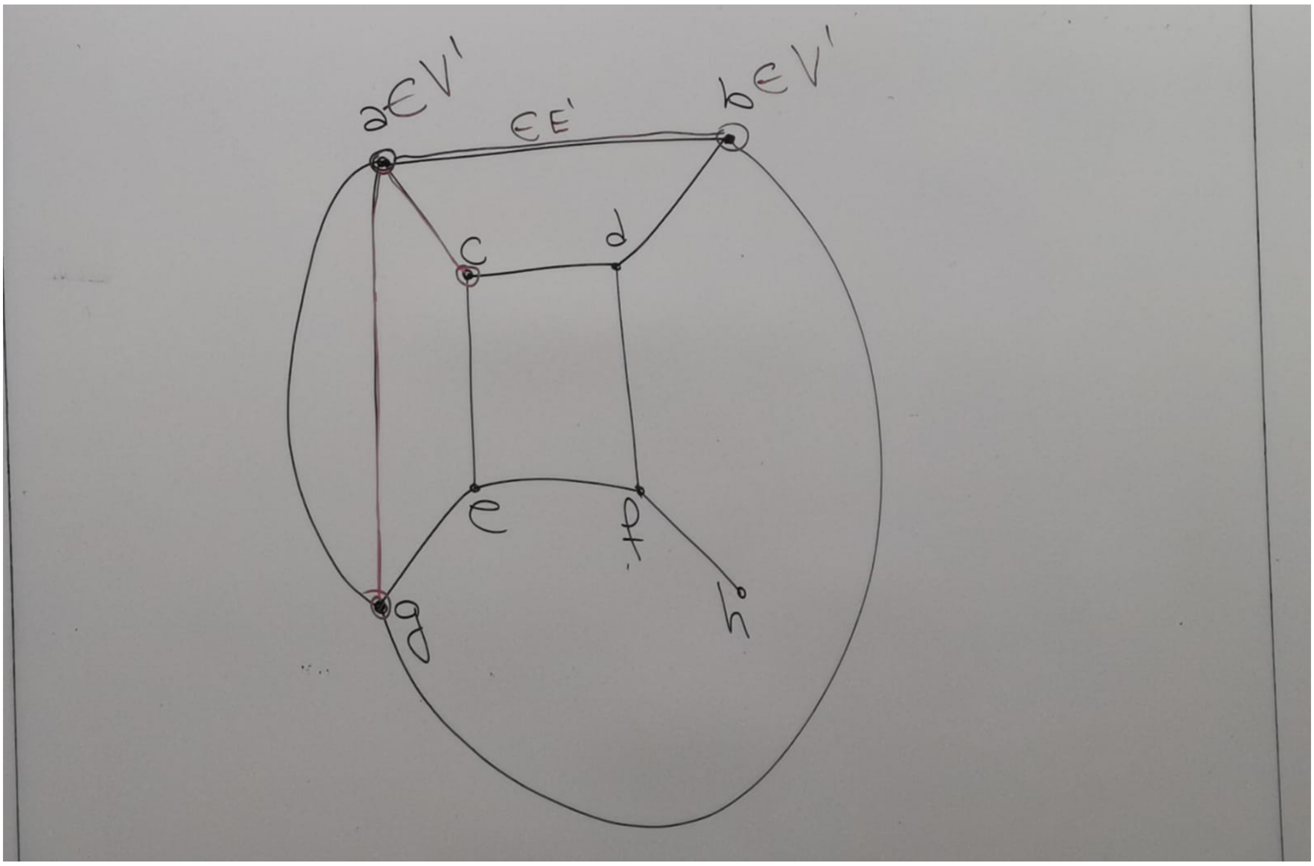
- Declare uma lista S e insira o vértice inicial v_1.
- Inicialize o array de vértices visitados V' e marque o vértice inicial v_1 como visitado.
- Inicialize o array de arestas E' como nulo.

Iterações

Siga o processo até a lista V' tiver todos os vértices de G.

- Adicione os vértices vizinhos na lista de vértices visitados V'
- Inserir as arestas entre x e os vértices vizinhos NÃO-visitados na lista de arestas E' seguindo a ordem pré-estabelecida.

Exemplo 1



Lista Ordenada de vértices:
(a, b, c, d, e, f, g, h)

S = (a)
V' = {a}
E' = ϕ

$V' = \{a, b\}$
$E' = \{(a, b)\}$

$V' = \{a, b, c\}$
$E' = \{(a, b), (a, c)\}$

$V' = \{a, b, c, d\}$
$E' = \{(a, b), (a, c), (a, g)\}$

$S = (b, c, g)$ //filhos de S

---

a (yes)

b
$V' = \{a, b, c, g, e\}$

$E' = \{(a,b),(a,c),(a,g),(b,d)\}$

c
$V' = \{a,b,c,g,d,e\}$
$E' = E' \cup (c,e)$

g
$S = (d,e)$

$V' = V' \cup f = \{a,b,c,d,g,d,e,f\}$
$E' = E' \cup (d,f)$
$S = (f)$

$V' = V' \cup h = V$
$E' = E' \cup (f,h)$
$S = (h)$

Output: árvore geradora $T' = (V', E')$

Exemplo 2

In general, a graph will have several spanning trees. Another spanning tree of the graph G of Figure 9.3.1 is shown in Figure 9.3.2.

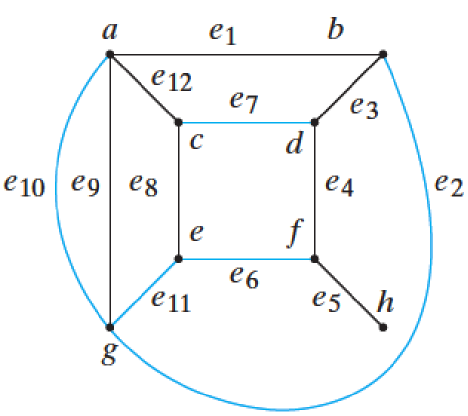


Figure 9.3.1 A graph and a spanning tree shown in black.

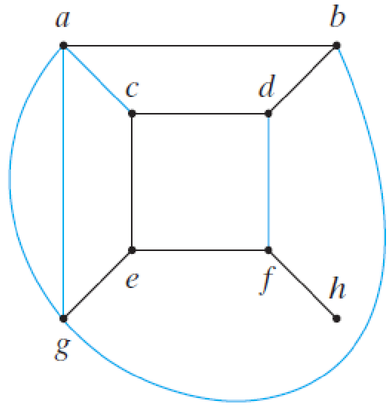


Figure 9.3.2 Another spanning tree (in black) of the graph of Figure 9.3.1.

Find a spanning tree for the graph G of Figure 9.3.1.

We will use a method called **breadth-first search** (Algorithm 9.3.6). The idea of breadth-first search is to process all the vertices on a given level before moving to the next-higher level.

First, select an ordering, say *abcdefgh*, of the vertices of G. Select the first vertex a and label it the root. Let T consist of the single vertex a and no edges. Add to T all edges (a, x) and vertices on which they are incident, for $x = b$ to h, that do not produce a cycle when added to T. We would add to T edges (a, b), (a, c), and (a, g). (We could use either of the parallel edges incident on a and g.) Repeat this procedure with the vertices on level 1 by examining each in order:

b: Include (b, d).

c: Include (c, e).

g: None

Repeat this procedure with the vertices on level 2:

d: Include (d, f).

e: None

Repeat this procedure with the vertices on level 3:

f: Include (f, h).

Since no edges can be added to the single vertex h on level 4, the procedure ends. We have found the spanning tree shown in Figure 9.3.1.

Árvore geradora minimal retornada:

$$E' = \{e_1, e_2, e_9, e_3, e_8, e_4, e_5\}$$