7 - Busca em profundidade

Introdução:

A escolha do vértice marcado torna-se única e sem ambiguidade.

Porém ainda são arbitrárias a escolha da raiz da busca, bem como a escolha da aresta a ser explorada a partir do vértice marcado.

Busca em profundidade NÃO DIRECIONADO:

Durante a busca, diversos atributos são definidos para os vértices:

- Cada vértice é incialmente desmarcado.
- Ele se torna marcado quando for descoberto.
- Torna-se explorado quando a lista de adjacência for completamente examinada.

- Cada vértice te dois valores associados a ele:
  - Tempo de descoberta: indica quando ele é visitado pela primeira vez.
  - Tempo de término: indica quando ele foi completamente explorado.

Tempo	v_1	v_2	v_2	...	v_{n-1}	v_n
Descoberta						
Término						

- Necessita-se de um contador global que seja incrementado cada vez que um novo vértice é descoberto e quando algum vértice é completamente explorado.

- Quando um vértice w for descoberto a partir de um vértice v, o predecessor do vértice w na busca será o vértice v ou ainda pai[w] = v.

Durante uma busca uma aresta pode ser rotulada de uma das seguintes formas:

- Aresta de árvore → quando a aresta é usada para visitar w pela primeira vez.
- Aresta de retorno → quando w já tiver sido marcado, porém w não seja predecessor de v na busca.

Algoritmo NÃO DIRECIONADO:

```
Inicialização / Chamada inicial
1.  t ← 0;                                          // Inicializar tempo global
2.  para todo vértice v ∈ V(G) faça
    a.  TD[v] ← 0;                                  // Inicializar tempo de descoberta
    b.  TT[v] ← 0;                                  // Inicializar tempo de término
    c.  pai[v] ← null;                              // Inicializar predecessor ou pai
3.  enquanto existir algum vértice v tal que TD[v] = 0 efetuar
    a.  Executar Busca_Profundidade(v);             // Executar busca para raiz v
```

```
Busca_Profundidade(v)
1.  t ← t + 1;      TD[v] ← t;                    // Definir tempo de descoberta
2.  para todo vértice w ∈ Γ(v) faça               // Para toda a vizinhança de v
    a.  se TD[w] = 0 então                        // Se w é visitado pela 1ª vez
            Visitar aresta {v, w};  pai[w] ← v;   // {v, w} é aresta de árvore
            Executar Busca_Profundidade(w);
    b.  senão se TT[w] = 0 e w ≠ pai[v] então     // Se w for ancestral mas não pai
            Visitar aresta {v, w};                // {v, w} é aresta de retorno
3.  t ← t + 1;      TT[v] ← t;                    // Definir tempo de término
```

O custo da busca é O(n + m), em que n = |V(G)| e m = |E(G)|

Exemplo NÃO DIRECIONADO:

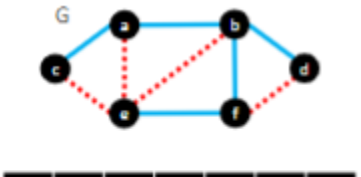
	a	b	c	d	e	f
TD	6	3	7	4	1	2
TT	9	10	8	5	12	11
pai	b	f	a	b	∅	e

Após ele fazer a busca e chegar em um local onde não tem mais para onde ir, como o algoritmo é baseado em pilha (recursividade), ele volta uma posição da recursividade e analisa as arestas a partir daquele ponto para ver se acha alguma aresta incidente a ele, até voltar ao ponto de origem, que no caso do exemplo é o ponto de origem 'e'.

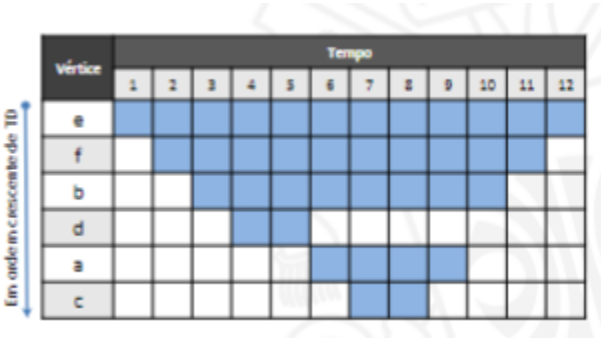
Intervalo de vida NÃO DIRECIONADO:

- Um vértice V é dito vivo durante o intervalo de descoberta e de termino, ao final do intervalo ele morre.
- Qualquer outro vértice W descoberto durante o intervalo de vida de V, morre antes do intervalo de vida de V.

- Portanto, os intervalos de vida de todos os vértices são encaixados uns dentro dos outros.
- O vértice W é descendente do vértice V se e somente se seu intervalo de vida está contido no intervalo de V.



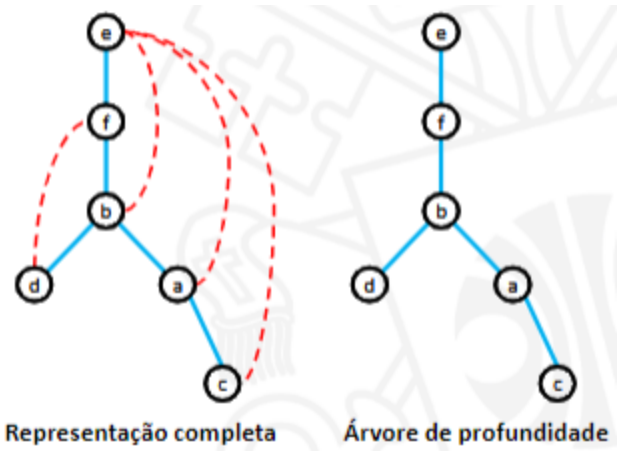
	a	b	c	d	e	f
TD	6	3	7	4	1	2
TT	9	10	8	5	12	11
pai	b	f	a	b	∅	e



Pode-se observar que o vértice ‘f’ é descendente de ‘e’, mas já o vértice ‘a’ não é descendente do vértice ‘d’, pois o intervalo de ‘d’ não está contido no intervalo de ‘a’.

Propriedades NÃO DIRECIONADO:

- Sabendo que o grafo conexo e acíclico é denominado árvore.
- A representação é formada por todos os vértices e apenas as arestas da árvore formam uma árvore de profundidade.
- Caso o grafo seja desconexo, a busca irá produzir várias árvores de profundidade.
- As arestas de retorno sempre representam um ciclo no grafo original.
- A busca não é única, por isso podemos obter mais de uma árvore de profundidade para o mesmo grafo.



Busca em profundidade DIRECIONADO:

Durante uma busca em profundidade em um grafo direcionado as arestas podem ser rotuladas das seguintes formas:

- Aresta de árvore: quando a aresta é usada para visitar W pela primeira vez.
- Aresta de retorno: caso W seja ancestral de V, mas não seu pai.
- Aresta de avanço: caso W seja descendente de V, mas não seja pai dele.
- Aresta de cruzamento: quando W não é descendente de V nem V é descendente de W.

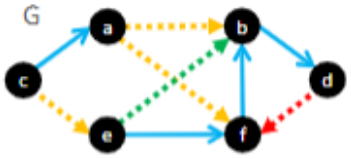
Algoritmo DIRECIONADO:

```
Inicialização / Chamada inicial                          // Idêntico ao anterior
1.  t ← 0;                                               // Inicializar tempo global
2.  para todo vértice v ∈ V(G) faça
    a.  TD[v] ← 0;                                       // Inicializar tempo de descoberta
    b.  TT[v] ← 0;                                       // Inicializar tempo de término
    c.  pai[v] ← null;                                   // Inicializar predecessor ou pai
3.  enquanto existir algum vértice v tal que TD[v] = 0 efetuar
    a.  Executar Busca_Profundidade(v);                  // Executar busca para raiz v
```

```
Busca_Profundidade(v)
1.   t ← t + 1;  TD[v] ← t;                                   // Definir tempo de descoberta
2.   para todo vértice w ∈ Γ+(v) faça                        // Para toda a vizinhança de v
     a.   se TD[w] = 0 então                                 // Se w é visitado pela 1ª vez
               Visitar aresta de árvore (v, w);    pai[w] ← v;
               Executar Busca_Profundidade(w);
     b.   senão
          i.   se TT[w] = 0 então
                    Visitar aresta de retorno (v, w);
          ii.  senão se TD[v] < TD[w] então Visitar aresta de avanço (v, w);
                    senão                   Visitar aresta de cruzamento (v, w);
3.   t ← t + 1;  TT[v] ← t;                                  // Definir tempo de término
```

O custo da busca é O(n + m), em que n = |V(G)| e m = |E(G)|

Exemplo DIRECIONAL:



	a	b	c	d	e	f
TD	10	3	9	4	1	2
TT	11	6	12	5	8	7
pai	c	f	∅	b	∅	e

$BP(e) \Rightarrow \Gamma^+(e) = \{ f, b \}$ $BP(d) \Rightarrow \Gamma^+(d) = \{ f \}$

$BP(f) \Rightarrow \Gamma^+(f) = \{ b \}$ $BP(c) \Rightarrow \Gamma^+(c) = \{ a, e \}$

$BP(b) \Rightarrow \Gamma^+(b) = \{ d \}$ $BP(a) \Rightarrow \Gamma^+(a) = \{ f, b \}$

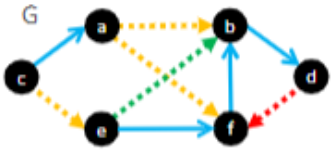
Vértice não explorado — Vértice marcado — Vértice explorado

Aresta não explorada — Aresta de árvore — Aresta de retorno

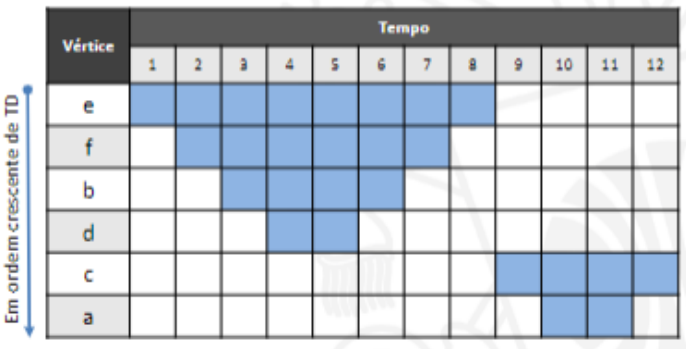
Aresta de avanço — Aresta de cruzamento

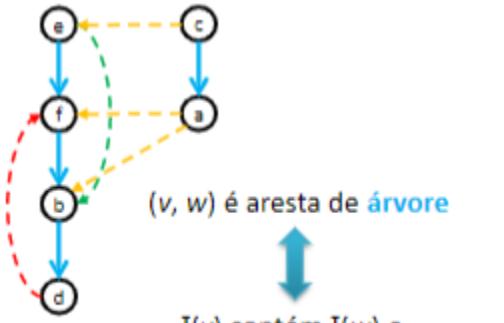
Representação da busca

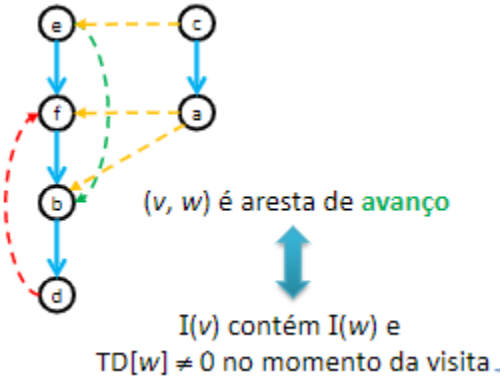
Intervalo de vida DIRECIONAL:

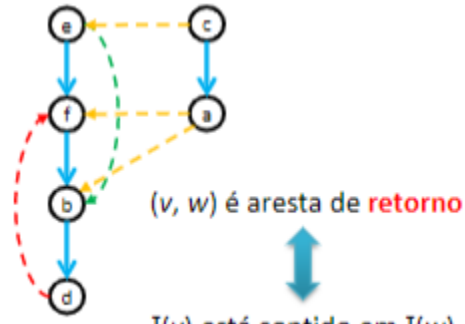


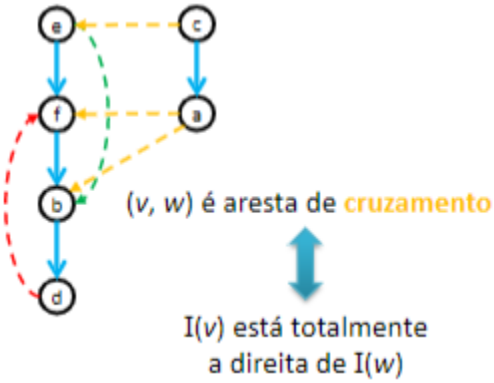
	a	b	c	d	e	f
TD	10	3	9	4	1	2
TT	11	6	12	5	8	7
pai	c	f	∅	b	∅	e




e
c
f
a
b
d
(v, w) é aresta de árvore
I(v) contém I(w) e
TD[w] = 0 no momento da visita


e
c
f
a
b
d
(v, w) é aresta de avanço
I(v) contém I(w) e
TD[w] ≠ 0 no momento da visita


e
c
f
a
b
d
(v, w) é aresta de retorno
I(v) está contido em I(w)


e
c
f
a
b
d
(v, w) é aresta de cruzamento
I(v) está totalmente
a direita de I(w)