Grafos e Algoritmos Computacionais

Busca em Grafos
Busca em Profundidade: Introdução

Prof. André Britto

Busca em Grafos

- **Busca em Árvore**
  - Problema simples
  - Semelhante a algoritmos já vistos em Estrutura de Dados.

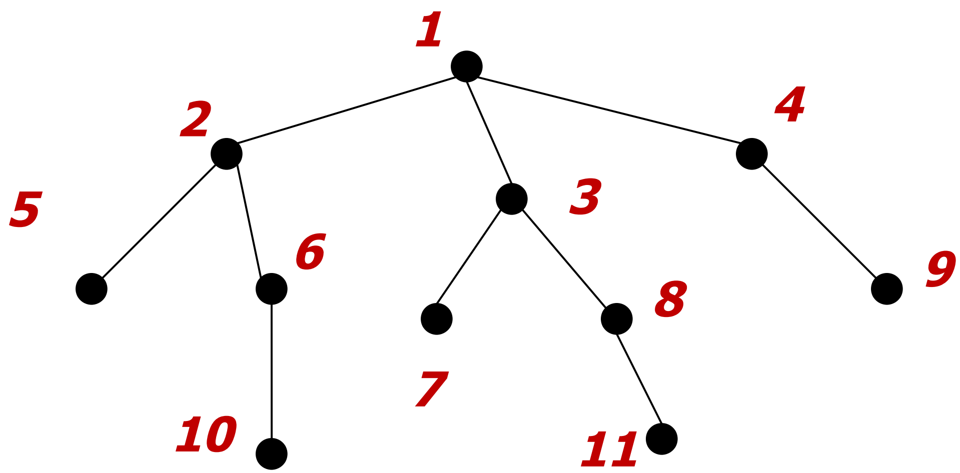
Busca em Árvore

```
algoritmo BuscaArvore
início
   Se árvore vazia então não faça nada
      Senão
         início
            caminhe pela subárvore mais a esquerda da raiz;
            após pela 2ª mais a esquerda;
            após pela 3ª mais a esquerda;
            e assim por diante;
         fim
fim
```

Busca em Árvore

- Que caminhamento é esse?

Ex.:

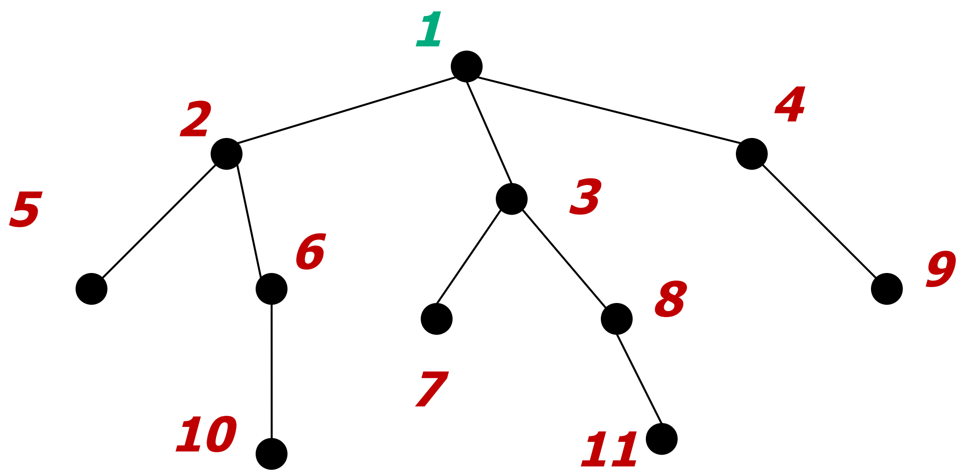


- Caminhamento :

Busca em Árvore

- Que caminhamento é esse?

Ex.:

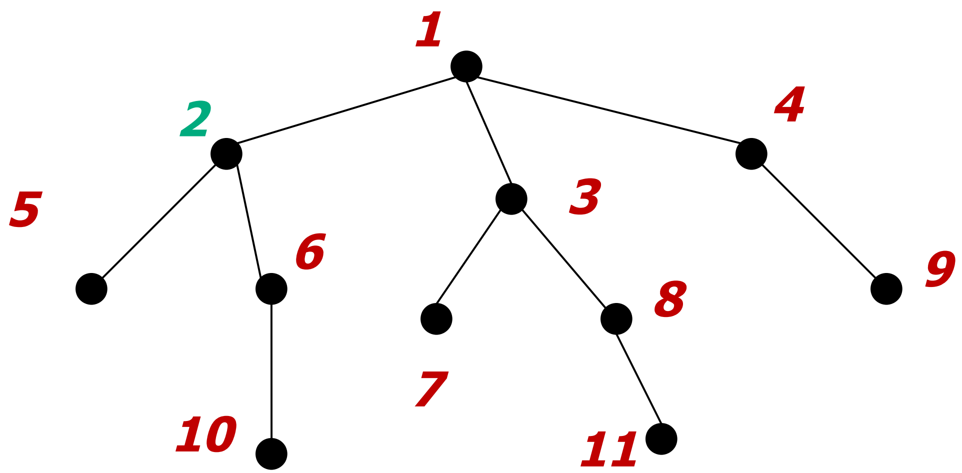


- Caminhamento : 1

Busca em Árvore

- Que caminhamento é esse?

Ex.:

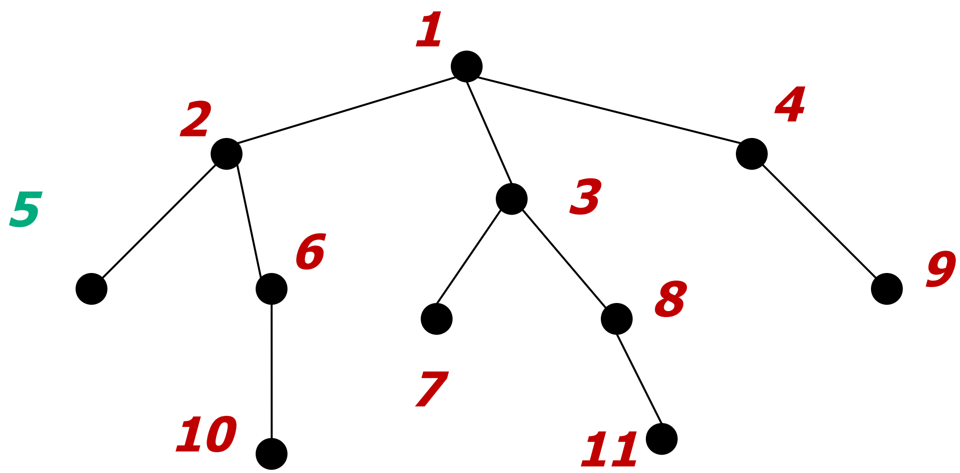


- Caminhamento : ***1 2***

Busca em Árvore

- Que caminhamento é esse?

Ex.:

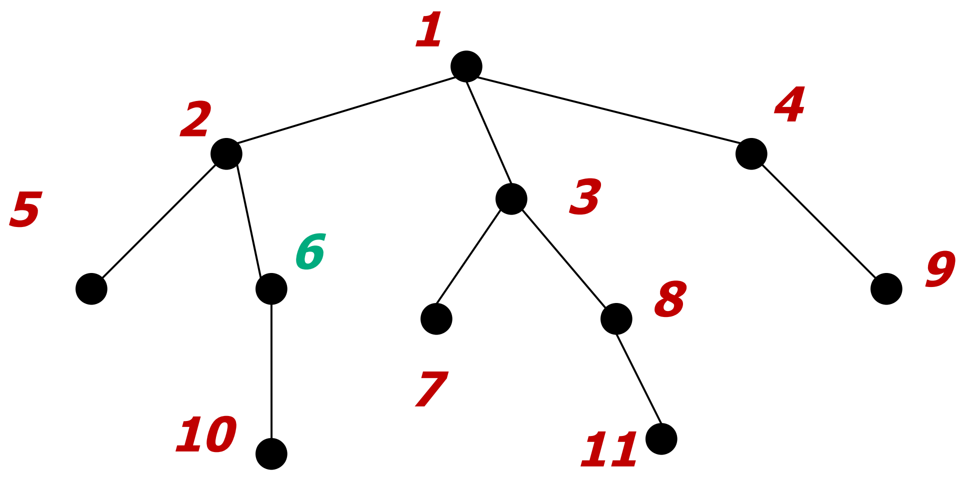


- Caminhamento : ***1 2 5***

Busca em Árvore

- Que caminhamento é esse?

Ex.:

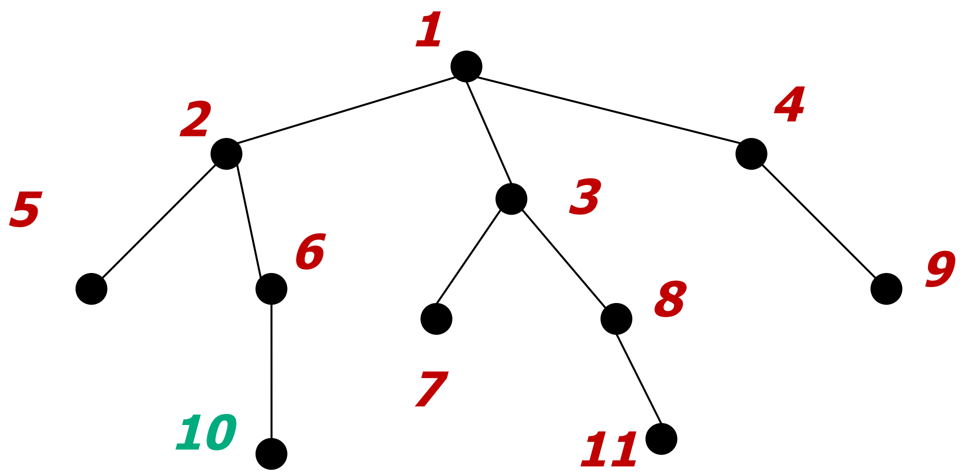


- Caminhamento : *1 2 5 6*

Busca em Árvore

- Que caminhamento é esse?

Ex.:

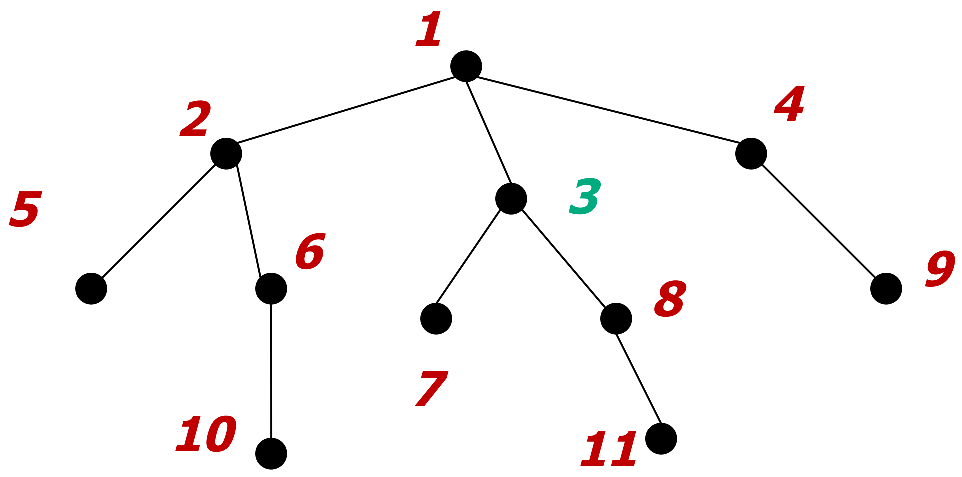


- Caminhamento : ***1 2 5 6 10***

Busca em Árvore

- Que caminhamento é esse?

Ex.:

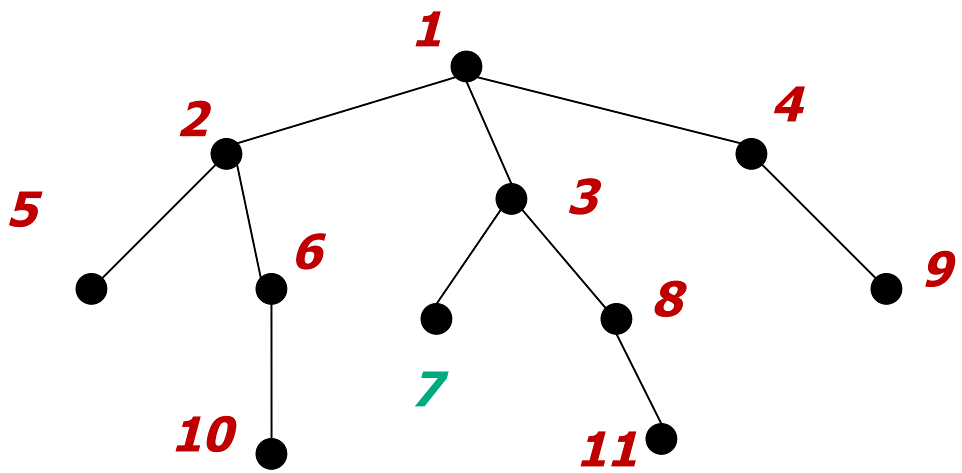


- Caminhamento : *1 2 5 6 10 3*

Busca em Árvore

- Que caminhamento é esse?

Ex.:

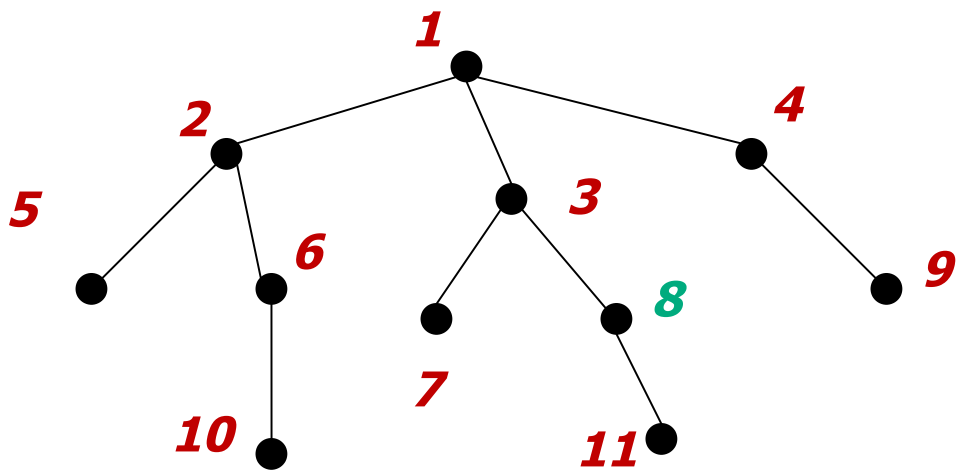


- Caminhamento : *1 2 5 6 10 3 7*

Busca em Árvore

- Que caminhamento é esse?

Ex.:

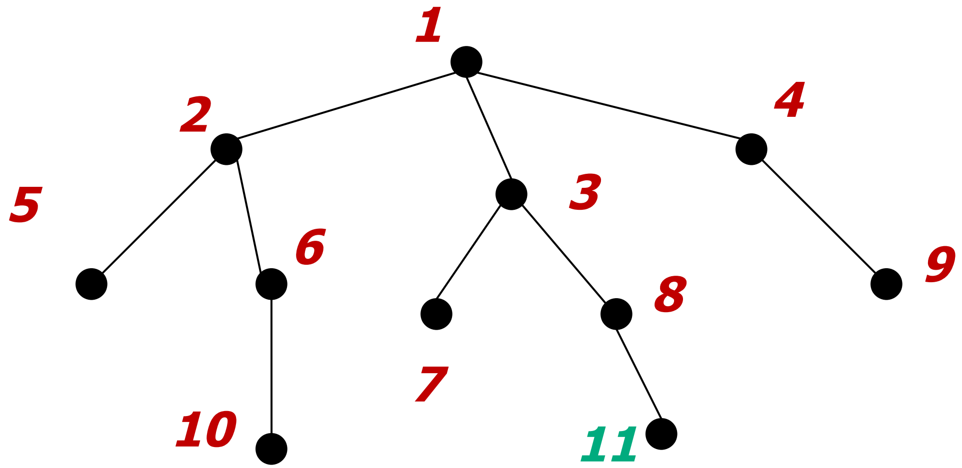


- Caminhamento : ***1 2 5 6 10 3 7 8***

Busca em Árvore

- Que caminhamento é esse?

Ex.:

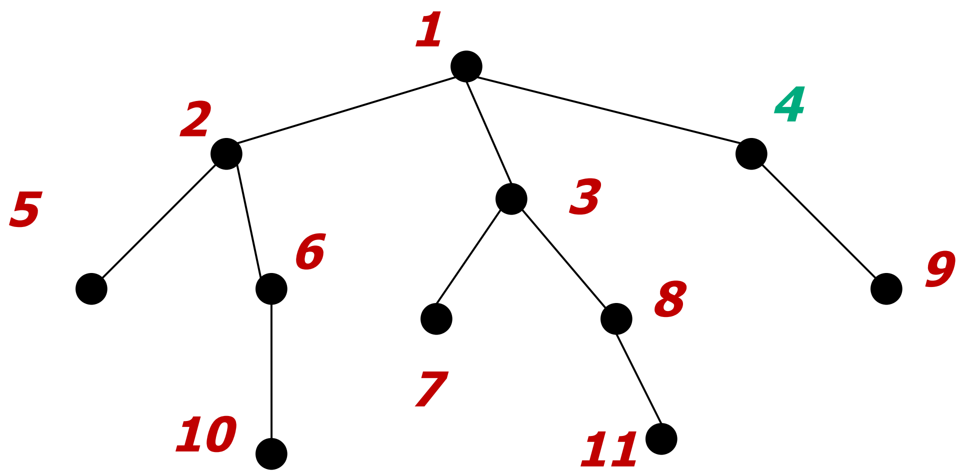


- Caminhamento : ***1 2 5 6 10 3 7 8 11***

Busca em Árvore

- Que caminhamento é esse?

Ex.:

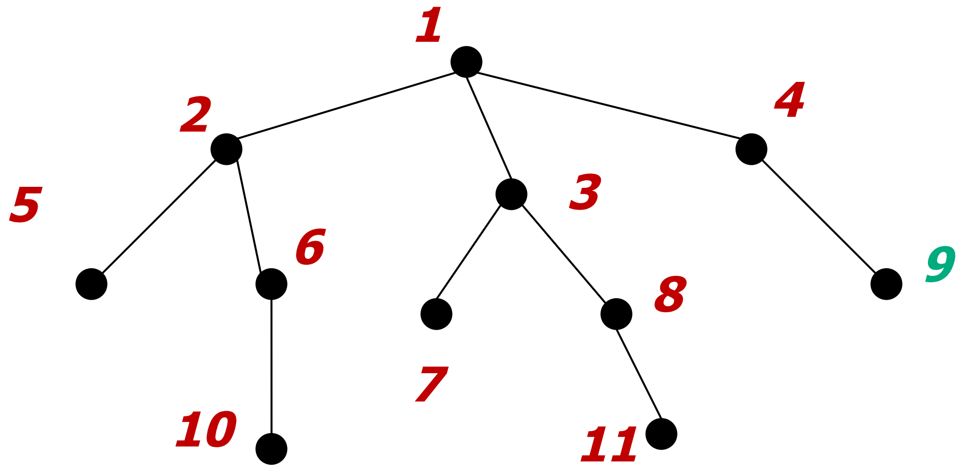


- Caminhamento : ***1 2 5 6 10 3 7 8 11 4***

Busca em Árvore

- Que caminhamento é esse?

Ex.:

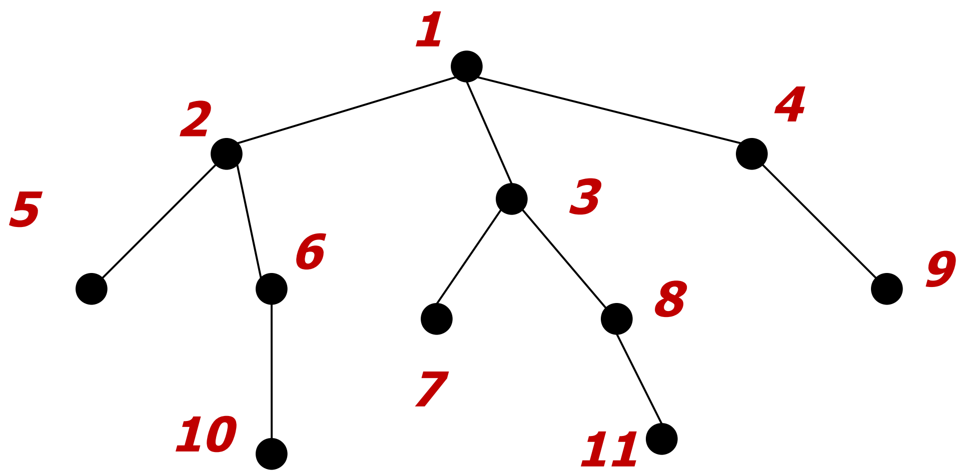


- Caminhamento : ***1 2 5 6 10 3 7 8 11 4 9***

Busca em Árvore

- Que caminhamento é esse?

Ex.:



- Caminhamento : ***1 2 5 6 10 3 7 8 11 4 9***

- **Busca em nível** ou **largura**

  Ex.: ***1 2 3 4 5 6 7 8 9 10 11***

Busca em Grafos

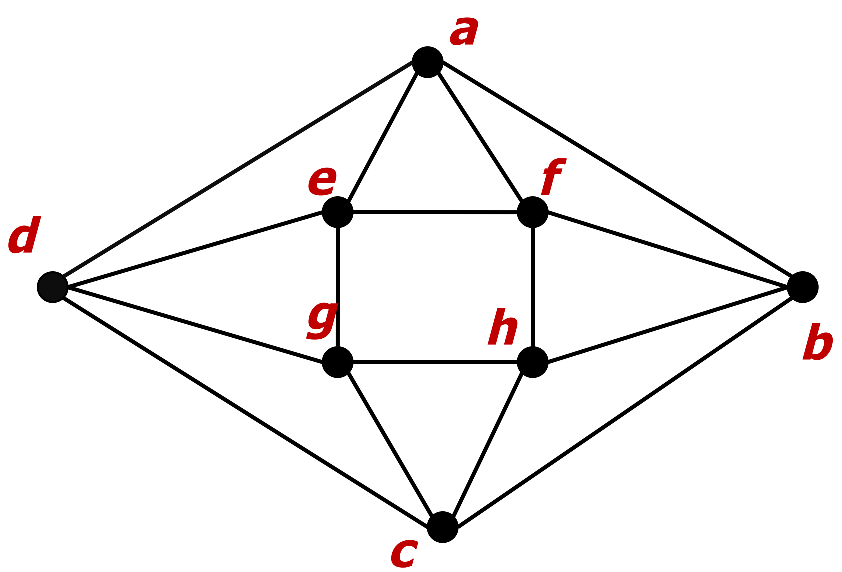
- Árvore → classe especial de grafo

- E para um grafo qualquer ?

  - Problemas: Falta de um referencial qual necessita de informação adicional para o processo de exploração.

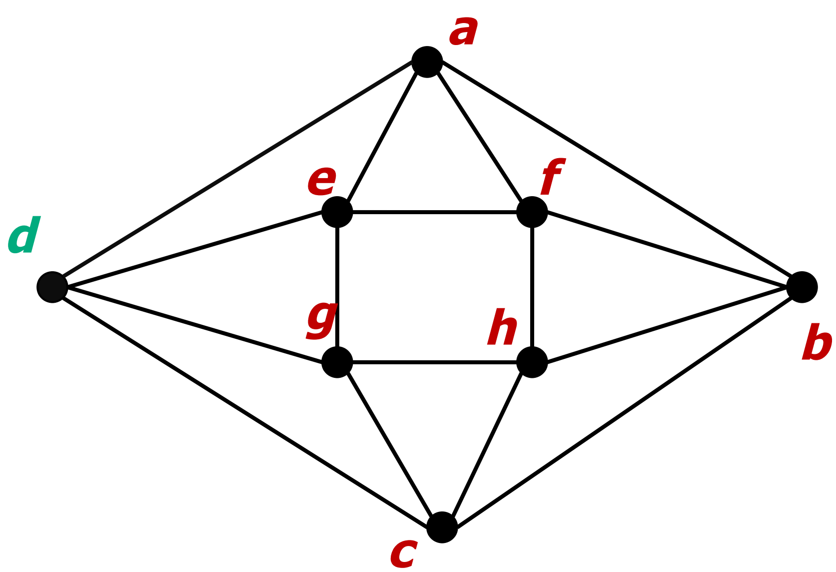
  - Marcas: Designa se o vértice foi ou não já considerado no processo de busca.

Algoritmo Básico(Ideia)



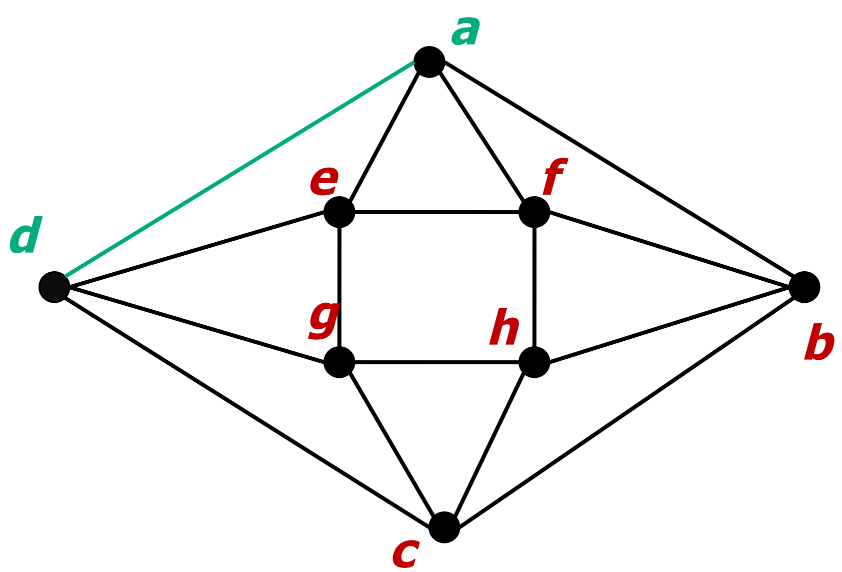
- Quando seleciona-se aresta *(v,w)* a partir do vértice marcado *v* dizemos que *(v,w)* foi **explorada** e *w* foi alcançado. Um vértice torna-se **explorado** quando todas arestas incidentes no mesmo tiverem sido exploradas.

Algoritmo Básico(Ideia)



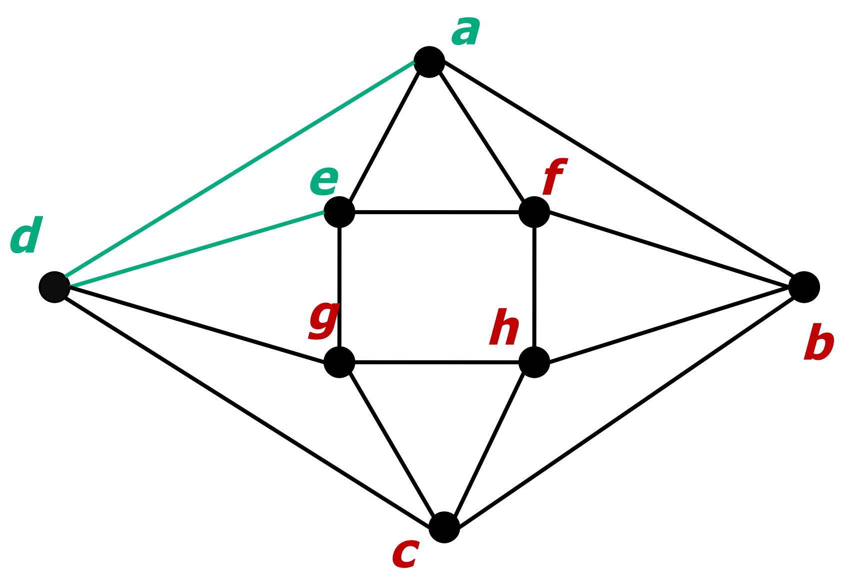
- Quando seleciona-se aresta *(v,w)* a partir do vértice marcado *v* dizemos que *(v,w)* foi **explorada** e *w* foi alcançado. Um vértice torna-se **explorado** quando todas arestas incidentes no mesmo tiverem sido exploradas.

Algoritmo Básico(Ideia)



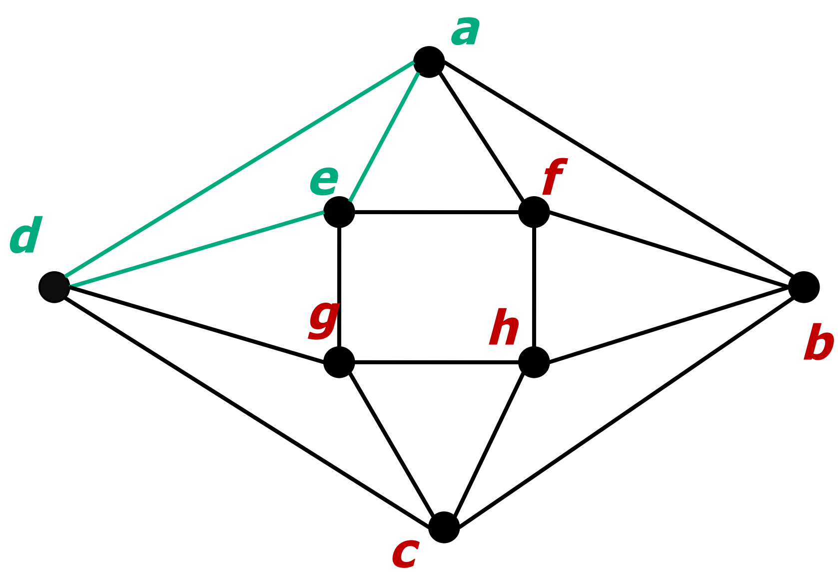
- Quando seleciona-se aresta *(v,w)* a partir do vértice marcado *v* dizemos que *(v,w)* foi **explorada** e *w* foi alcançado. Um vértice torna-se **explorado** quando todas arestas incidentes no mesmo tiverem sido exploradas.

Algoritmo Básico(Ideia)



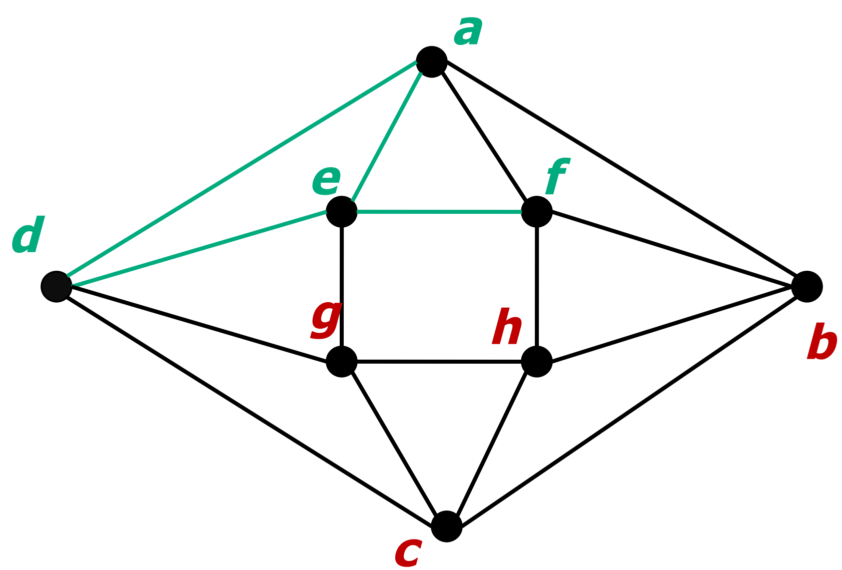
- Quando seleciona-se aresta *(v,w)* a partir do vértice marcado *v* dizemos que *(v,w)* foi **explorada** e *w* foi alcançado. Um vértice torna-se **explorado** quando todas arestas incidentes no mesmo tiverem sido exploradas.

Algoritmo Básico(Ideia)



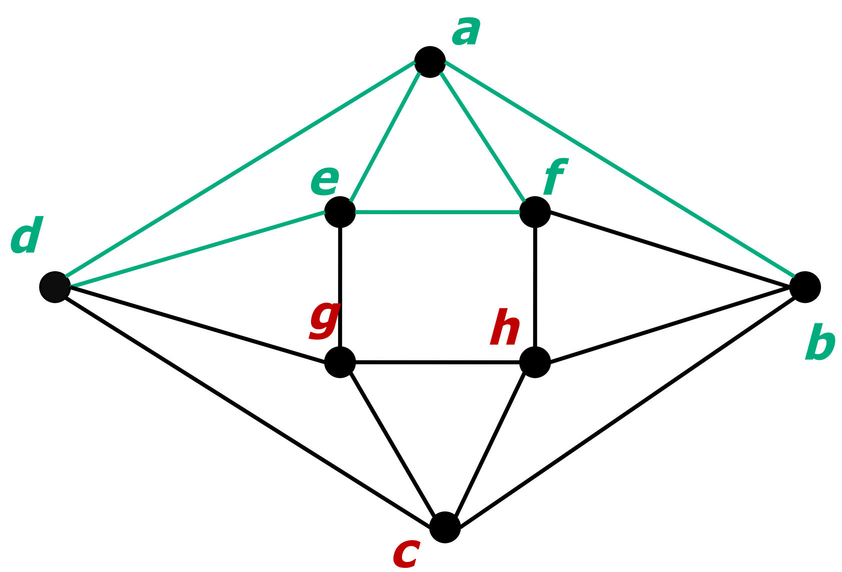
- Quando seleciona-se aresta *(v,w)* a partir do vértice marcado *v* dizemos que *(v,w)* foi **explorada** e *w* foi alcançado. Um vértice torna-se **explorado** quando todas arestas incidentes no mesmo tiverem sido exploradas.

Algoritmo Básico(Ideia)



- Quando seleciona-se aresta *(v,w)* a partir do vértice marcado *v* dizemos que *(v,w)* foi **explorada** e *w* foi alcançado. Um vértice torna-se **explorado** quando todas arestas incidentes no mesmo tiverem sido exploradas.

Algoritmo Básico(Ideia)



- Quando seleciona-se aresta *(v,w)* a partir do vértice marcado *v* dizemos que *(v,w)* foi **explorada** e *w* foi alcançado. Um vértice torna-se **explorado** quando todas arestas incidentes no mesmo tiverem sido exploradas.

Busca Geral

```
algoritmo BuscaGeral(G);
início
   escolher e marcar um vértice inicial;
   enquanto existir algum vértice v marcado e incidente a
            uma aresta (v,w) não explorada faça
              início
                 escolher o vértice v e explorar (v,w);
                 se w é não marcado então marcar w;
              fim
fim
```

Busca Geral

```
algoritmo BuscaGeral(G);
início
   escolher e marcar um vértice inicial;
   enquanto existir algum vértice v marcado e incidente a
            uma aresta (v,w) não explorada faça
              início
                 escolher o vértice v e explorar (v,w);
                 se w é não marcado então marcar w;
              fim
fim
```

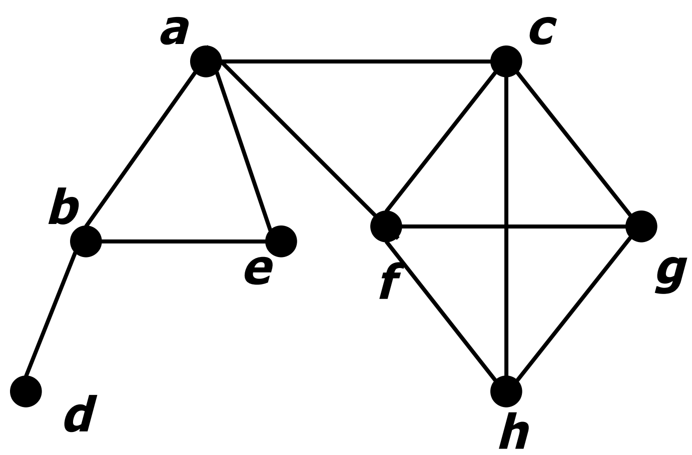
- Esta busca fornece um só resultado?

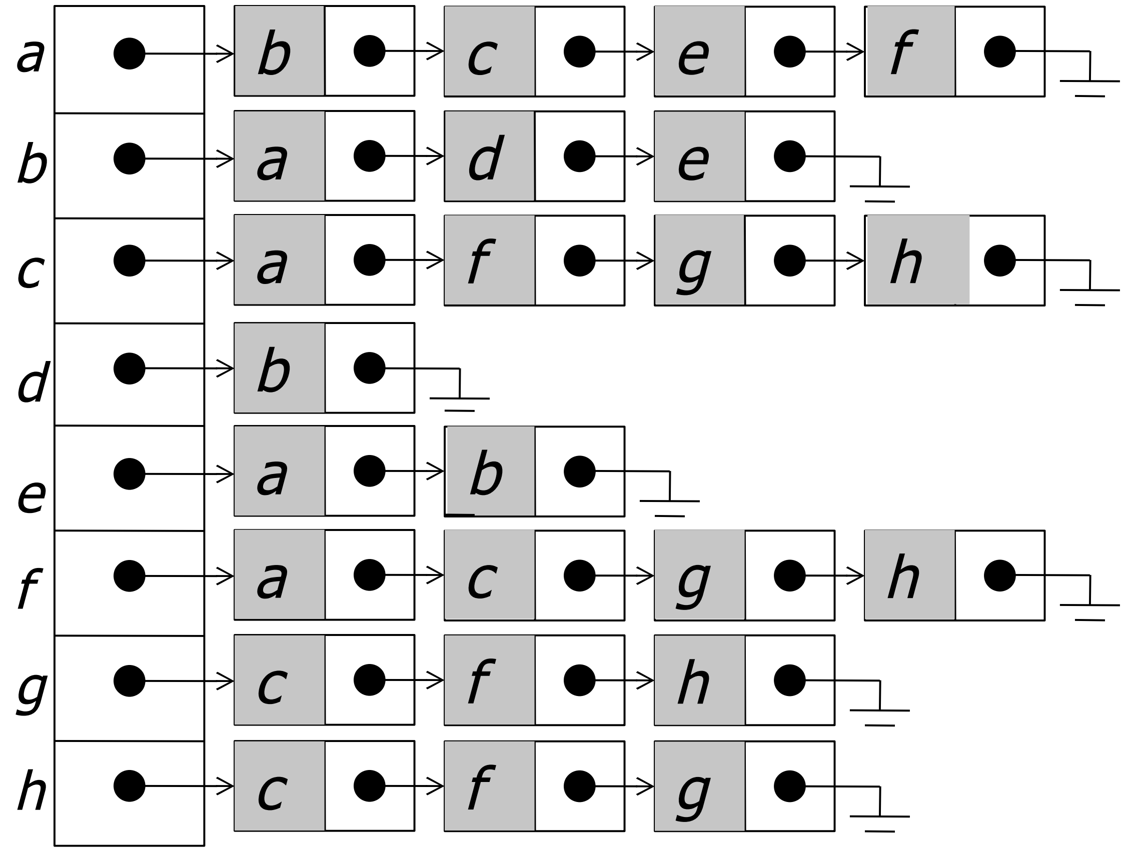
Busca em Profundidade

Critério: "dentre os vértices marcados incidentes a alguma aresta não explorada, escolher aquele mais **recentemente** alcançado na busca.".

Busca em Profundidade

Ex.:



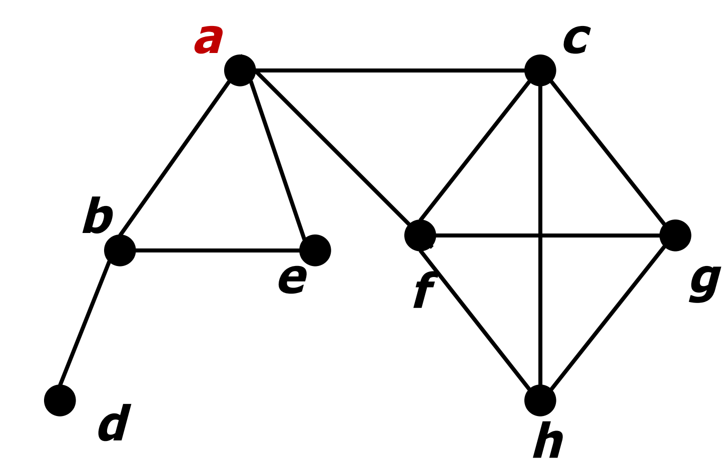


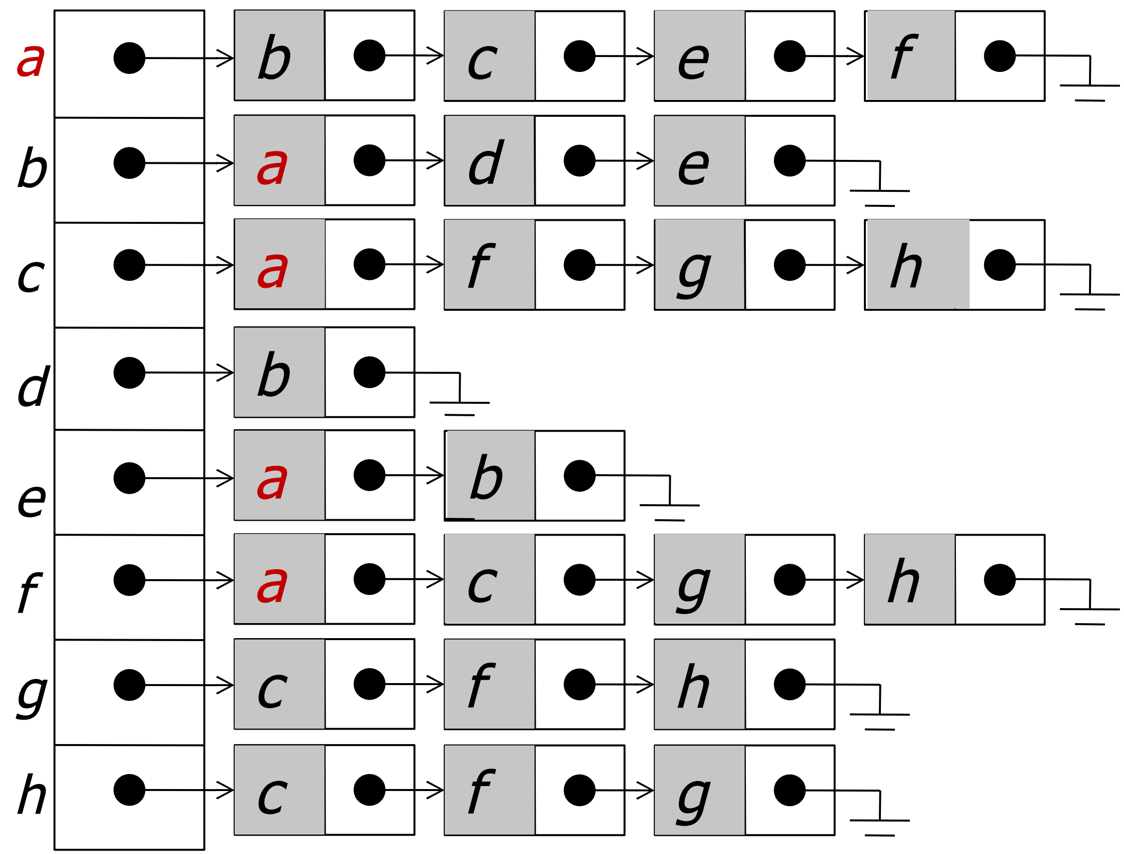
aresta da árvore

aresta de retorno

Busca em Profundidade

Ex.:



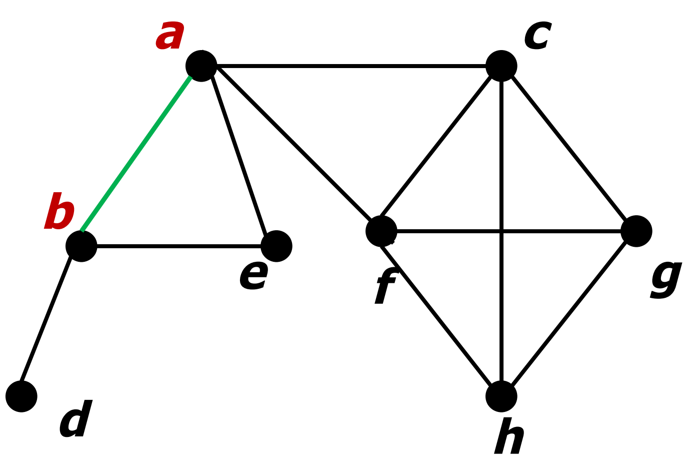


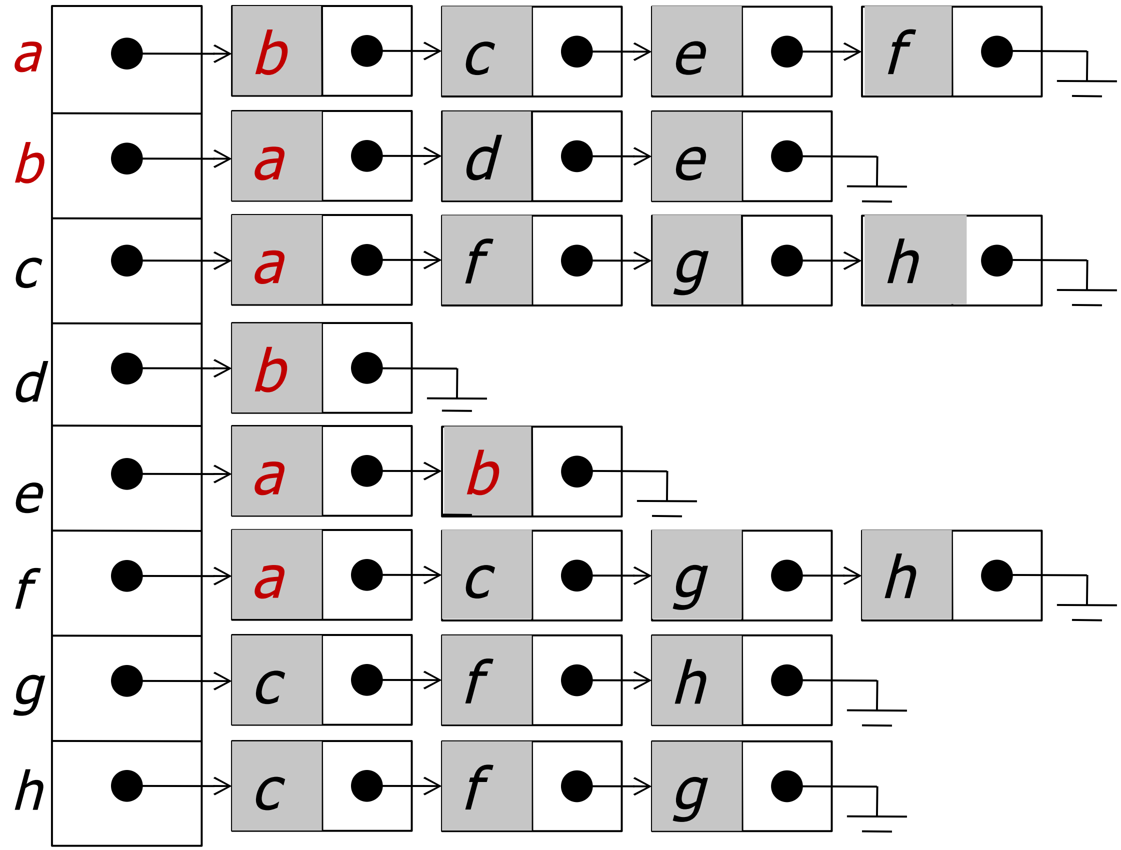
aresta da árvore

aresta de retorno

Busca em Profundidade

Ex.:



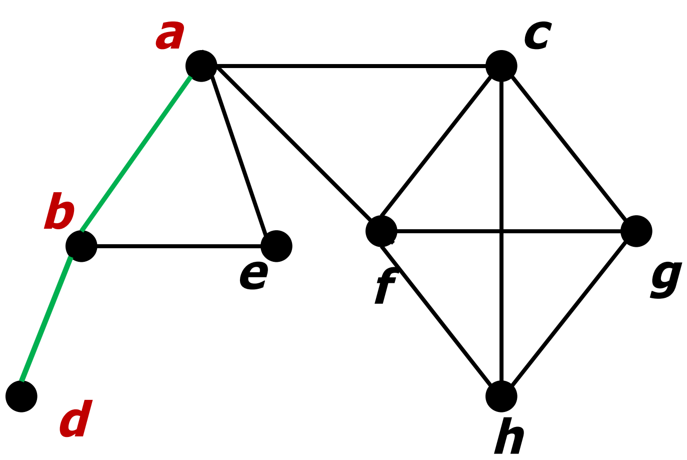


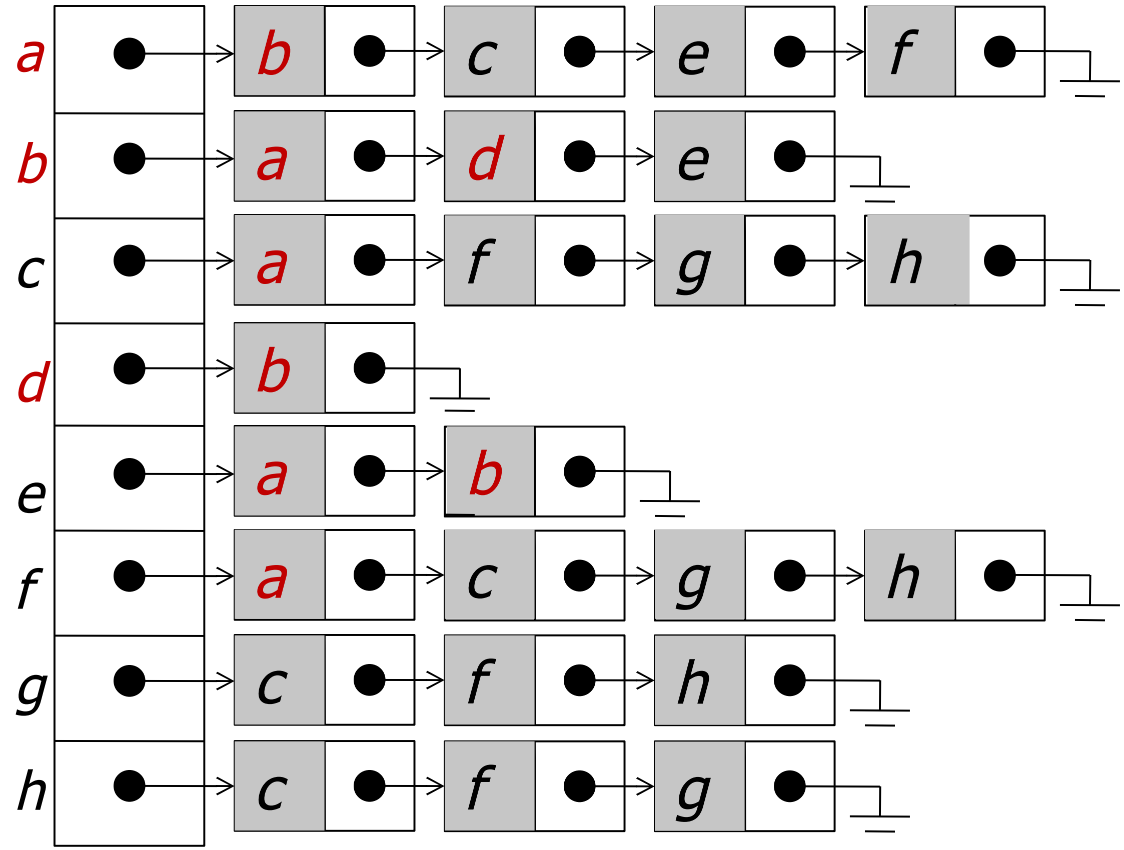
aresta da árvore

aresta de retorno

Busca em Profundidade

Ex.:



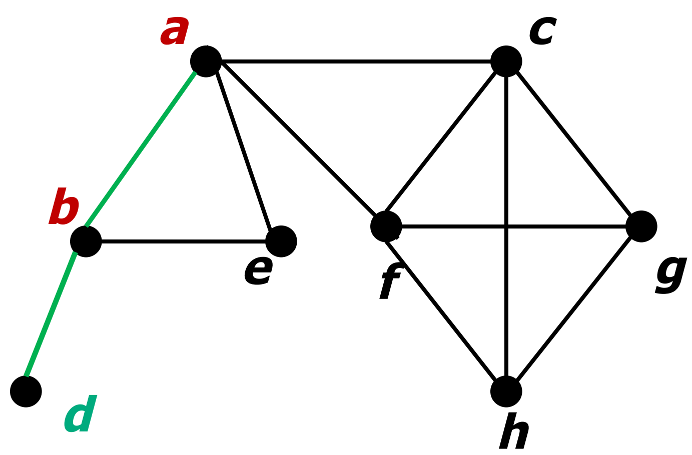


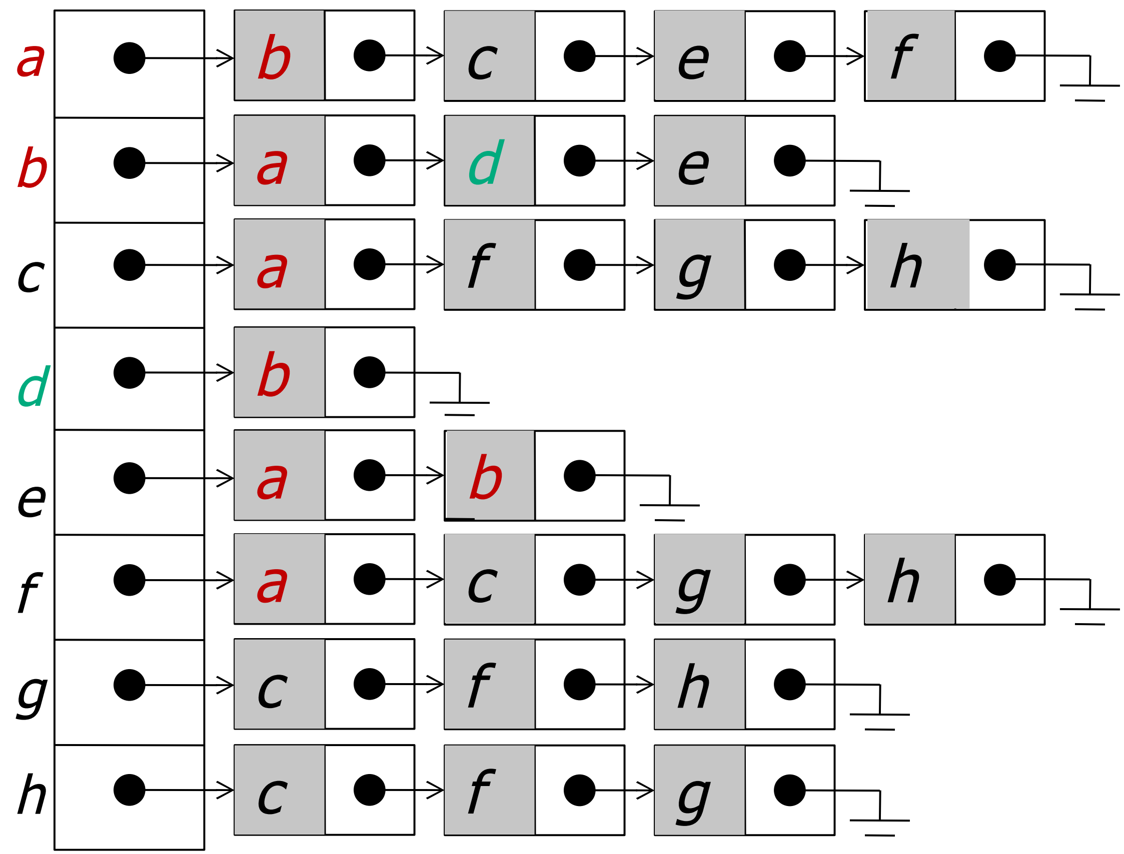
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:





aresta da árvore

aresta de retorno

Busca em Profundidade

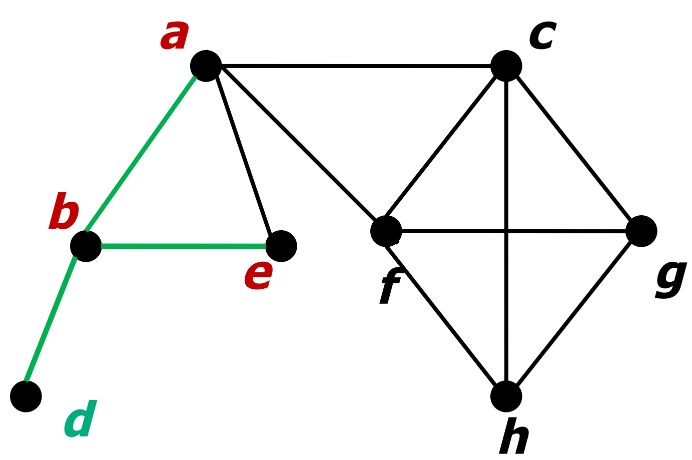
A cada aresta explorada um novo vértice ainda não visitado é marcado.

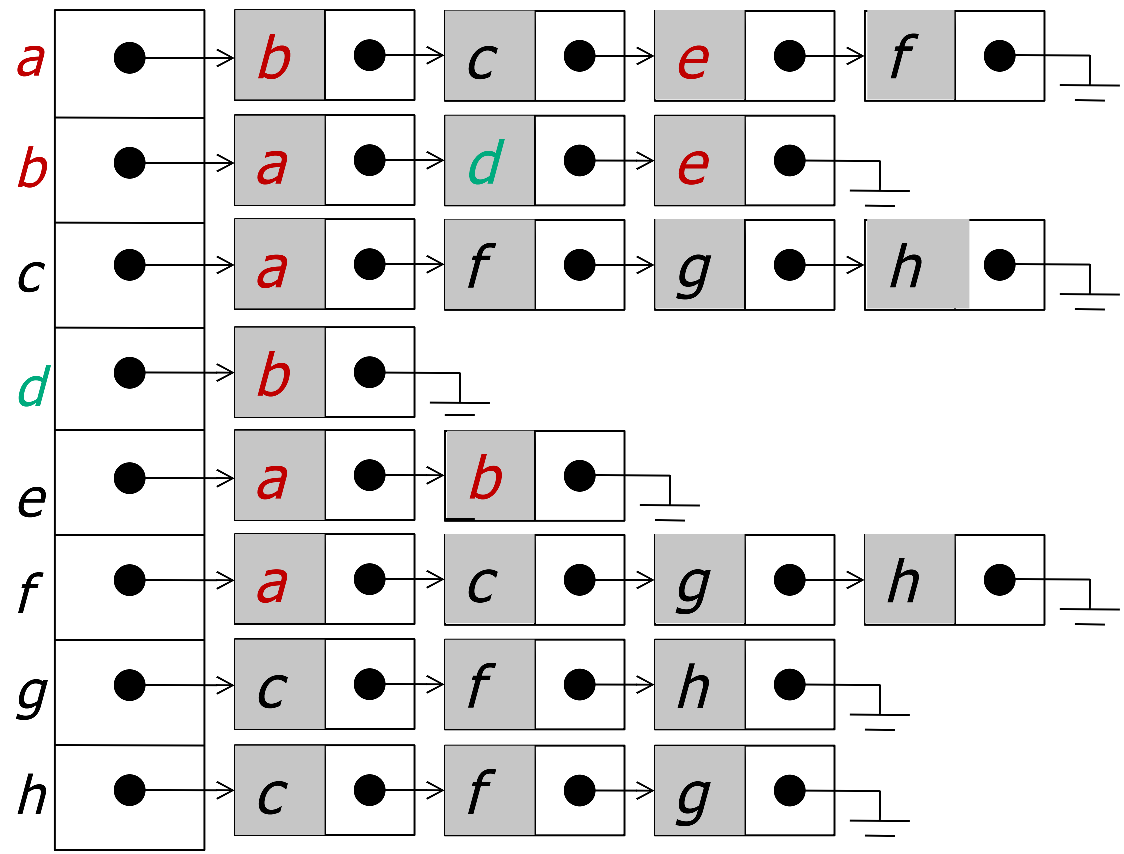
A busca constrói um árvore que chamamos uma árvore de profundidade.

As arestas dessa árvore são chamadas de arestas da árvore.

Busca em Profundidade

Ex.:





aresta da árvore

aresta de retorno

Busca em Profundidade

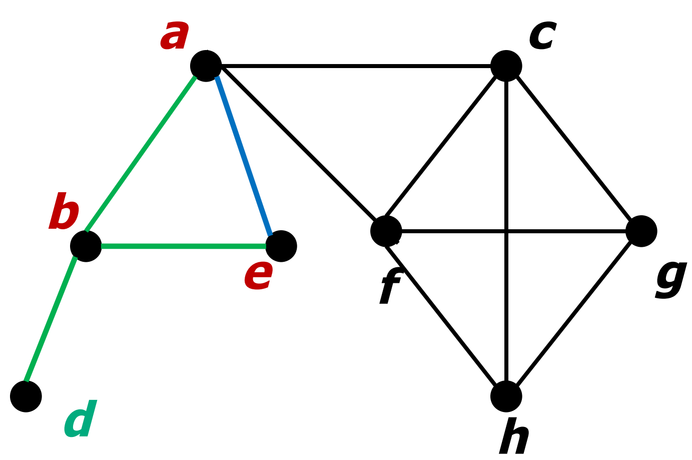
O que acontece quando encontramos um vértice já visitado?

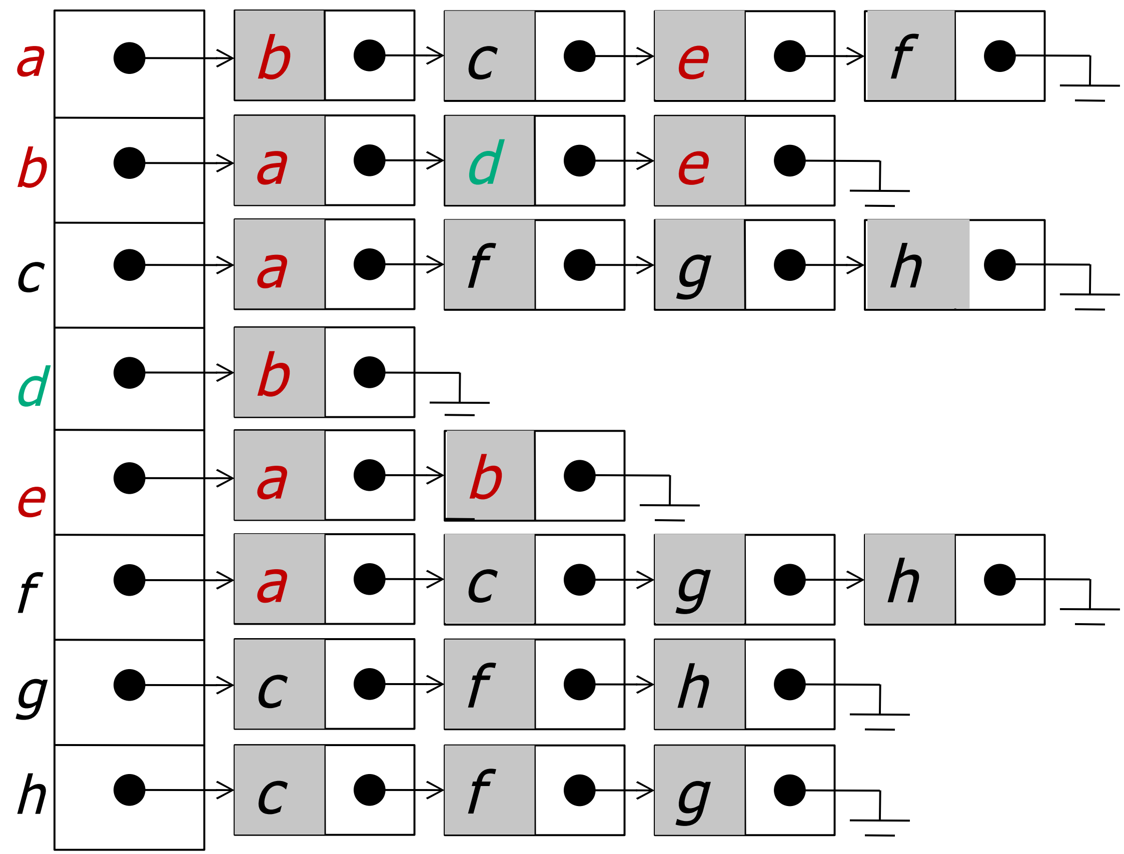
O vértice já é marcado, logo a busca não visita ele novamente.

Essas arestas são chamadas de arestas de retorno.

Busca em Profundidade

Ex.:



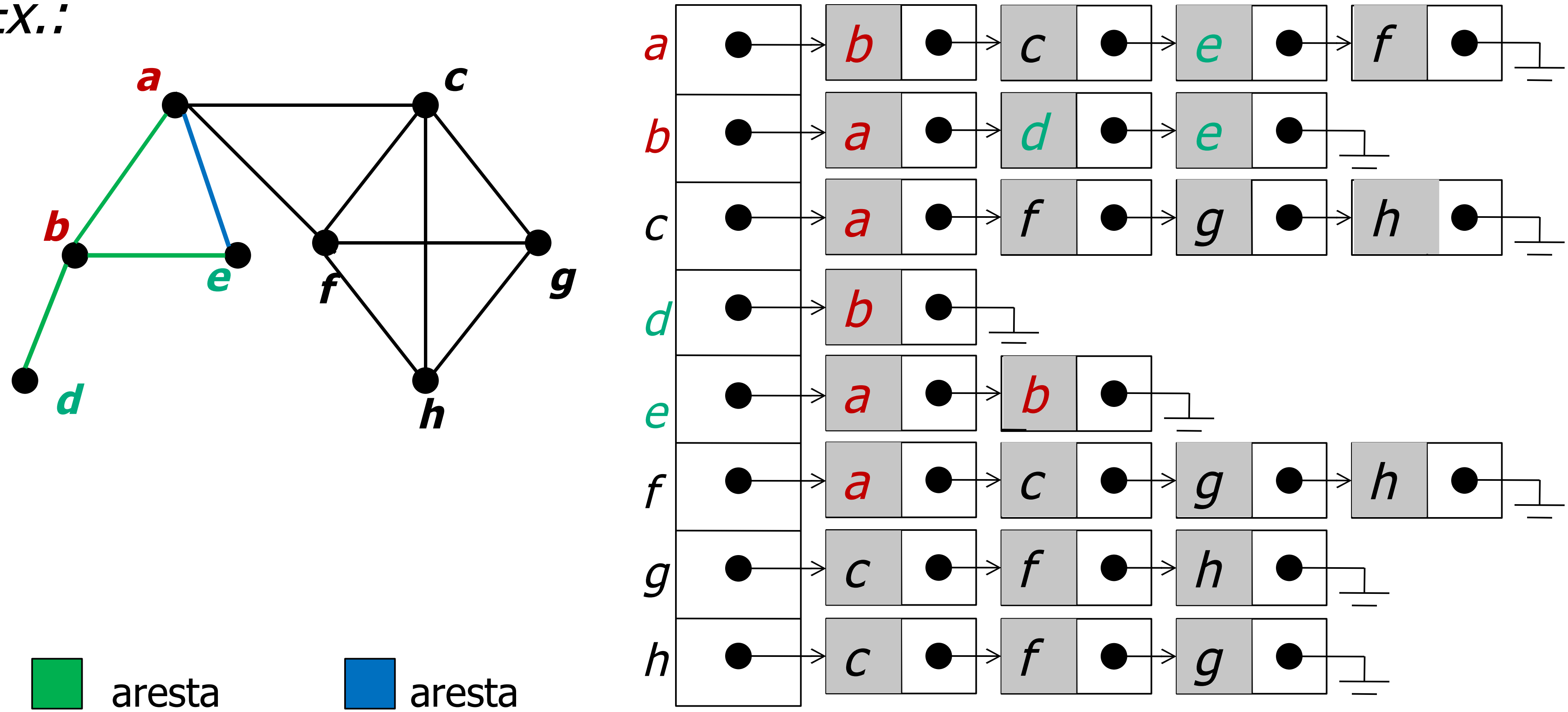


aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:

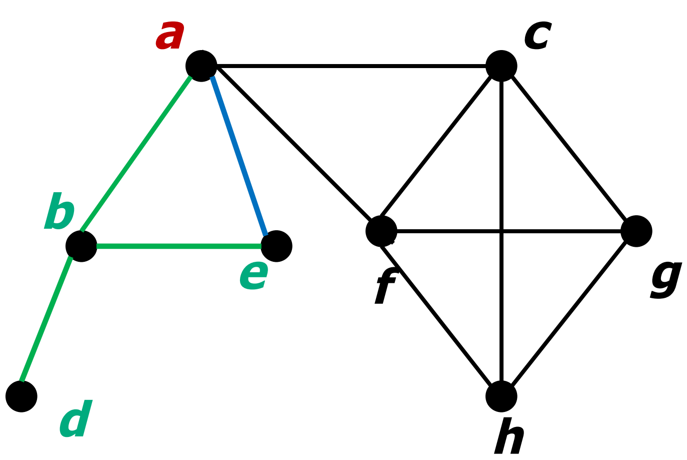


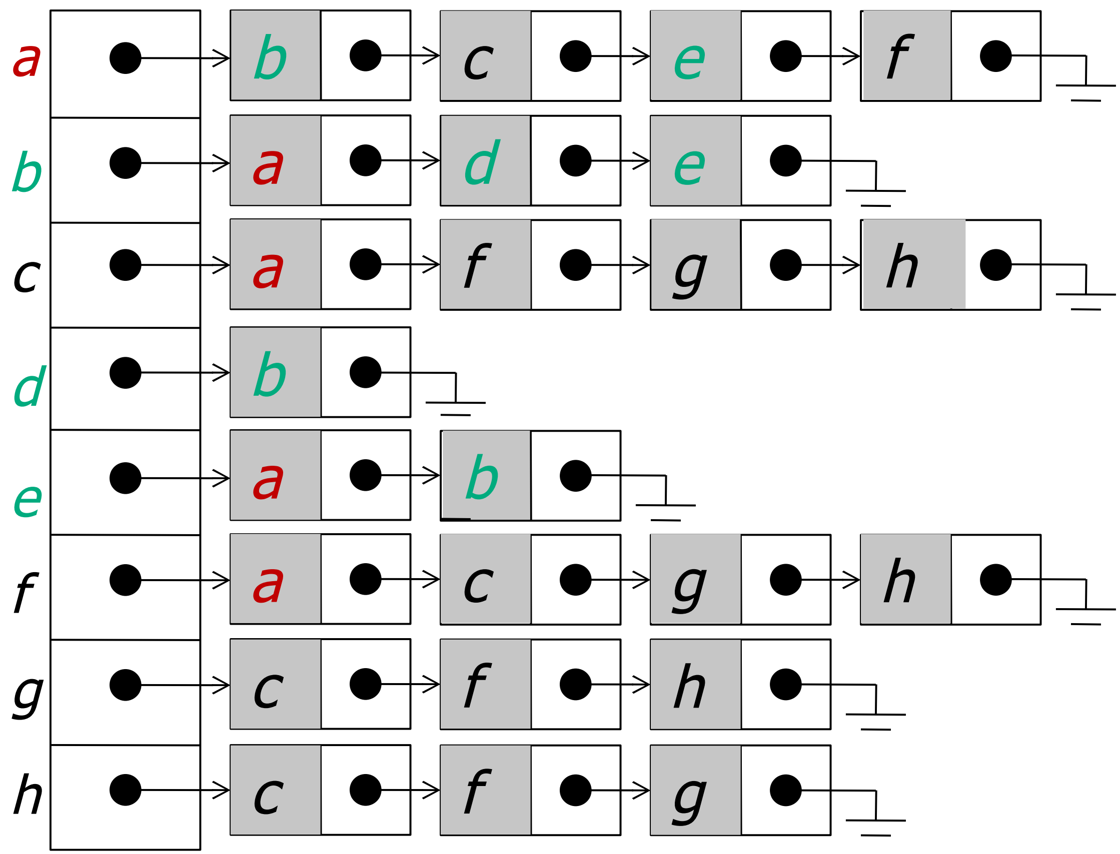
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:



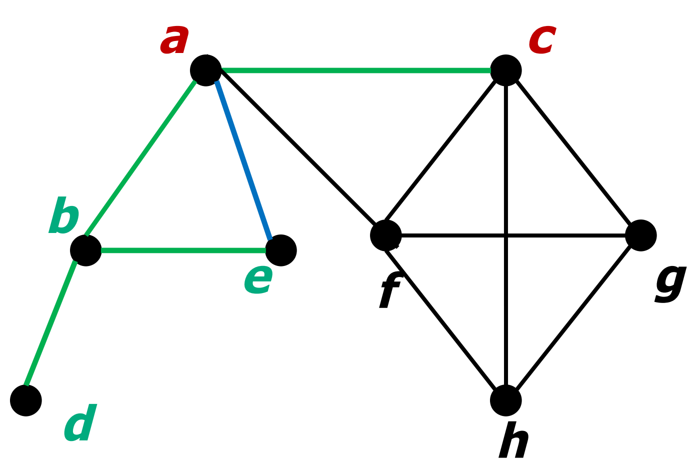


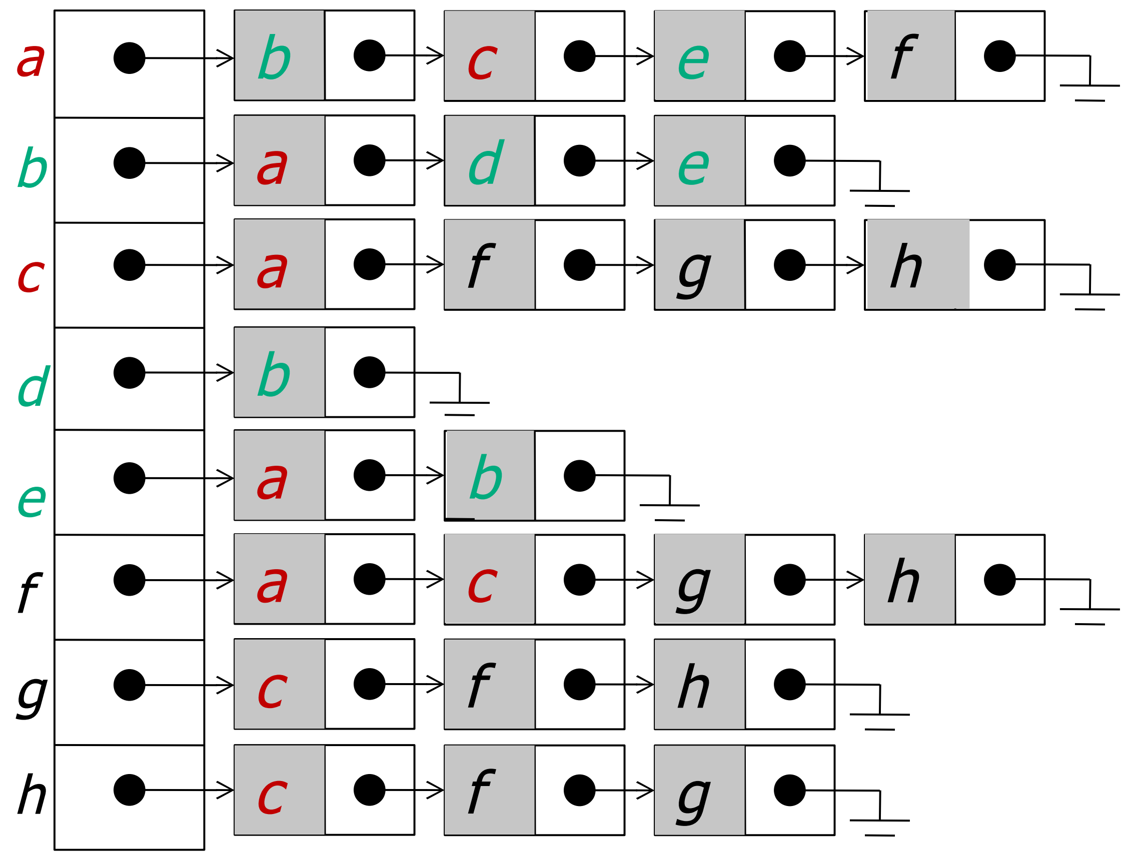
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:



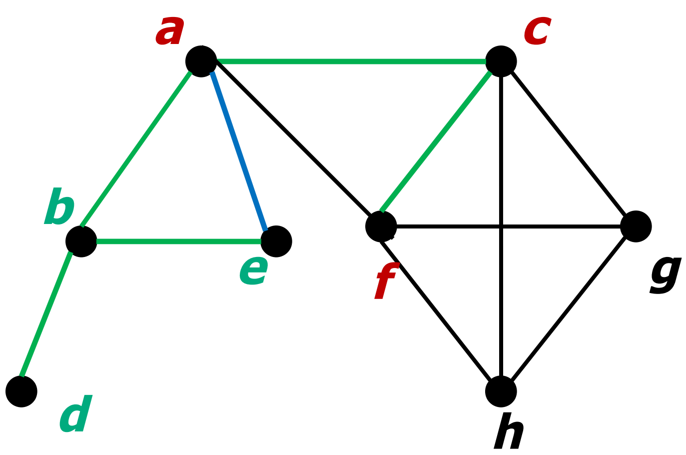


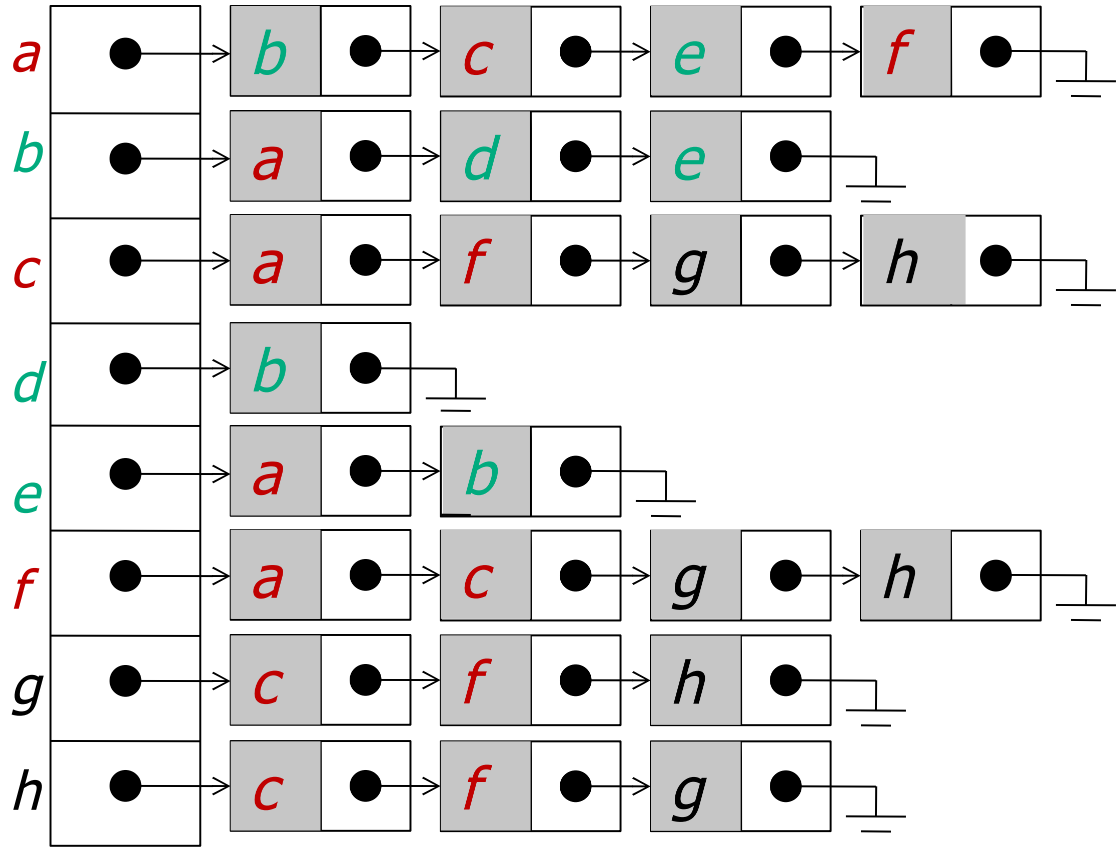
aresta da árvore

aresta de retorno

Busca em Profundidade

Ex.:



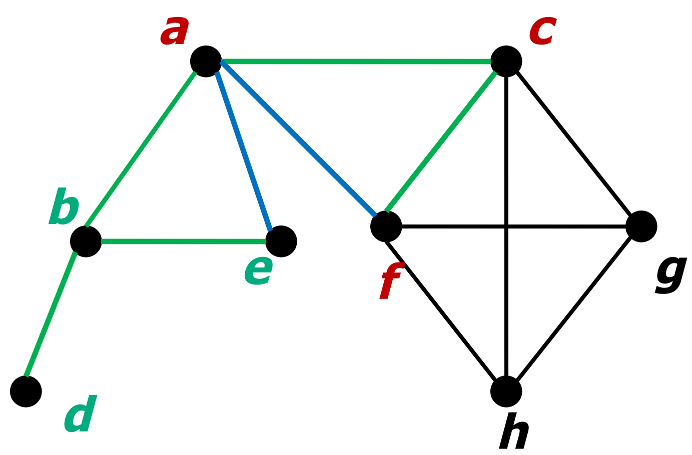


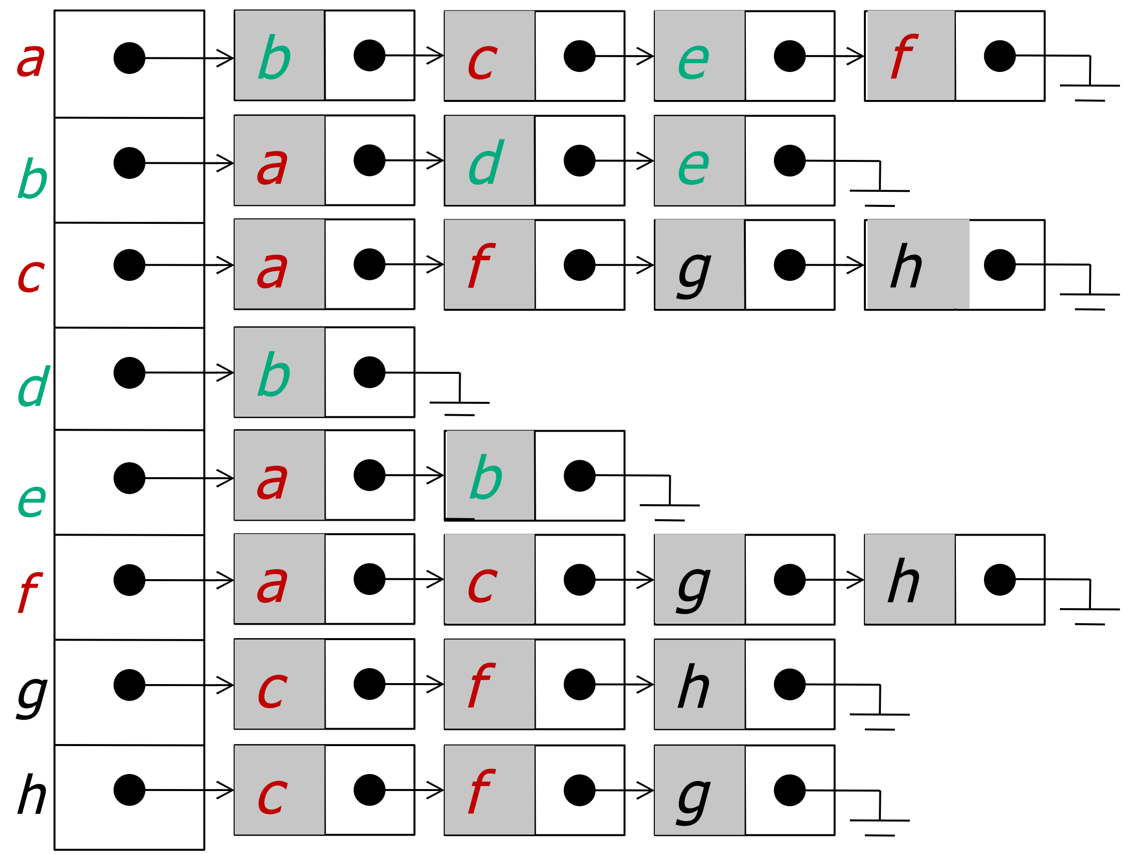
aresta da árvore

aresta de retorno

Busca em Profundidade

Ex.:



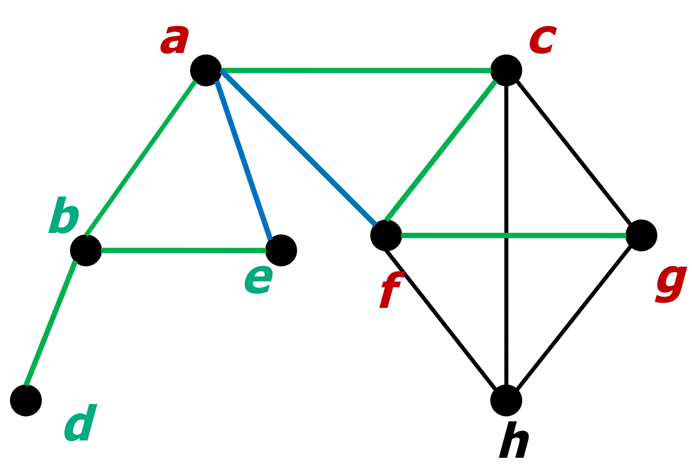


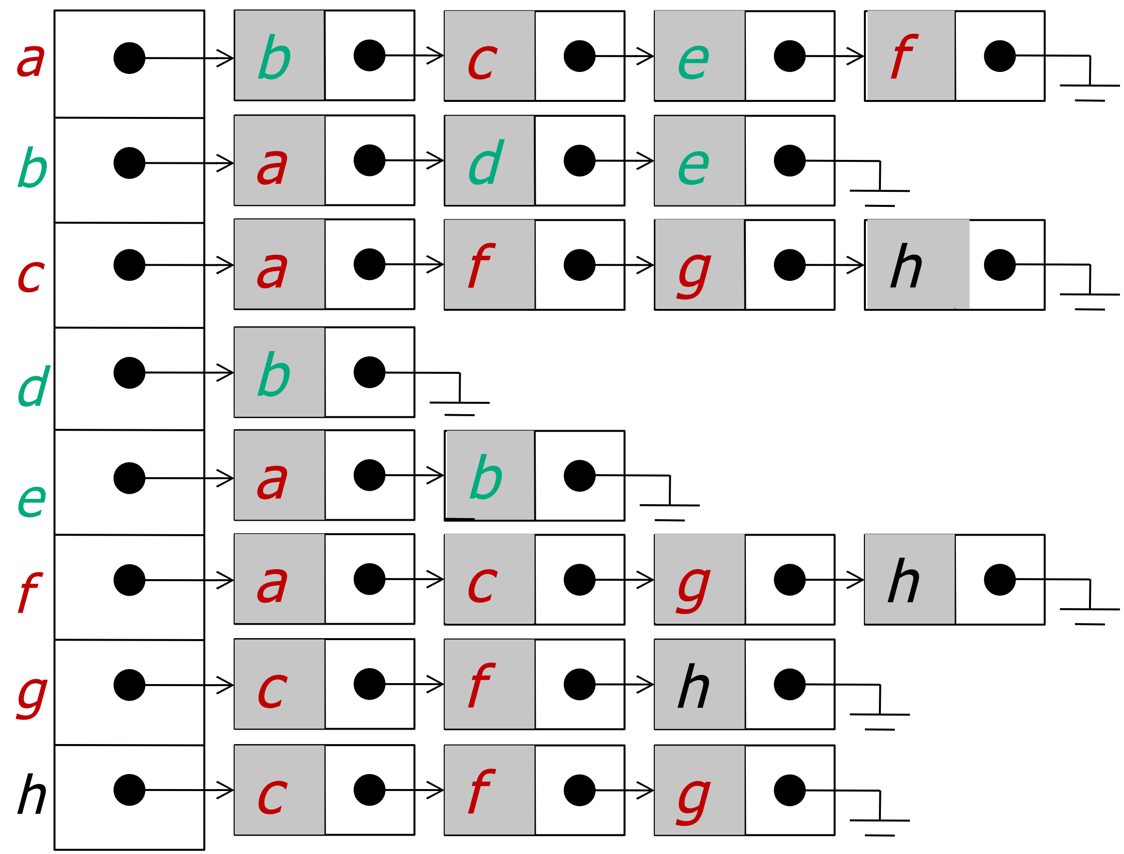
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:



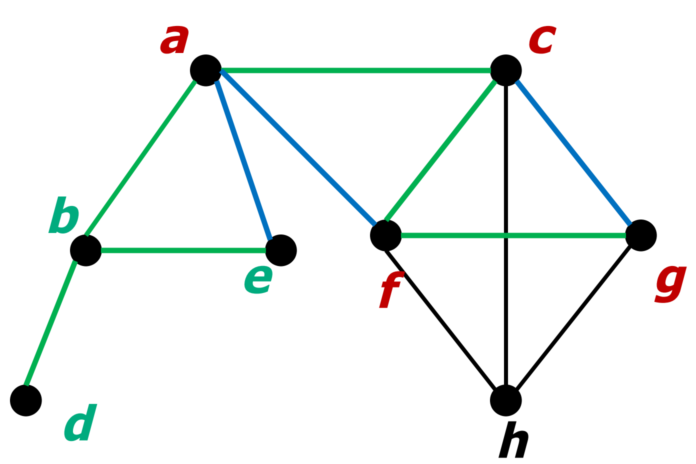


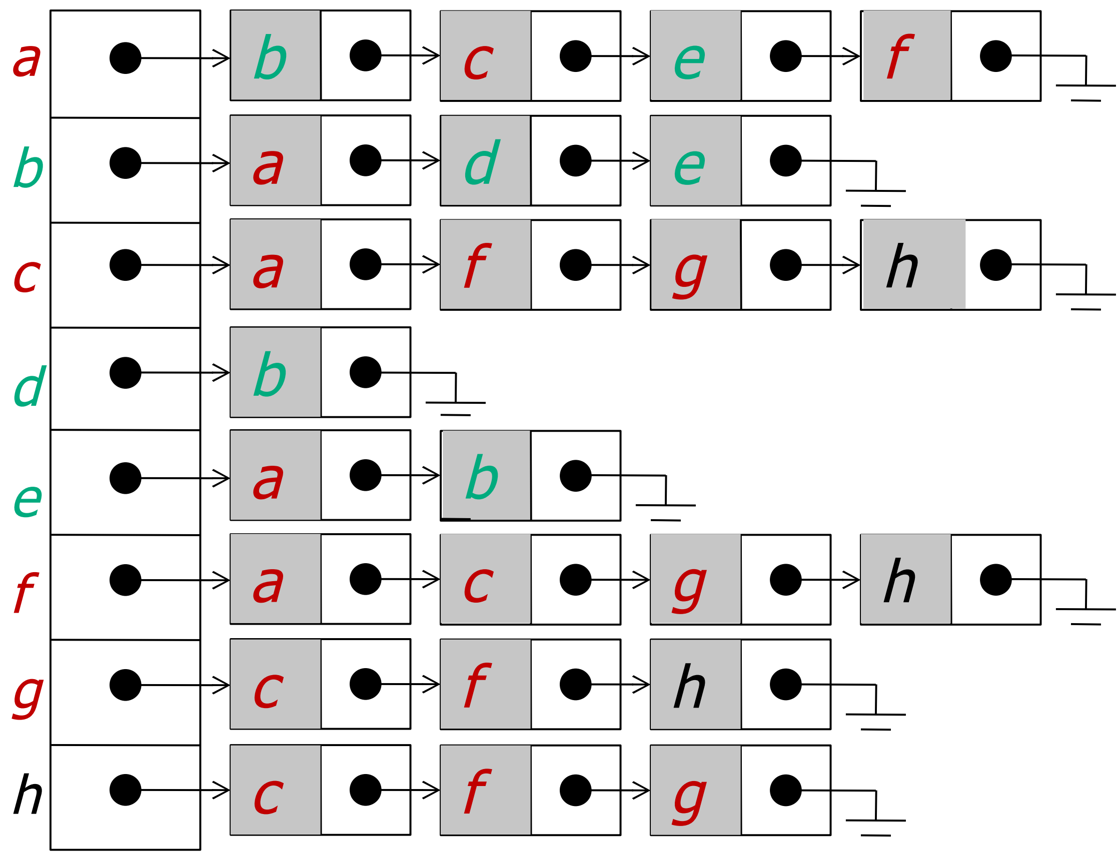
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:



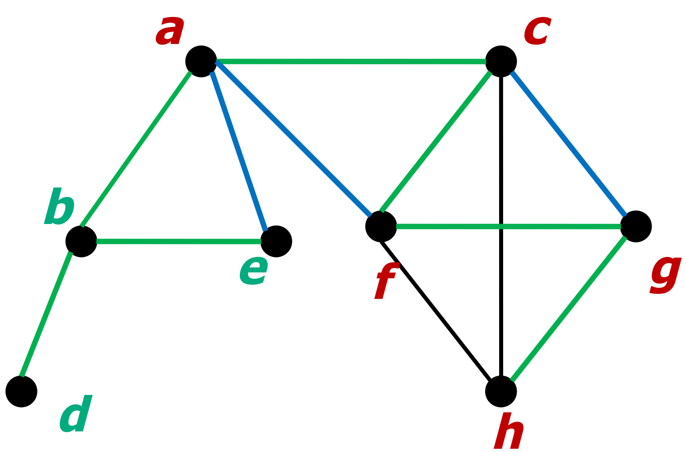


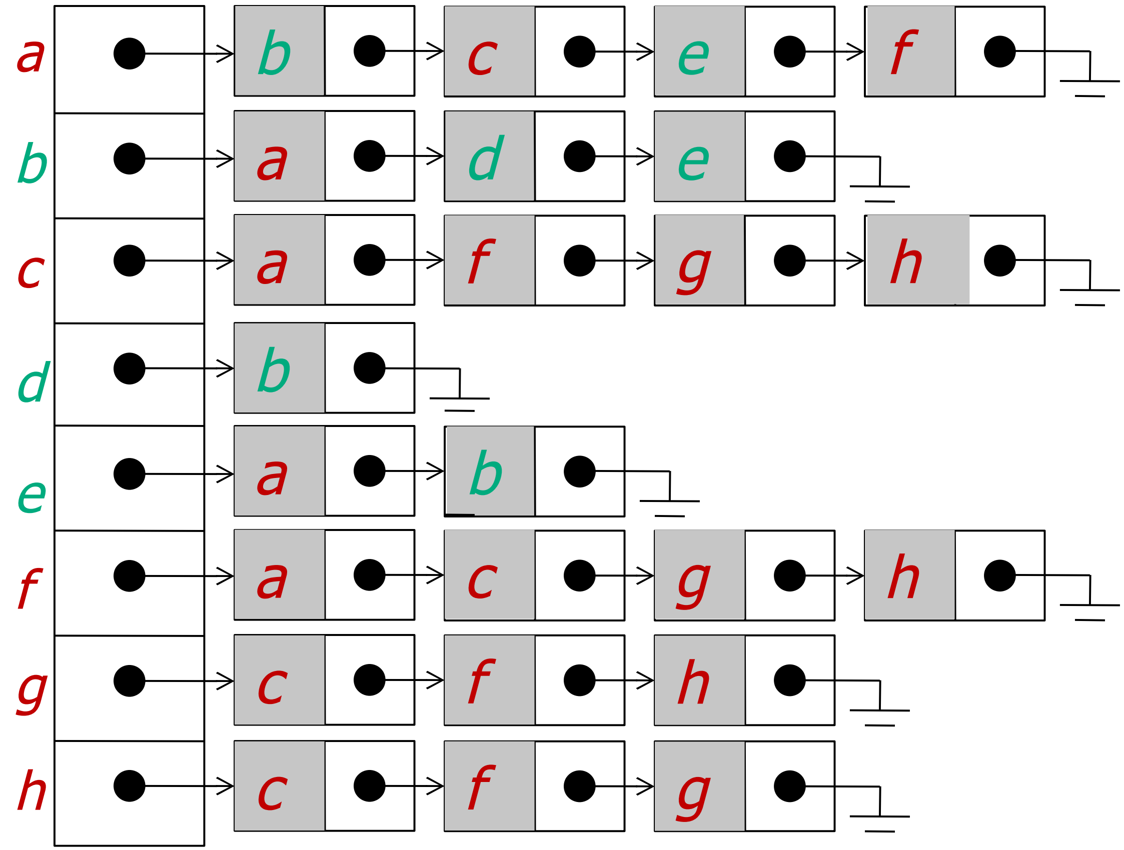
aresta da árvore

aresta de retorno

Busca em Profundidade

Ex.:



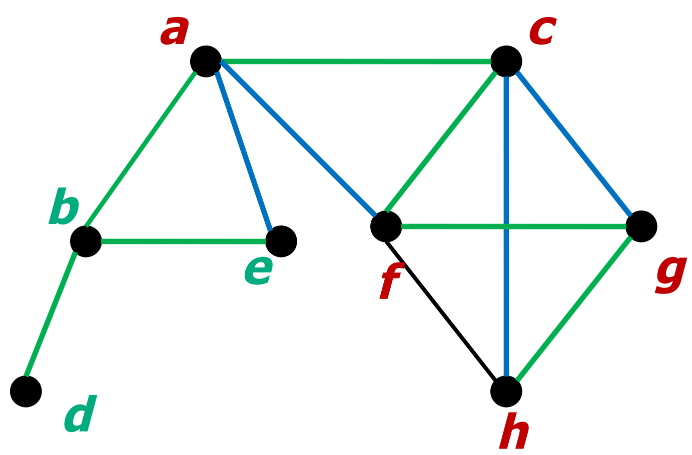


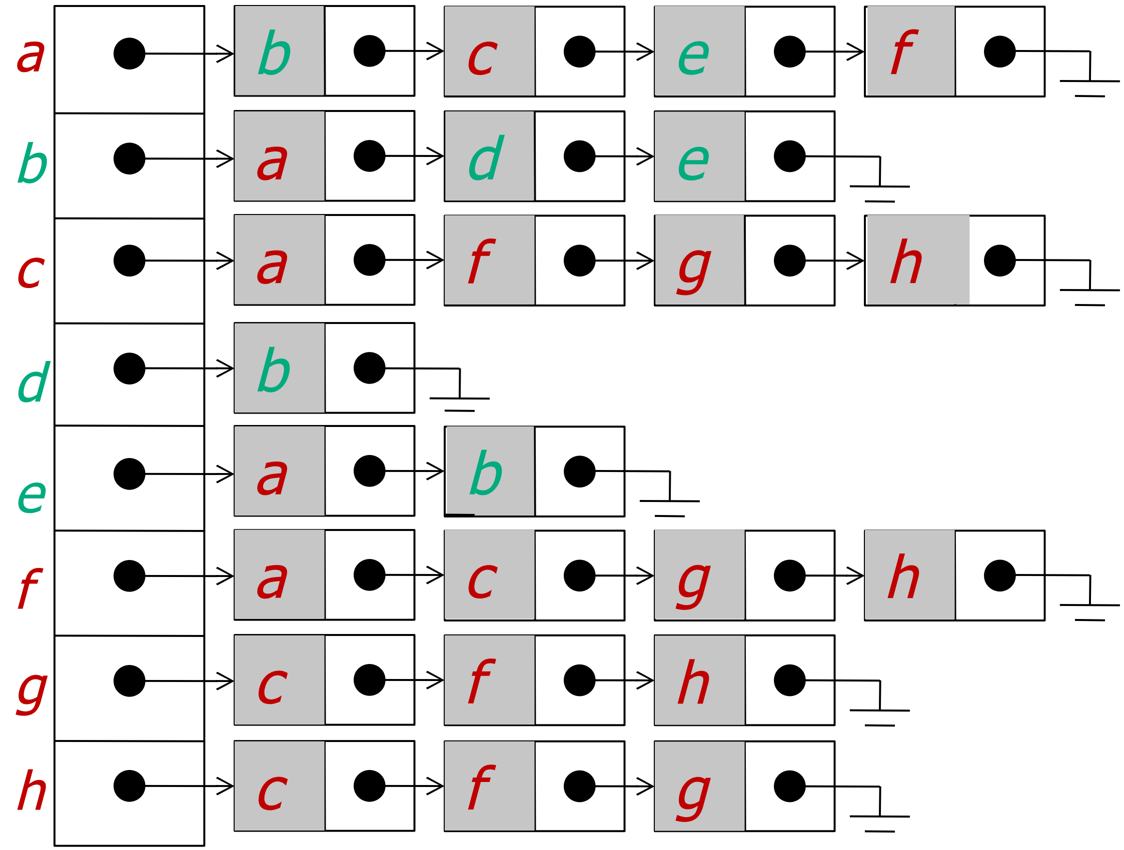
aresta da árvore

aresta de retorno

Busca em Profundidade

Ex.:



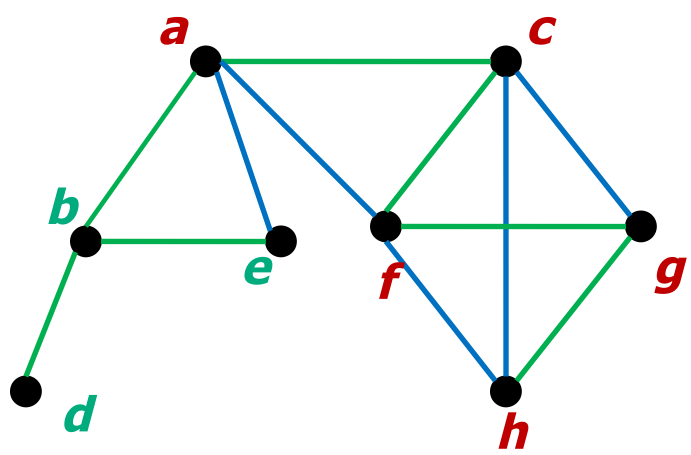


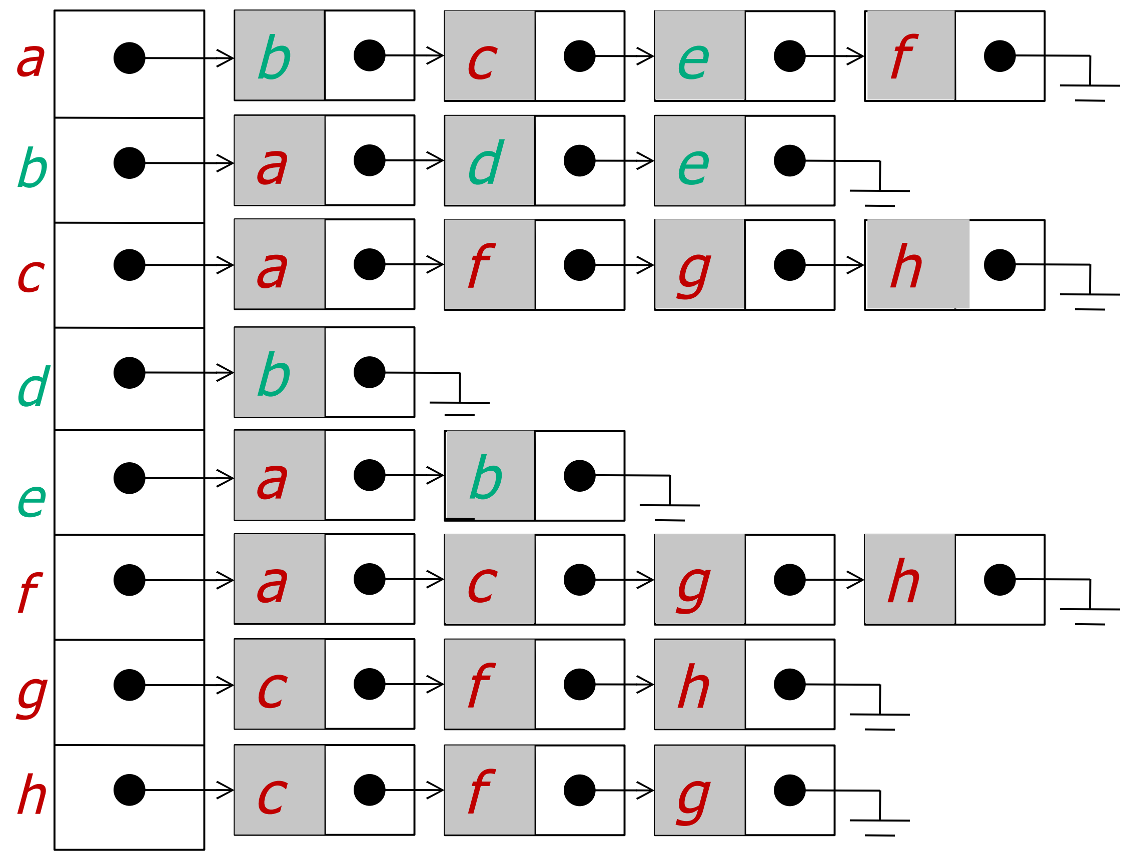
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:



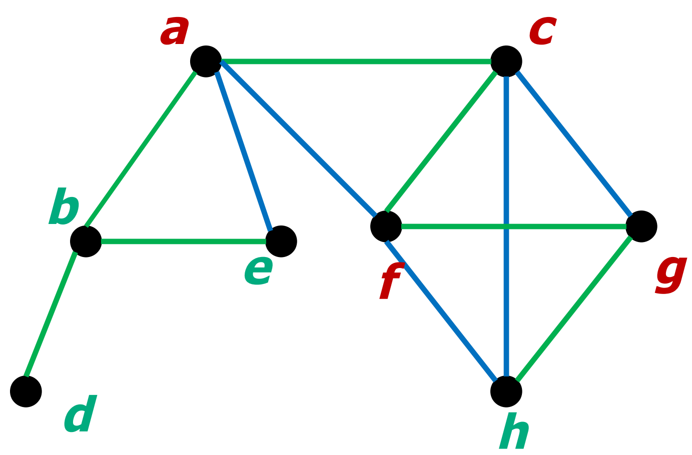


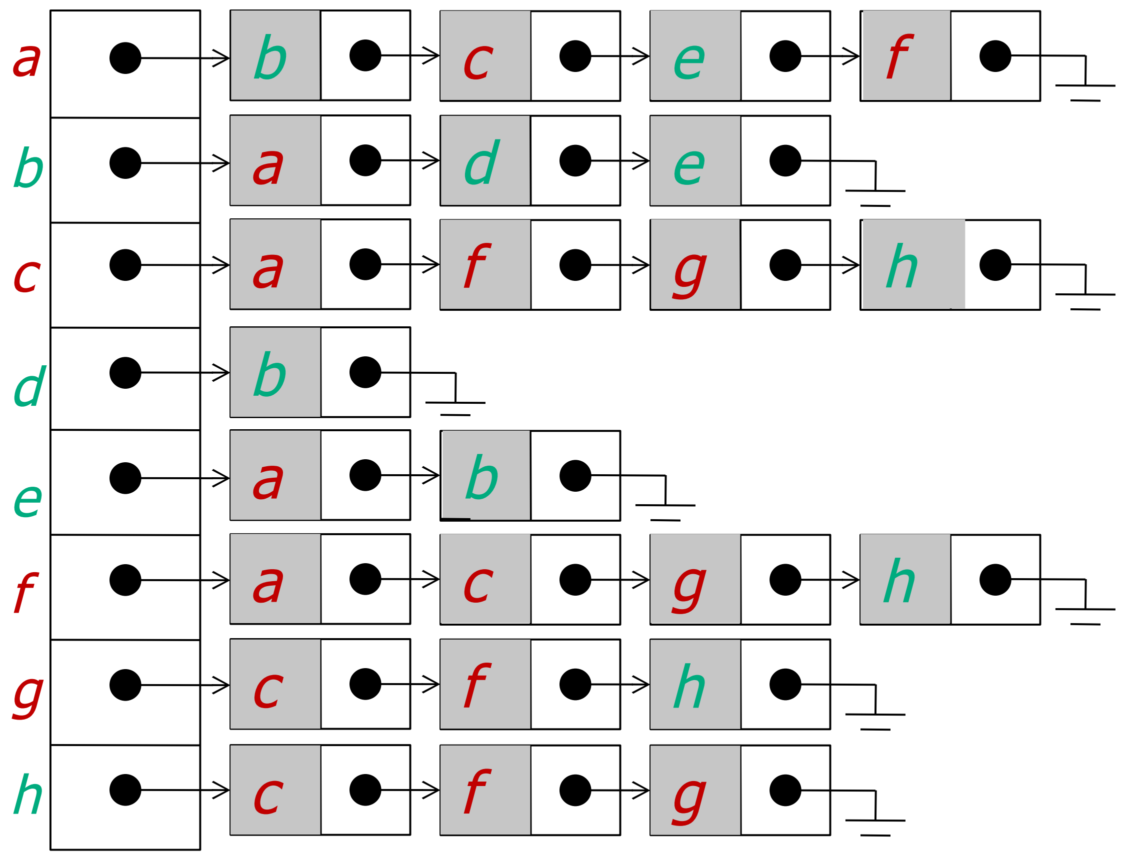
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:



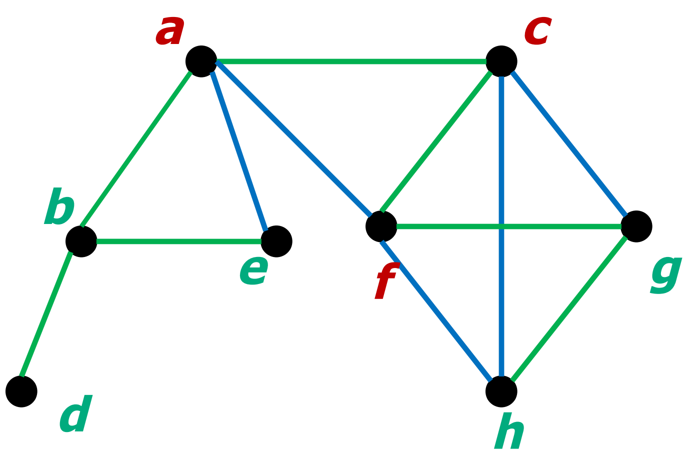


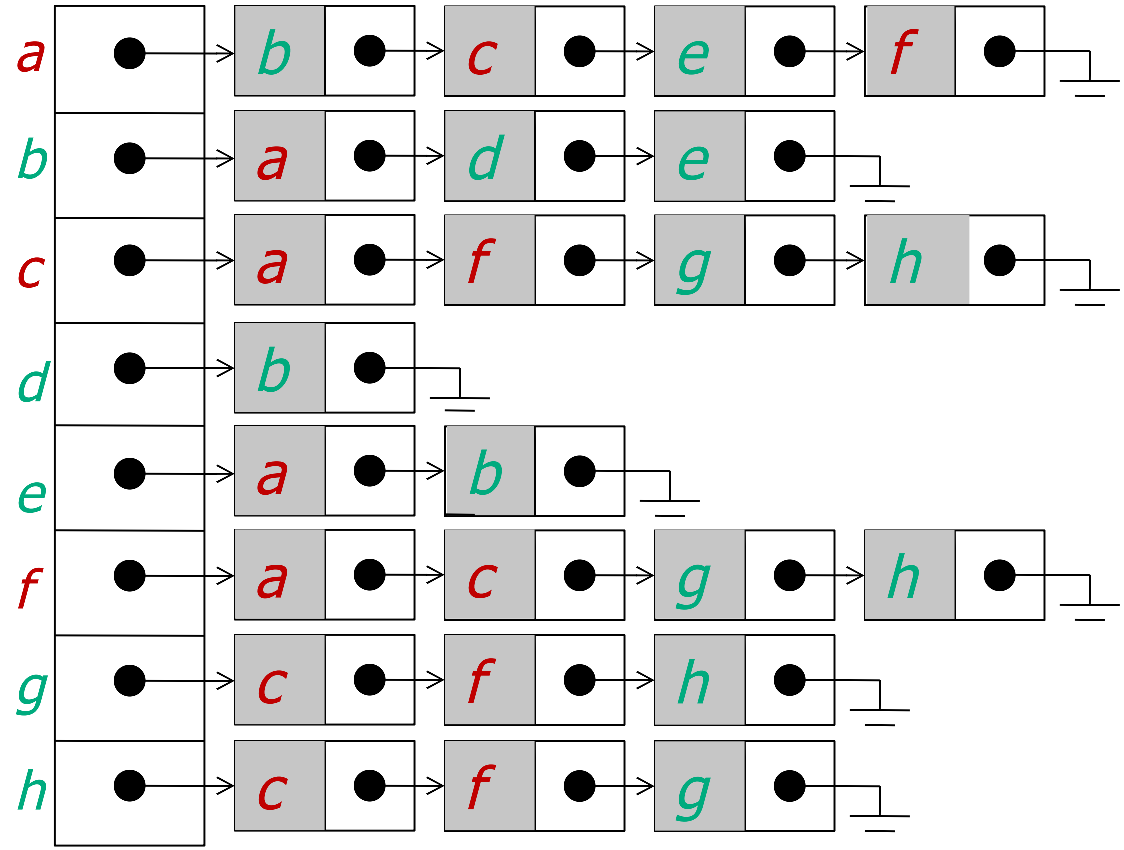
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:



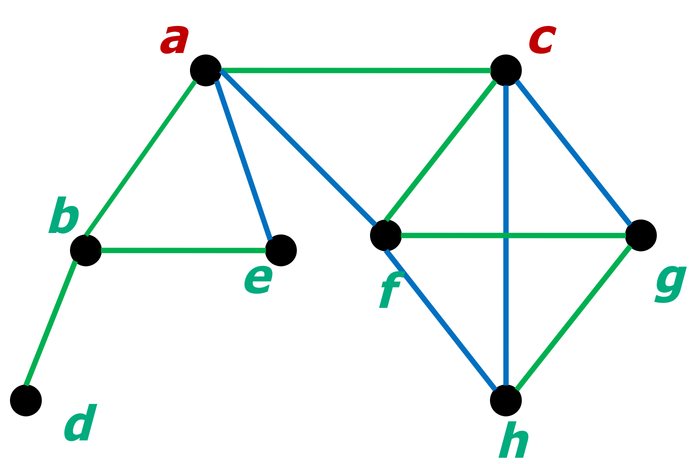


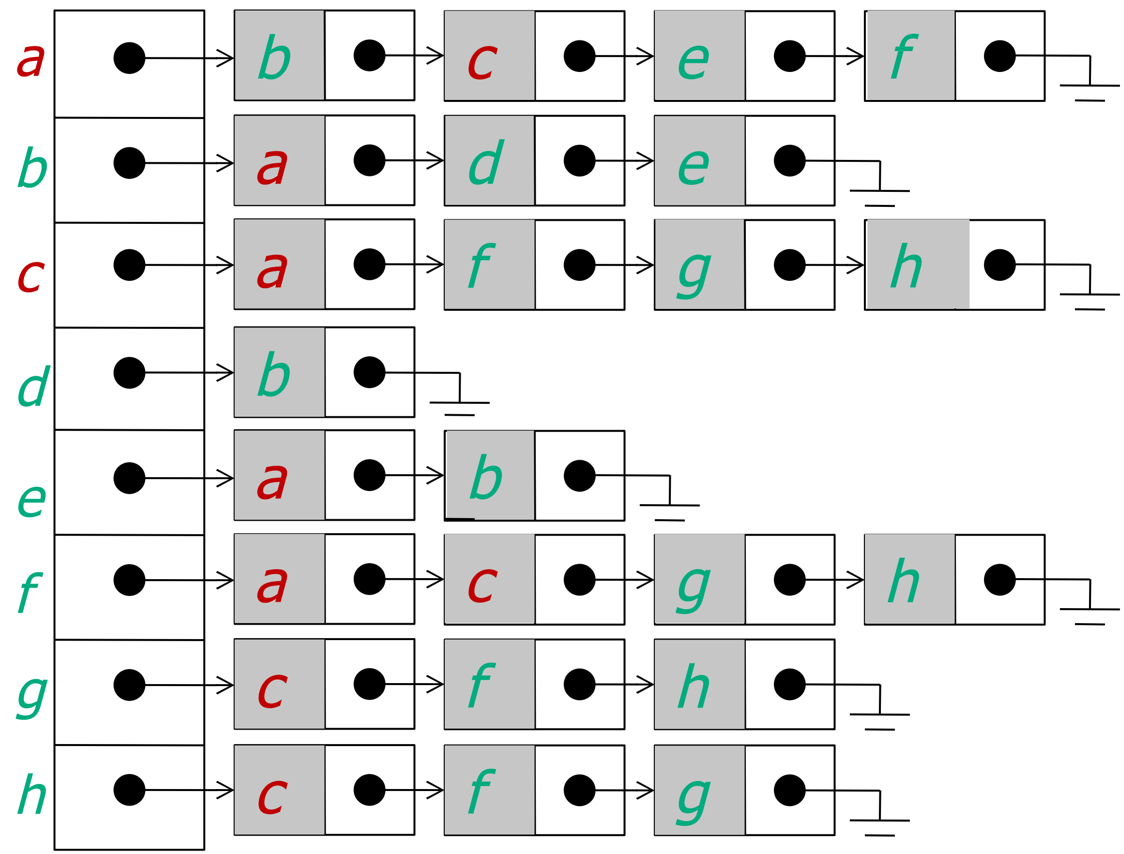
aresta da árvore

aresta de retorno

Busca em Profundidade

Ex.:



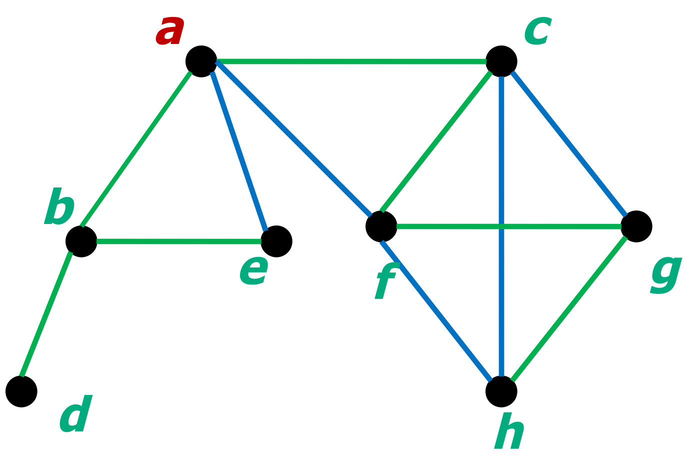


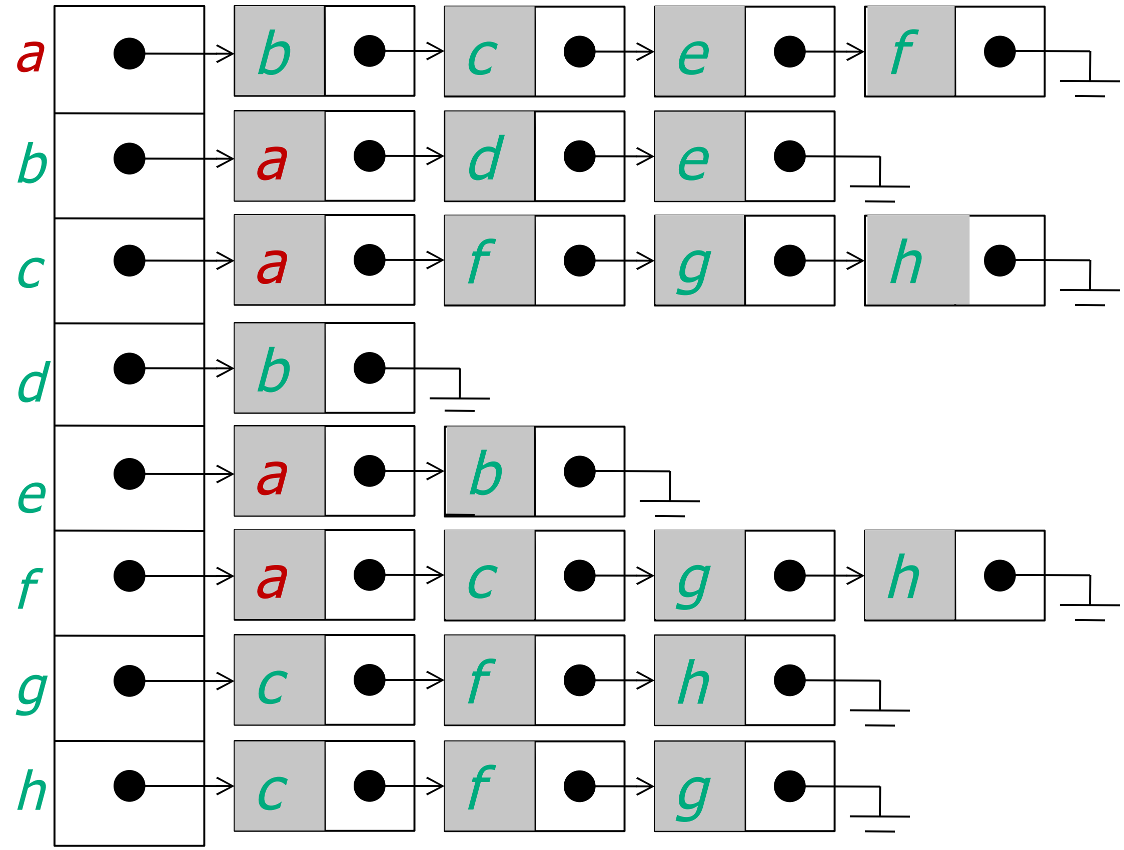
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:



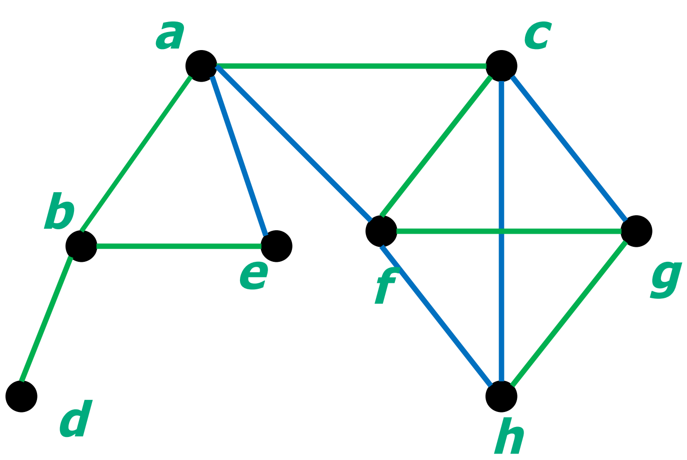


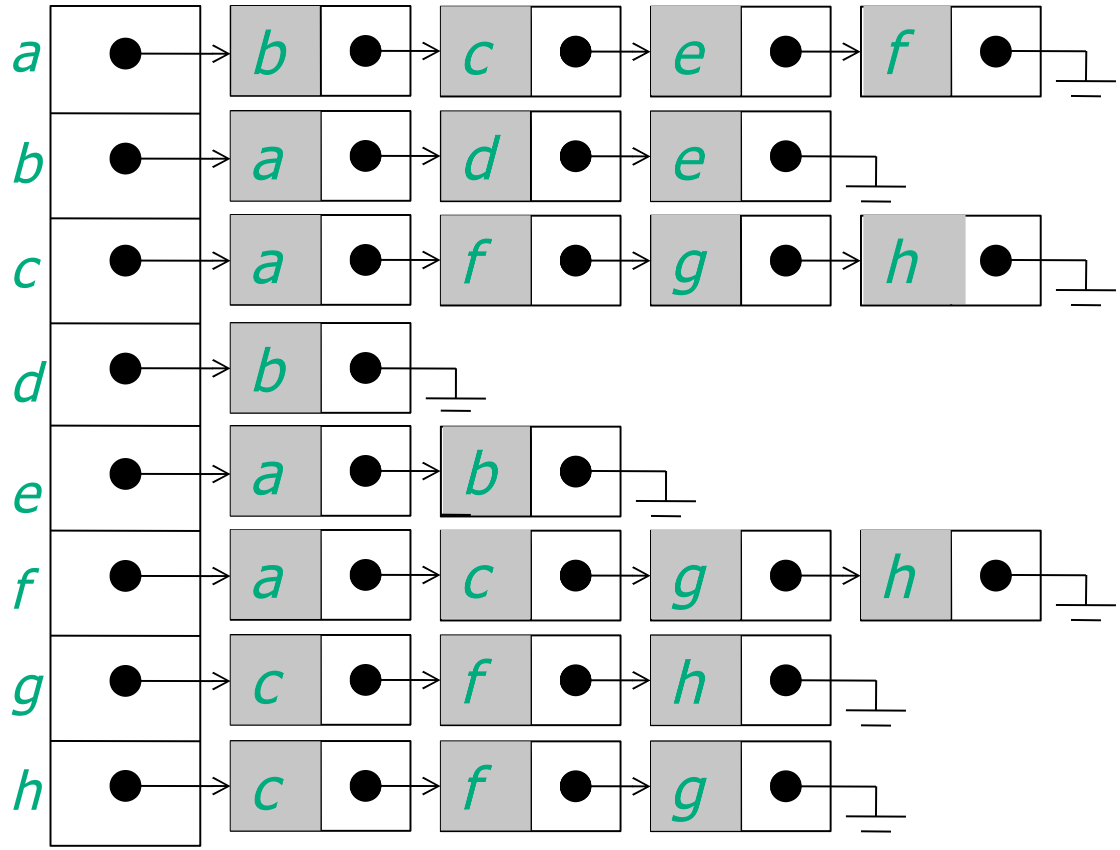
aresta
da árvore

aresta
de retorno

Busca em Profundidade

Ex.:





aresta
da árvore

aresta
de retorno

Busca em Profundidade

```
algoritmo BP(G,v) {em inglês DFS}
{dados: um grafo simples e conexo G e um vértice v para
início da busca}
início
   marque v;
   para todas as arestas (v,w) faça
   {é equivalente a para todo w ∈ Adj(v) faça}
      início
         se w é não marcado então BP(G,w);
      fim
fim
```

Busca em Profundidade

- **Complexidade do algoritmo básico BP**
  - **Espaço:**

Busca em Profundidade

- **Complexidade do algoritmo básico BP**
  - **Espaço:** **O(n+m)** EA para armazenar o grafo.

Busca em Profundidade

- **Complexidade do algoritmo básico BP**
  - **Espaço: O(n+m)** EA para armazenar o grafo.
  - **Tempo:**

Busca em Profundidade

- **Complexidade do algoritmo básico BP**
  - **Espaço: O(n+m)** EA para armazenar o grafo.

  - **Tempo:** Depende de quantas chamadas fazemos e do trabalho realizado em cada uma delas. Ao se encerrar uma chamada de *BP(G,v)* temos que o trabalho realizado por ela foi na lista de adjacentes do vértice *v*: $O(\Sigma\ Adj(v)\ )$. Mas, podemos realizar n chamadas, cada uma delas na marcação de cada vértice do grafo, portanto o tempo total é dado por:

Busca em Profundidade

- **Complexidade do algoritmo básico BP**
  - **Espaço: O(n+m)** EA para armazenar o grafo.
  - **Tempo:** Depende de quantas chamadas fazemos e do trabalho realizado em cada uma delas. Ao se encerrar uma chamada de *BP(G,v)* temos que o trabalho realizado por ela foi na lista de adjacentes do vértice *v: O(Σ Adj(v)).* Mas, podemos realizar n chamadas, cada uma delas na marcação de cada vértice do grafo, portanto o tempo total é dado por:

varrer toda EA

$$\mathbf{O(\sum_{i=1}^{n} Adj(v_i)) = O(n+m)}$$

Busca em Profundidade

Precisamos provar que o algoritmo de Busca em Profundidade está correto, ou seja que ele consegue percorrer todos os vértices e arestas do grafo, se o grafo for conexo.

Proposição 1

Se G é conexo então todos os vértices serão marcados pelo algoritmo BP e todas as arestas serão visitadas pelo menos uma vez durante a execução do algoritmo.

Busca em Profundidade

Prova da Proposição 1

Suponha que BP foi aplicado em um grafo conexo *G* e seja *U* o conjunto de vértices não marcados após aplicação de BP. Como *G* é conexo, então existe uma aresta entre algum vértice marcado *v* e um vértice *w* de U. Mas, isso é uma contradição pois quando *v* foi explorado pelo BP todos os vértices adjacentes a ele foram inspecionados pelo algoritmo e se não estavam marcados, foram marcados nesse passo. Logo, todos os vértices foram visitados e como quando um vértice é visitado todas suas arestas são exploradas, então todas arestas de *G* foram exploradas. □

Busca em Profundidade

E se *G* não for conexo?

Busca em Profundidade

E se *G* não for conexo?

BP pode determinar os Componentes de *G*

Aplicação de Busca em Profundidade

```
algoritmo Componentes(G);
{supõe campo na EAD para guardar a numeração dos
 componentes}
procedimento BP(G,v)
início
   marque v;
   v.comp := ncomp;
   para todas as arestas (v,w) faça
      início
         se w é não marcado então BP(G,w);
      fim
fim
```

Aplicação de Busca em Profundidade

```
{programa principal}
Início
   ncomp := 0;
   enquanto existir algum vértice v não marcado faça
      início
         BP(G,v);
         ncomp := ncomp + 1;
      fim
Fim
```

Busca em Profundidade

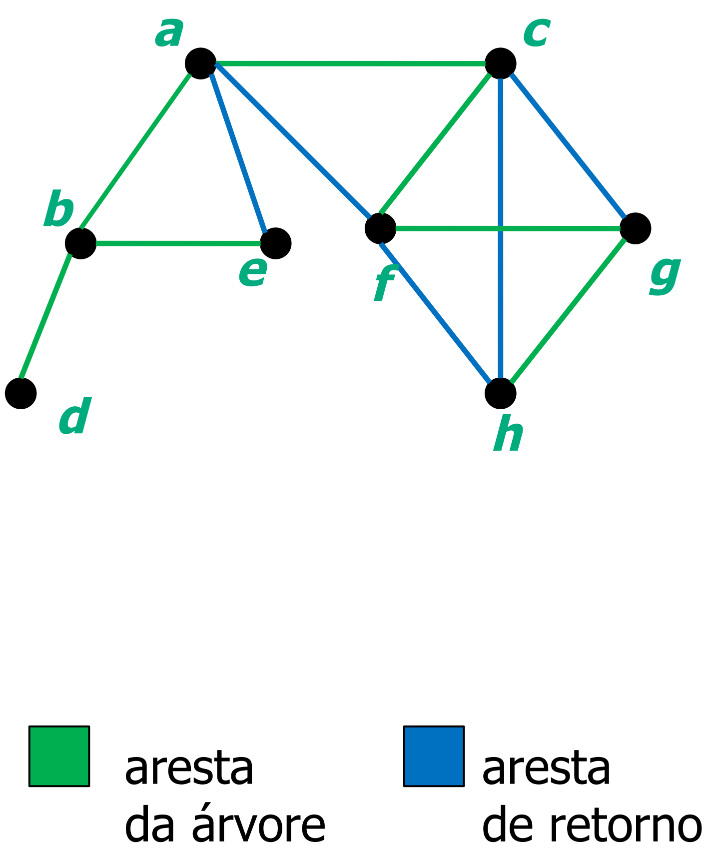
Se G é conexo, o algoritmo BP explora todos os vértices e arestas do grafo e divide as arestas em dois tipos: da **árvore** e de **retorno**.

A árvore geradora construída pelo BP é uma árvore geradora de G e é chamada de **árvore de profundidade**.

Caso G seja desconexo, a aplicação sucessiva de BP forma uma **floresta de profundidade**.

Busca em Profundidade

Ex.:



Busca em Profundidade

```
algoritmo ArvoreGeradora(G,v);
{use o algoritmo BP com pequena variação}
início
   marque v;
   para todas as arestas (v,w) faça
      início
         se w é não marcado então
            início
               adicione (v,w) a T;
               ArvoreGeradora(G,w);
            fim
      fim
fim
```

Busca em Profundidade

Normalmente os problemas que são solucionados através da aplicação de BP, introduzem modificações no código do algoritmo em dois locais: após a marcação de *v* (antes da chamada recursiva) ou após a chamada recursiva.

```
algoritmo BP(G,v) {em inglês DFS}
{dados: um grafo simples e conexo G e um vértice
v para início da busca}
início
   marque v; TrabalhoAnterior em v;
   para todas as arestas (v,w) faça
   {é equivalente a para todo w ∈ Adj(v) faça}
       início
         se w é não marcado então BP(G,w);
         TrabalhoPosterior em (v,w);
       fim
fim
```

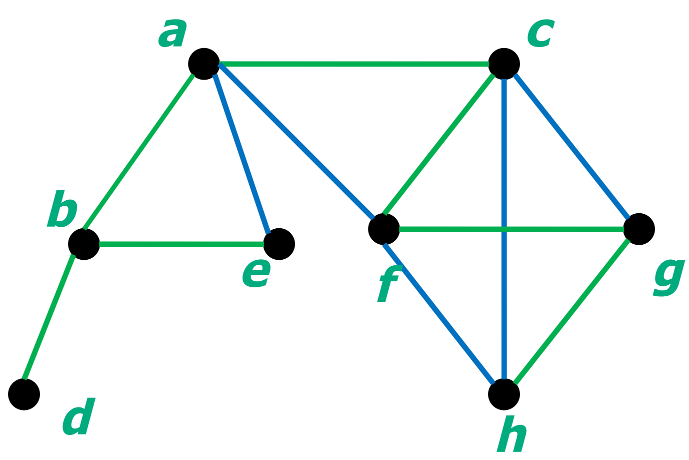
Busca em Profundidade

A ordem em que os vértices são visitados ou explorados é também muitas vezes importante para aplicações. Assim, define-se **profundidade de entrada** e de **saída** de *v* a ordem em que o vértice *v* foi alcançado pela primeira vez e explorado, respectivamente.

Busca em Profundidade

Ex.:

(utilizando a BP feita no exemplo)



	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
PE(*v*)	1	2	5	3	4	6	7	8
PS(*v*)	8	3	7	1	2	6	5	4

Busca em Profundidade

- **EXERCÍCIO DE FIXAÇÃO**
  - Fazer algoritmo de Busca em Profundidade não recursivo. Pensar em que lugar da Busca em Profundidade posso colocar o cômputo da profundidade de entrada e de saída.

Exercícios Recomendados

- Jayme Szwarcfiter: 4.1*, 4.2*, 4.4*, 4.5**, 4.6*
- * Exercício de fixação
- ** Exercício de prova

1 - Prove que um grafo simples e seu complemento não podem ser ambos desconexos.

2 - Os grafos bipartidos completos K1,n, conhecidos como grafos estrelas, são árvores. Prove que grafos estrelas são os únicos grafos bipartidos completos que são árvores

3 - Prove que um grafo é bipartido se e somente se todos os seus ciclos tiverem comprimento par

Referências

- Seções 4.1, 4.2 e 4.3 do Szwarcfiter, J. L., *Grafos e Algoritmos Computacionais,* Ed. Campus, 1983.
- Seção 22.3 do Cormen, *Introduction to Algorithms,* MIT Press, 2001.
- Adaptado do material de aula da Profa. Leila Silva
- Seção 1.7 do Grafos: conceitos, algoritmos e aplicações. Goldbarg, E. e Goldbarg M. Elsevier, 2012