GRAFOS

Algoritmos de Pesquisa

Grafos

Propriedades

- Algumas propriedades simples de grafos são fáceis de determinar, pois não dependendo da ordem pela qual se examinam as arestas
  - por exemplo: o grau de todos os vértices
- Outras propriedades, como as associadas a caminhos, é necessário identificá-las através de pesquisa realizada de vértice em vértice ao longo das arestas do caminho
- A maioria dos algoritmos que estudaremos usa este modelo abstracto básico
- Torna-se então necessário analisar o essencial dos algoritmos de pesquisa em grafos e as suas propriedades estruturais
- Pesquisar em grafos é equivalente a percorrer labirintos, sendo necessário
  - marcar pontos já visitados
  - ser-se capaz de recuar, num caminho efetuado, até ao ponto de partida

Algoritmos de pesquisa

Definição

- Dado um grafo G = (V, A) e um vértice S de G, um algoritmo de pesquisa explora o grafo passando por todos os vértices alcançáveis a partir de S
- Serão estudados dois tipos de algoritmos de pesquisa em grafos
  - pesquisa em Largura Primeiro (BFS – "Breadth-first-search")
    - usando uma EAD Fila (FIFO)
  - pesquisa em Profundidade Primeiro (DFS – "Depth-first-search")
    - usando recursividade ou uma EAD Pilha explícita (LIFO)
  - estes algoritmos designam-se também por algoritmos de travessia em grafos

Algoritmo de Pesquisa em Largura Primeiro (BFS)

Propriedades

- O algoritmo de pesquisa em largura primeiro:
  - calcula a distância (menor número de arestas) de S a cada vértice de G
  - produz uma árvore (subgrafo de G) com raiz em S contendo todos os vértices alcançáveis a partir de S
  - nessa árvore o caminho da raiz S a cada vértice corresponde ao caminho mais curto entre os dois vértices
  - aplicado a grafos orientados e não orientados

Propriedades

- Utiliza a seguinte estratégia para efectuar a travessia do grafo
  - todos os vértices à distância k de S são visitados antes de qualquer vértice à distância k+1
- O algoritmo pinta os vértices (inicialmente brancos) de cinzento ou preto:
  - um vértice **branco** ainda não foi descoberto
  - um vértice **cinzento** já foi visitado mas pode ter adjacentes ainda não descobertos (brancos)
  - um vértice **preto** só tem adjacentes já descobertos (cinzentos ou pretos)
- Os cinzentos correspondem à fronteira entre descobertos e não descobertos
- A árvore de travessia em largura é expandida atravessando-se a lista de adjacências de cada vértice cinzento **u**
  - para cada vértice adjacente **v** acrescenta-se à árvore a aresta (**u**, **v**)

Algoritmo

Entrada: grafo G, número de vértices de G e vértice inicial S
Saída: array 1D de inteiros **Pai**

algoritmo BFS:
  para cada vértice u $\in$ V, u $\neq$ S **fazer**
    Cor[u] $\leftarrow$ branco
    Dist[u] $\leftarrow \infty$
    Pai[u] $\leftarrow$ NULO
  fim_para
  Q $\leftarrow$ **criarFila**()
  Q $\leftarrow$ **juntar**(S, Q)
  Cor[S] $\leftarrow$ cinzento
  Dist[S] $\leftarrow$ 0
  Pai[S] $\leftarrow$ NULO
...

Algoritmo (cont)

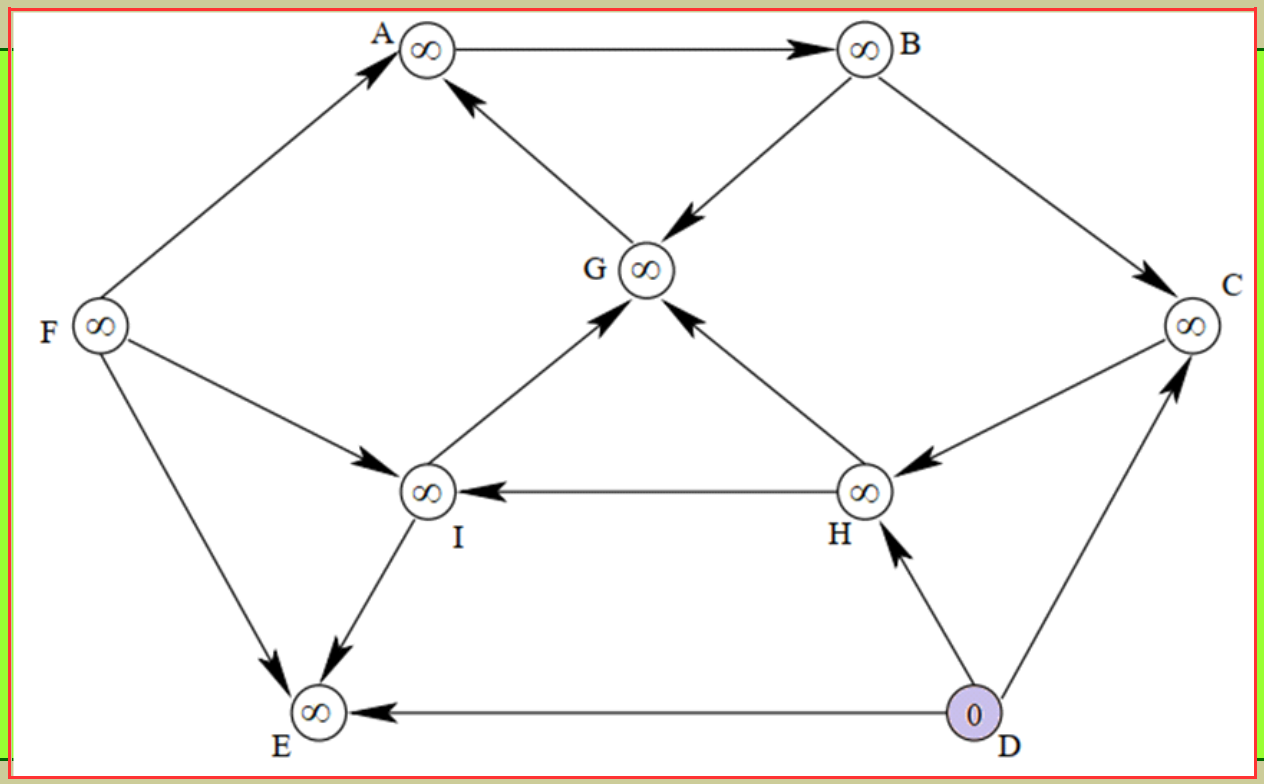
```
...
  enquanto (filaVazia(Q) = falso) fazer
      u ← frente(Q)
      Q ← remover(Q)
      para cada vértice v adjacente ao vértice u fazer
          se (Cor[v] = branco) então
              Q ← juntar(v, Q)
              Cor[v] ← cinzento
              Dist[v] ← Dist[u] + 1
              Pai[v] ← u
          fim_se
      fim_para
      Cor[u] ← preto
  fim_enquanto
fim_algoritmo
```

Estruturas de dados do algoritmo

- Utiliza-se uma fila de espera (estrutura linear FIFO) – ver EAD "Fila"
- Outras estruturas de dados
  - array 1D **Cor** para guardar as cores dos vértices,
  - array 1D **Dist** para as distâncias ao vértice S,
  - array 1D **Pai** para o pai de cada vértice na árvore construída
- Normalmente utiliza-se uma representação por listas de adjacências para guardar os dados do grafo
- A árvore construída não é única, pois depende da ordem pela qual são visitados os vértices adjacentes de um determinado vértice
- No entanto as distâncias de cada vértice ao vértice **S** são únicas

Exemplo (S = D)

u = Frente(Q) = D
Q = [ ]

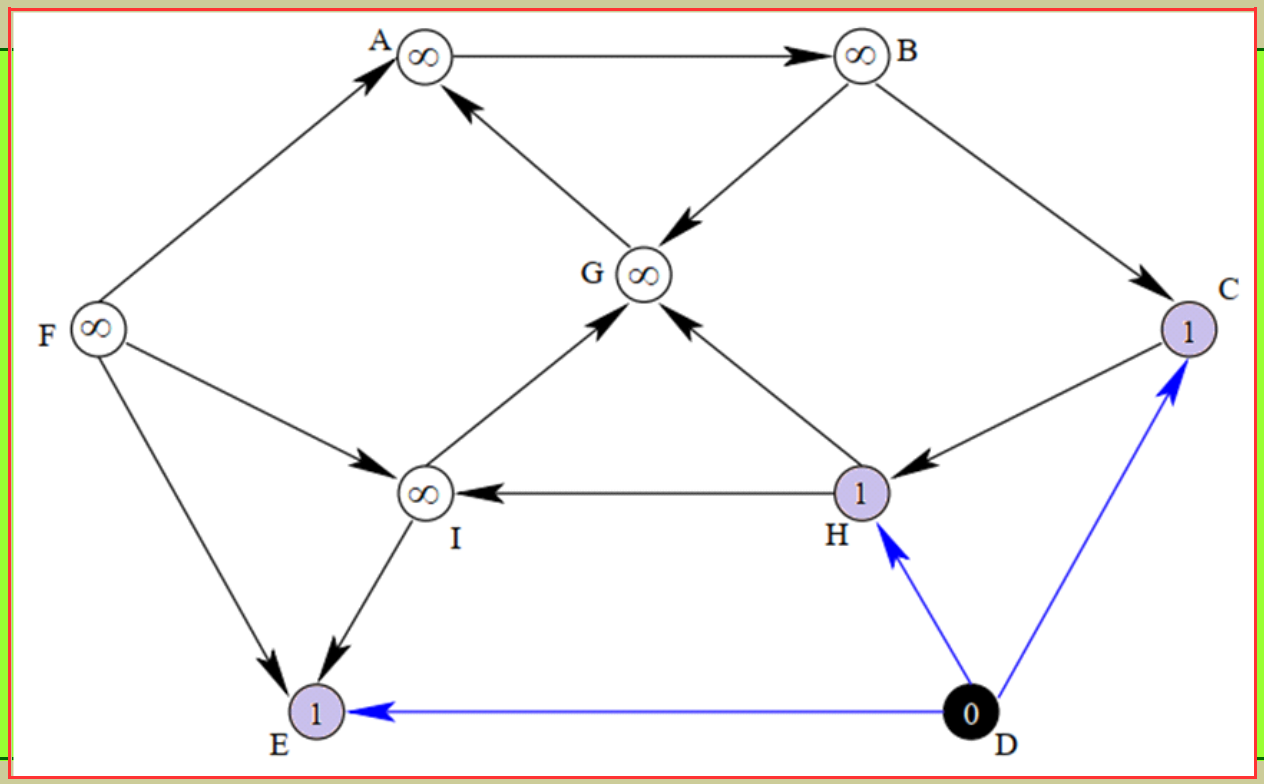


para cada vértice u ∈ V, u ≠ S **fazer**
    Cor[u] ← branco
    Dist[u] ← ∞
    Pai[u] ← NULO
fim_para
Q ← **criarFila**()
Q ← **juntar**(S, Q)
Cor[S] ← cinzento
Dist[S] ← 0
Pai[S] ← NULO

Q = [D]
Pai[D] = NULO

Exemplo (S = D)

u = Frente(Q) = D
Q = [ ]

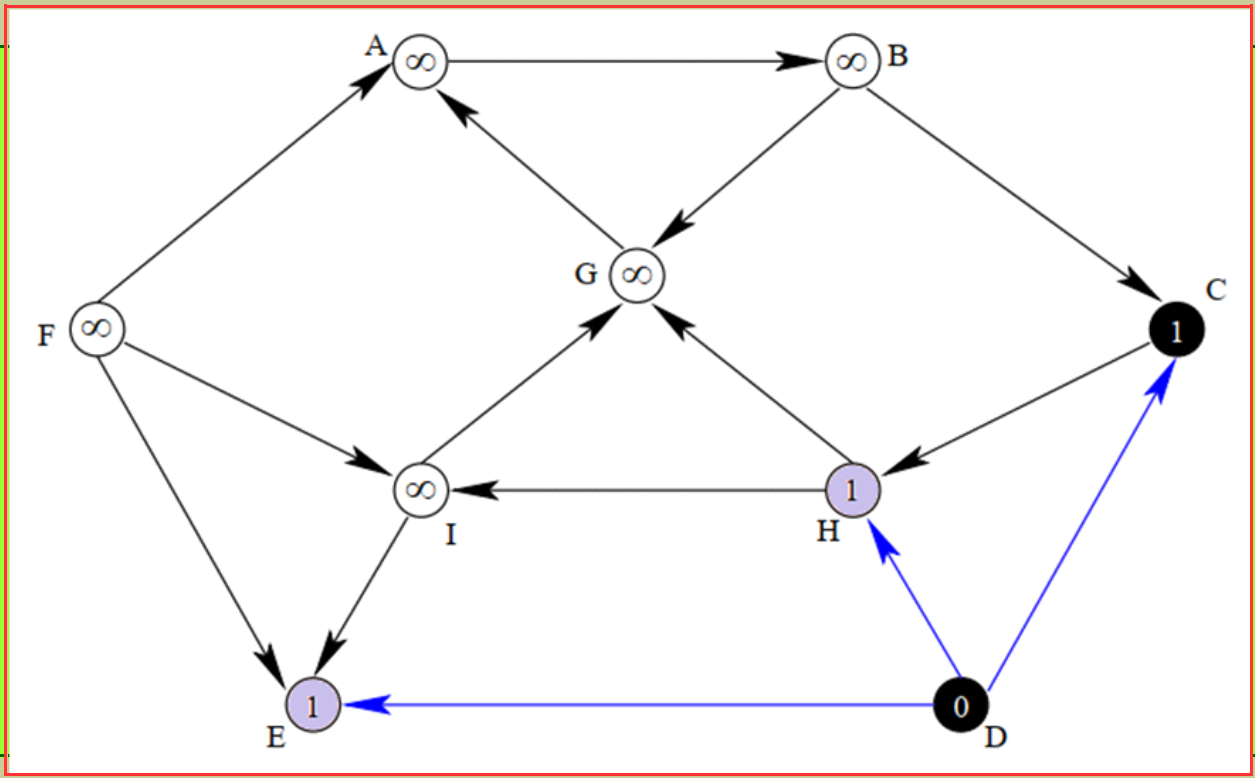



```
enquanto (filaVazia(Q) = falso) fazer
  u ← frente(Q)
  Q ← remover(Q)
  para cada vértice v adjacente a u fazer
    se (Cor[v] = branco) então
      Q ← juntar(v, Q)
      Cor[v] ← cinzento
      Dist[v] ← Dist[u] + 1
      Pai[v] ← u
    fim_se
  fim_para
  Cor[u] ← preto
fim_enquanto
```

Q = [C, E, H]
Pai[C] = D Pai[E] = D Pai[H] = D

Exemplo (S = D)

u = Frente(Q) = C
Q = [E, H]

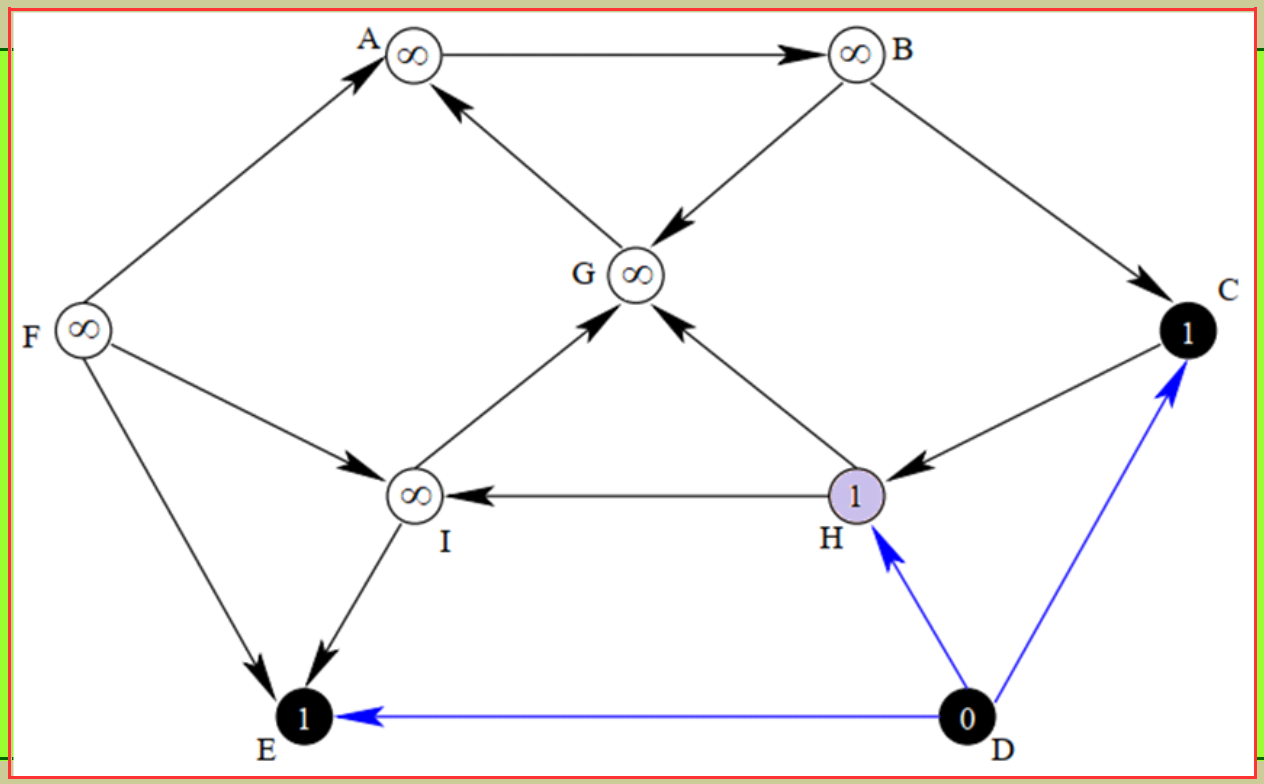



```
enquanto (filaVazia(Q) = falso) fazer
  u ← frente(Q)
  Q ← remover(Q)
  para cada vértice v adjacente a u fazer
    se (Cor[v] = branco) então
      Q ← juntar(v, Q)
      Cor[v] ← cinzento
      Dist[v] ← Dist[u] + 1
      Pai[v] ← u
    fim_se
  fim_para
  Cor[u] ← preto
fim_enquanto
```

Q = [E, H]

Exemplo (S = D)

u = Frente(Q) = E
Q = [H]

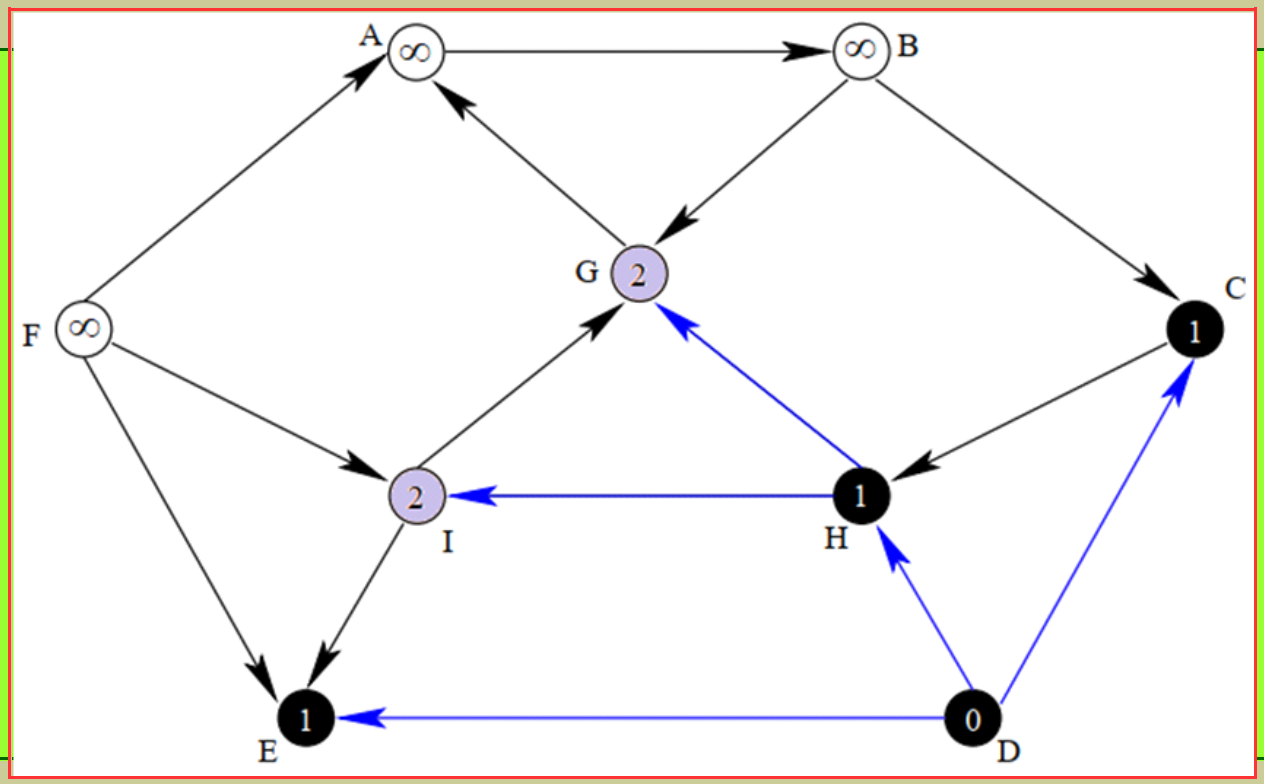



```
enquanto (filaVazia(Q) = falso) fazer
  u ← frente(Q)
  Q ← remover(Q)
  para cada vértice v adjacente a u fazer
    se (Cor[v] = branco) então
      Q ← juntar(v, Q)
      Cor[v] ← cinzento
      Dist[v] ← Dist[u] + 1
      Pai[v] ← u
    fim_se
  fim_para
  Cor[u] ← preto
fim_enquanto
```

Q = [H]

Exemplo (S = D)

u = Frente(Q) = H
Q = [ ]

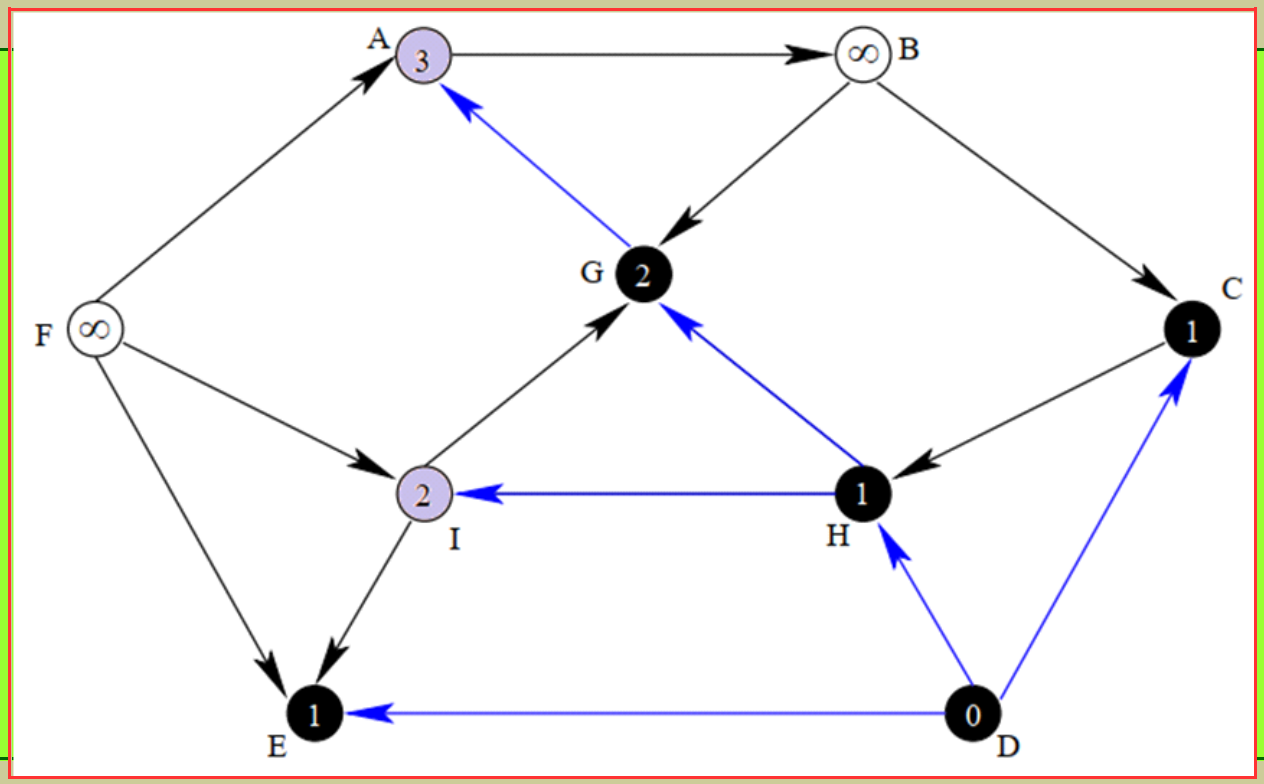



```
enquanto (filaVazia(Q) = falso) fazer
  u ← frente(Q)
  Q ← remover(Q)
  para cada vértice v adjacente a u fazer
    se (Cor[v] = branco) então
      Q ← juntar(v, Q)
      Cor[v] ← cinzento
      Dist[v] ← Dist[u] + 1
      Pai[v] ← u
    fim_se
  fim_para
  Cor[u] ← preto
fim_enquanto
```

Q = [G, I]
Pai[G] = H Pai[I] = H

Exemplo (S = D)

u = Frente(Q) = G
Q = [I]

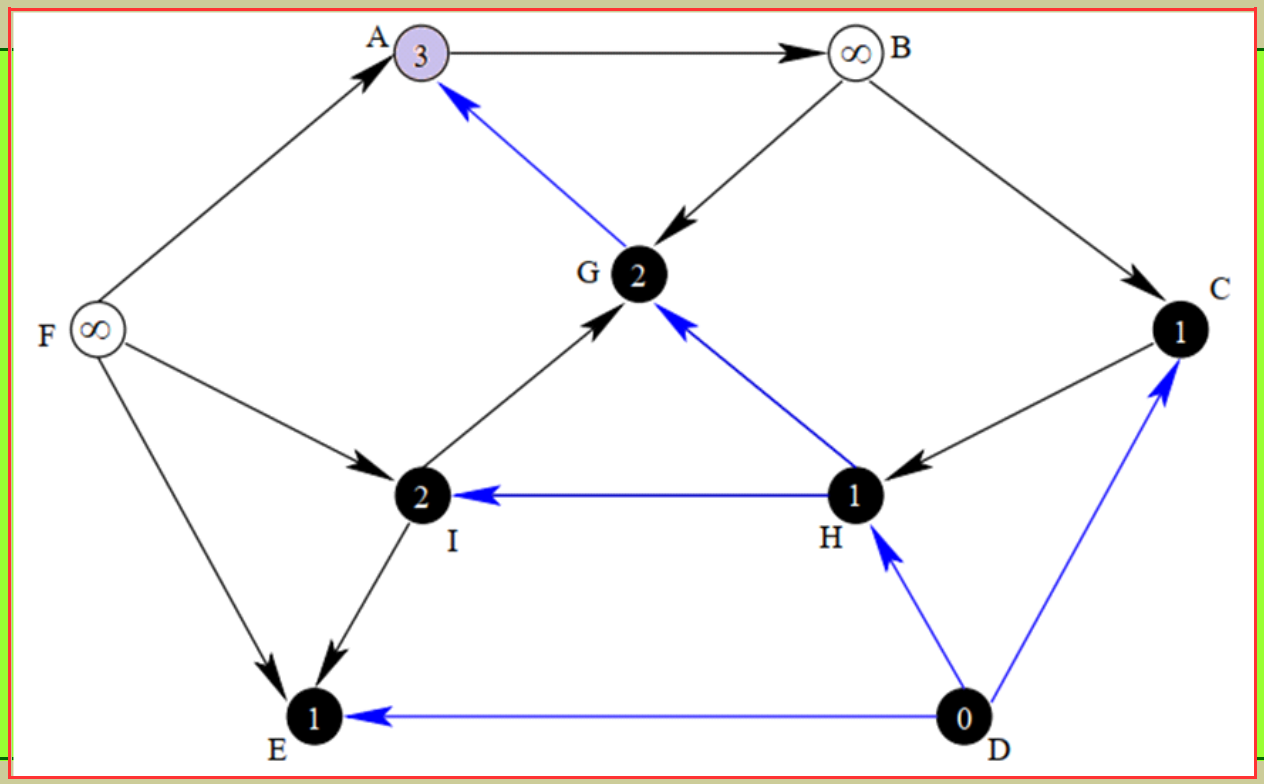



```
enquanto (filaVazia(Q) = falso) fazer
  u ← frente(Q)
  Q ← remover(Q)
  para cada vértice v adjacente a u fazer
    se (Cor[v] = branco) então
      Q ← juntar(v, Q)
      Cor[v] ← cinzento
      Dist[v] ← Dist[u] + 1
      Pai[v] ← u
    fim_se
  fim_para
  Cor[u] ← preto
fim_enquanto
```

Q = [I, A]
Pai[A] = G

Exemplo (S = D)

u = Frente(Q) = I
Q = [A]

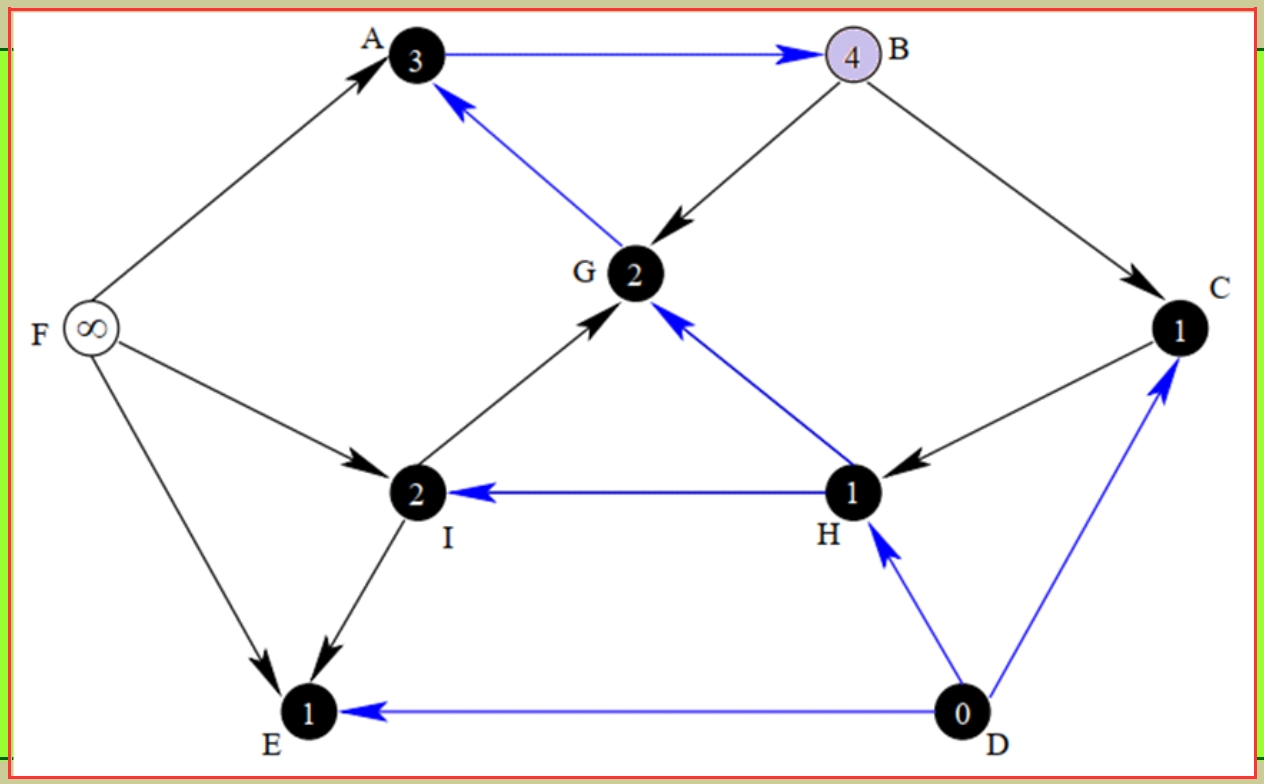



```
enquanto (filaVazia(Q) = falso) fazer
  u ← frente(Q)
  Q ← remover(Q)
  para cada vértice v adjacente a u fazer
    se (Cor[v] = branco) então
      Q ← juntar(v, Q)
      Cor[v] ← cinzento
      Dist[v] ← Dist[u] + 1
      Pai[v] ← u
    fim_se
  fim_para
  Cor[u] ← preto
fim_enquanto
```

Q = [A]

Exemplo (S = D)

u = Frente(Q) = A
Q = [ ]

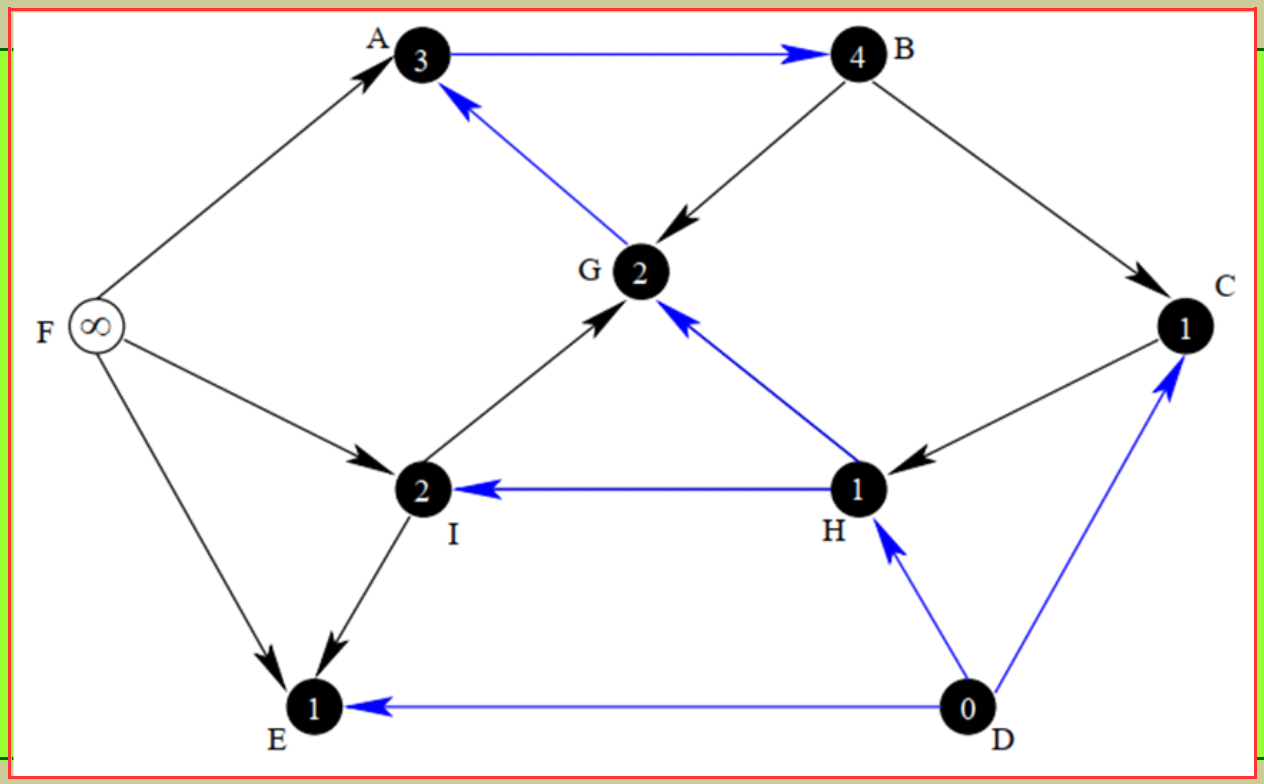



```
enquanto (filaVazia(Q) = falso) fazer
  u ← frente(Q)
  Q ← remover(Q)
  para cada vértice v adjacente a u fazer
    se (Cor[v] = branco) então
      Q ← juntar(v, Q)
      Cor[v] ← cinzento
      Dist[v] ← Dist[u] + 1
      Pai[v] ← u
    fim_se
  fim_para
  Cor[u] ← preto
fim_enquanto
```

Q = [B]
Pai[B] = A

Exemplo (S = D)

u = Frente(Q) = B
Q = [ ]




```
enquanto (filaVazia(Q) = falso) fazer
  u ← frente(Q)
  Q ← remover(Q)
  para cada vértice v adjacente a u fazer
    se (Cor[v] = branco) então
      Q ← juntar(v, Q)
      Cor[v] ← cinzento
      Dist[v] ← Dist[u] + 1
      Pai[v] ← u
    fim_se
  fim_para
  Cor[u] ← preto
fim_enquanto
```

Q = [ ]

Exemplo (resultados)

- O vértice F não é alcançável a partir do vértice D, logo o algoritmo não passa por este vértice
- Em geral, num grafo não fortemente ligado, é necessário iniciar uma nova pesquisa em cada componente ligado, para garantir que todos os vértices são alcançados
- A árvore de pesquisa em largura do grafo (a **azul** no exemplo) fica definida pelo array **Pai**:
  - Pai[C] = D; Pai[E] = D; Pai[H] = D
  - Pai[G] = H; Pai[I] = H
  - Pai[A] = G
  - Pai[B] = A
- Valores do array **Dist** aparecem dentro dos vértices

Árvore de Pesquisa em Largura (APL)

- Dado G = (V, A) e um array **Pai** dos antecessores de cada vértice,

  define-se $G_{pai} = (V_{Pai}, A_{Pai})$, o sub-grafo dos antecessores de G, como se segue

  $$V_{pai} = \{ v \in V \mid Pai[v] \neq NULL \} \cup \{ S \}$$

  $$A_{pai} = \{ (Pai[v], v) \mid v \in V_{pai} - \{ S \} \}$$

- Este grafo é uma **árvore de pesquisa em largura** (**APL**) de G a partir de **S** se
  - V_{Pai} for o conjunto de vértices alcançáveis a partir de **S**, e
  - para cada **v** $\in V_{Pai}$ o caminho mais curto (em G) de **S** até **v** pertencer a G_{Pai}

- Teorema (Para G orientado ou não orientado)

  O algoritmo de pesquisa em largura constrói um array Pai tal que G_{Pai} é uma **árvore de pesquisa em largura** de G

Algoritmo de Pesquisa em Profundidade Primeiro (DFS)

Propriedades

- Este algoritmo utiliza a seguinte estratégia para efectuar a travessia do grafo
  - as próximas arestas a explorar têm origem no mais recente vértice descoberto que ainda tenha vértices adjacentes não explorados
- Assim, quando todos os vértices adjacentes ao vértice **v** tiverem sido explorados, o algoritmo recua ("backtracks") para explorar vértices com origem no vértice a partir do qual **v** foi descoberto
- Estudaremos a versão deste algoritmo que percorre todos os vértices do grafo:
  - depois de terminada a pesquisa com origem em **S**, serão feitas novas pesquisas para descobrir os vértices não alcançáveis a partir de **S**
- O grafo dos antecessores de G é neste caso uma floresta composta de várias **árvores de pesquisa em profundidade** (**APP**)

Propriedades

- A coloração (branco, cinzento e preto) garante que cada vértice pertence a uma única árvore; estas são disjuntas
- O algoritmo usa ainda uma noção de marcas temporais
  - d[v] para o instante em que o vértice é descoberto (passa a cinzento)
  - f[v] quando todos os seus adjacentes são descobertos (passa a preto)
  - as marcas são tais que: d[v] < f[v]
  - estas marcas são valores inteiros entre 1 e 2x|V|

Algoritmo

- Algoritmo secundário DFS(G, S, tempo)
  - inicia a pesquisa no vértice S
  - assume que foram feitas todas as inicializações: tempo, Cor e Pai

Algoritmo

Entrada**:** grafo G, vértice inicial S e tempo
Saída: tempo atualizado

```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
     se (Cor[v] = branco) então
        Pai[v] ← S
        DFS(G, v, tempo)
     fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Algoritmo

- Algoritmo principal: CDFS(G, S)
  - utiliza o algoritmo DFS como sub-rotina
  - efetua todas as inicializações
  - garante que todos os vértices são descobertos
  - cada chamada do algoritmo DFS(G, u, tempo) gera uma nova APP com raiz em **u**

Algoritmo

Entrada: grafo G e o vértice inicial S

Saída: as várias árvores de pesquisa em profundidade (APP)

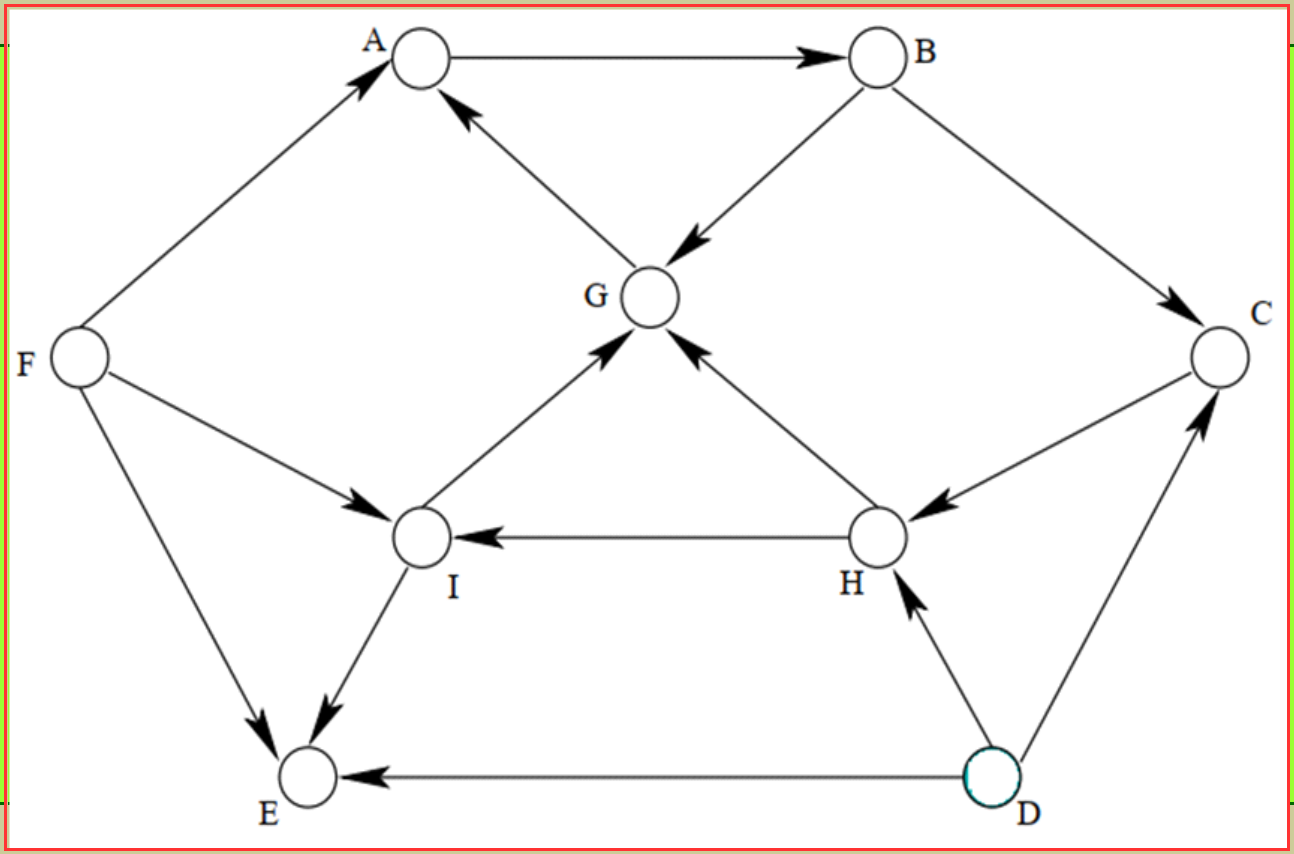
```
algoritmo CDFS(G, S)
   para cada u ∈ V fazer
      Cor[u] ← branco
      Pai[u] ← NULO
   fim_para
   tempo ← 0
   para cada u ∈ V fazer     { o primeiro u = S }
      se (Cor[u] = branco) então
         DFS(G, u, tempo)
      fim_se
   fim_para
fim_algoritmo
```

Observações:

- No fim da execução do algoritmo CDFS todo o vértice u de G tem marcas temporais atribuídas: d[u] e f[u]
- Os resultados produzidos pelo algoritmo (floresta gerada e marcas temporais) não são únicos, pois dependem da ordem pela qual são percorridos os vértices nos diversos ciclos
- No entanto, nas aplicações típicas deste algoritmo, os diversos resultados possíveis são equivalentes

Exemplo: chamada de CDFS(G, D)

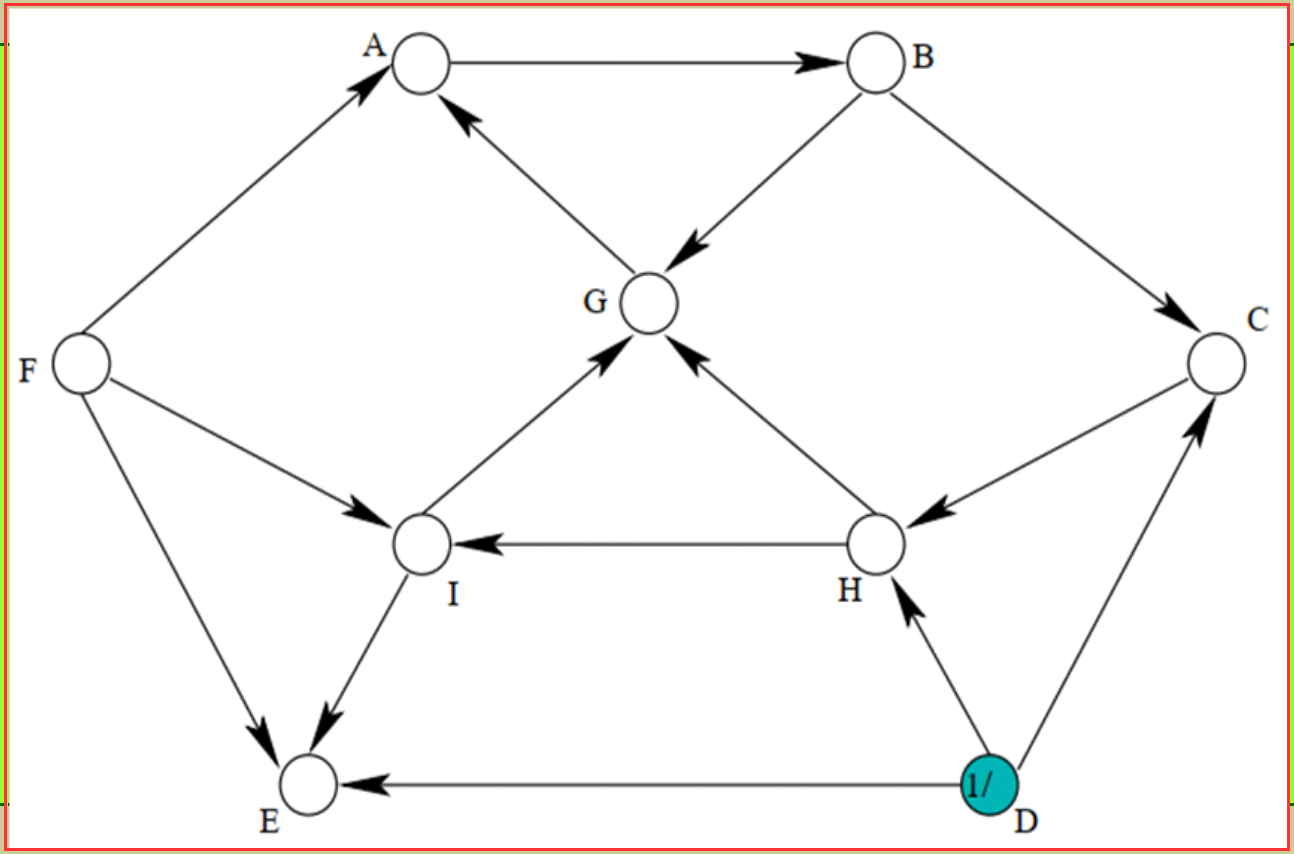
Cor[u] = branco, para todo o vértice u de G
Pai[u] = NULO, para todo o vértice u de G
tempo = 0
Cor[D] = branco ⇒ chamada de **DFS(G, D, 0)**




```
algoritmo CDFS(G, S)
  para cada u ∈ V fazer
     Cor[u] ← branco
     Pai[u] ← NULO
  fim_para
  tempo ← 0
  para cada u ∈ V fazer  // primeiro u = S
     se (Cor[u] = branco) então
        DFS(G, u, tempo)
     fim_se
  fim_para
fim_algoritmo
```

Exemplo: chamada de DFS(G, D, 0)

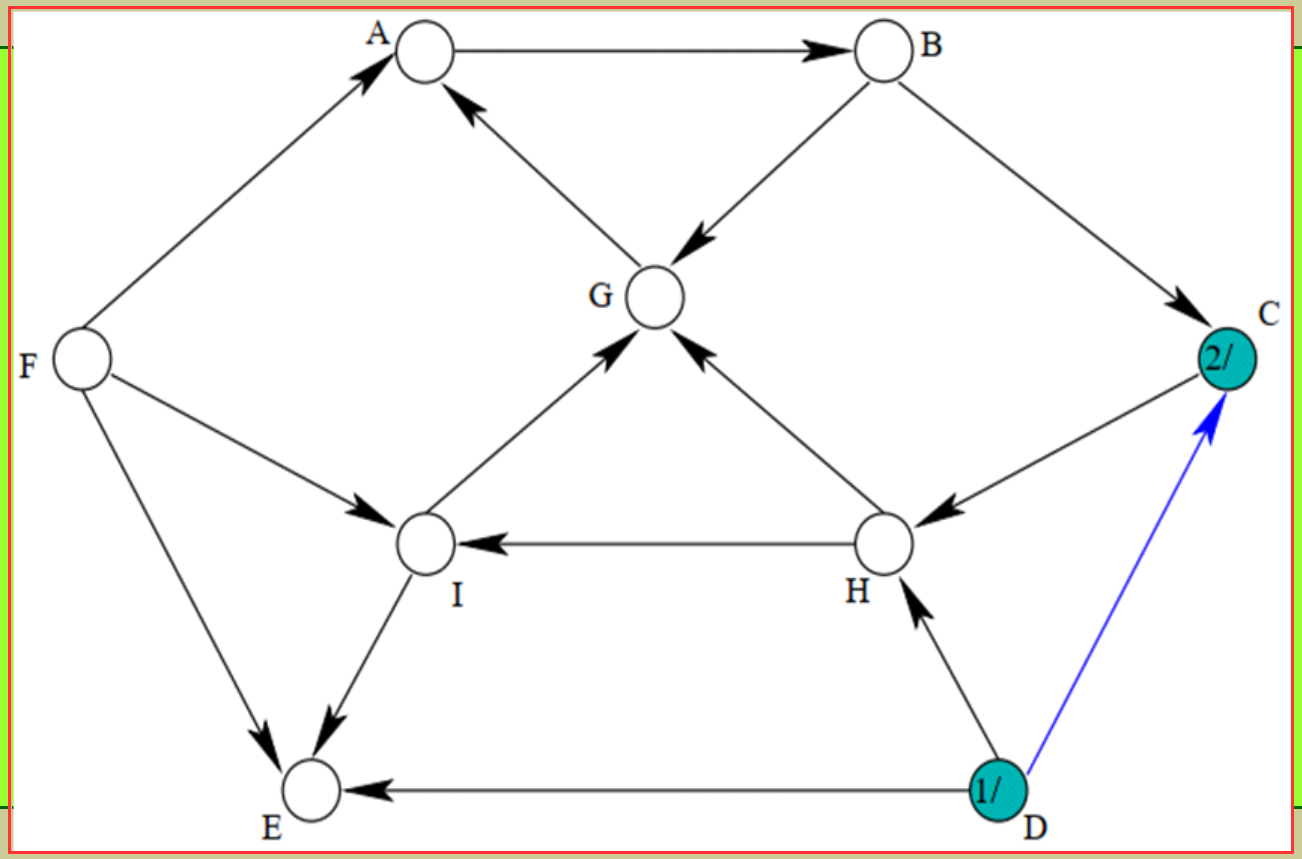
Cor[D] = cinzento; tempo = 1; d[D] = 1
vértice adjacente C (Cor[C] = branco):
    Pai[C] = D
    DFS(G, C, 1)




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: chamada de DFS(G, C, 1)

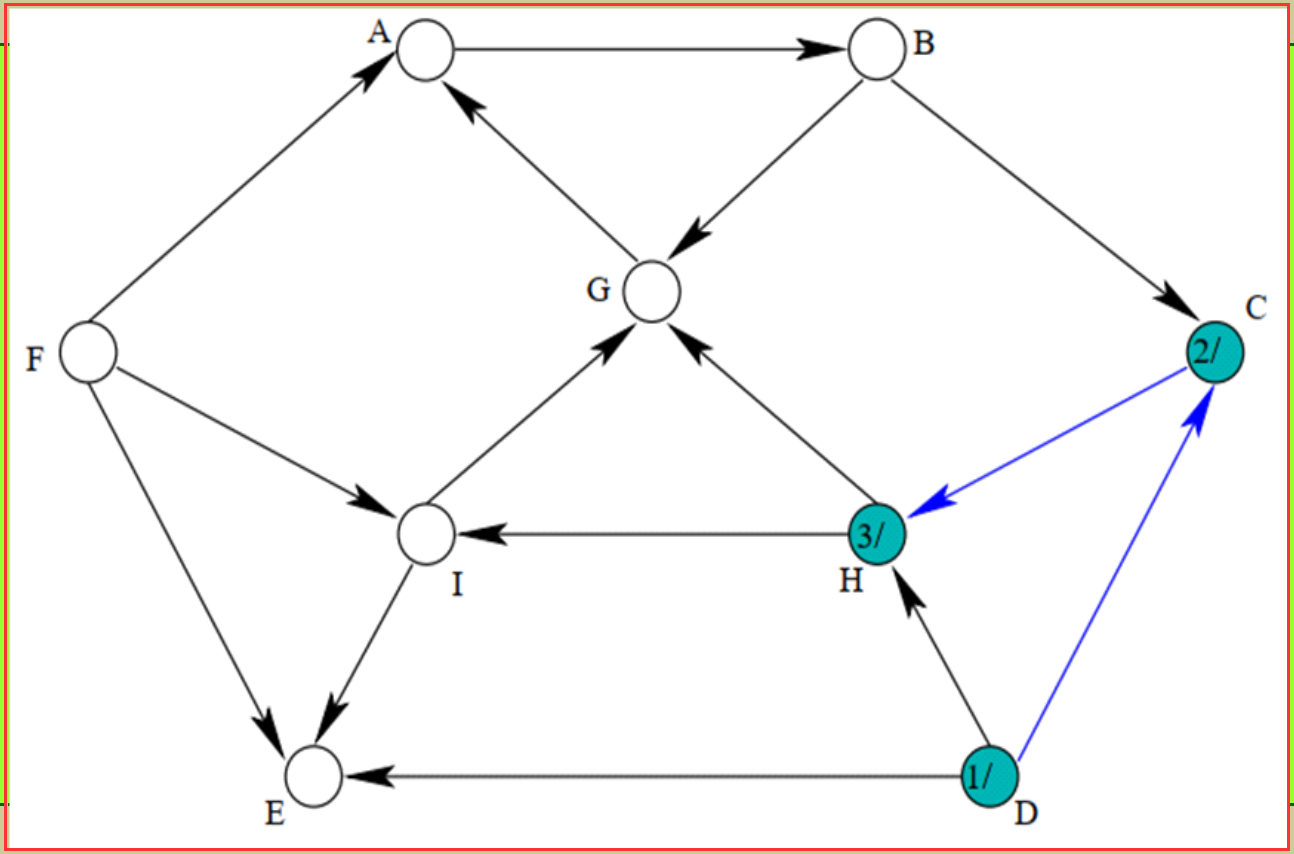
Cor[C] = cinzento; tempo = 2; d[C] = 2
vértice adjacente H (Cor[H] = branco):
    Pai[H] = C
    DFS(G, H, 2)




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: chamada de DFS(G, H, 2)

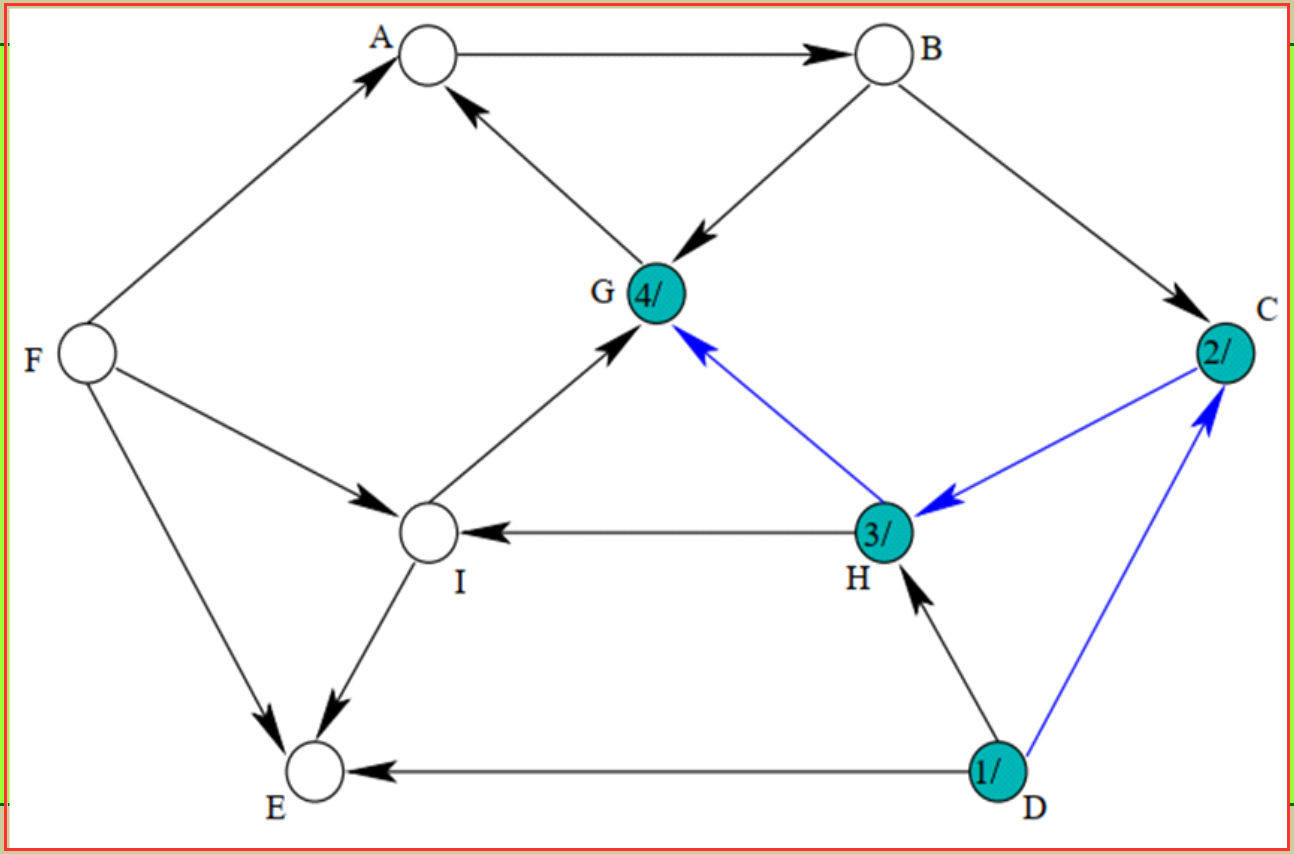
Cor[H] = cinzento; tempo = 3; d[H] = 3
vértice adjacente G (Cor[G] = branco):
    Pai[G] = H
    DFS(G, G, 3)




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: chamada de DFS(G, G, 3)

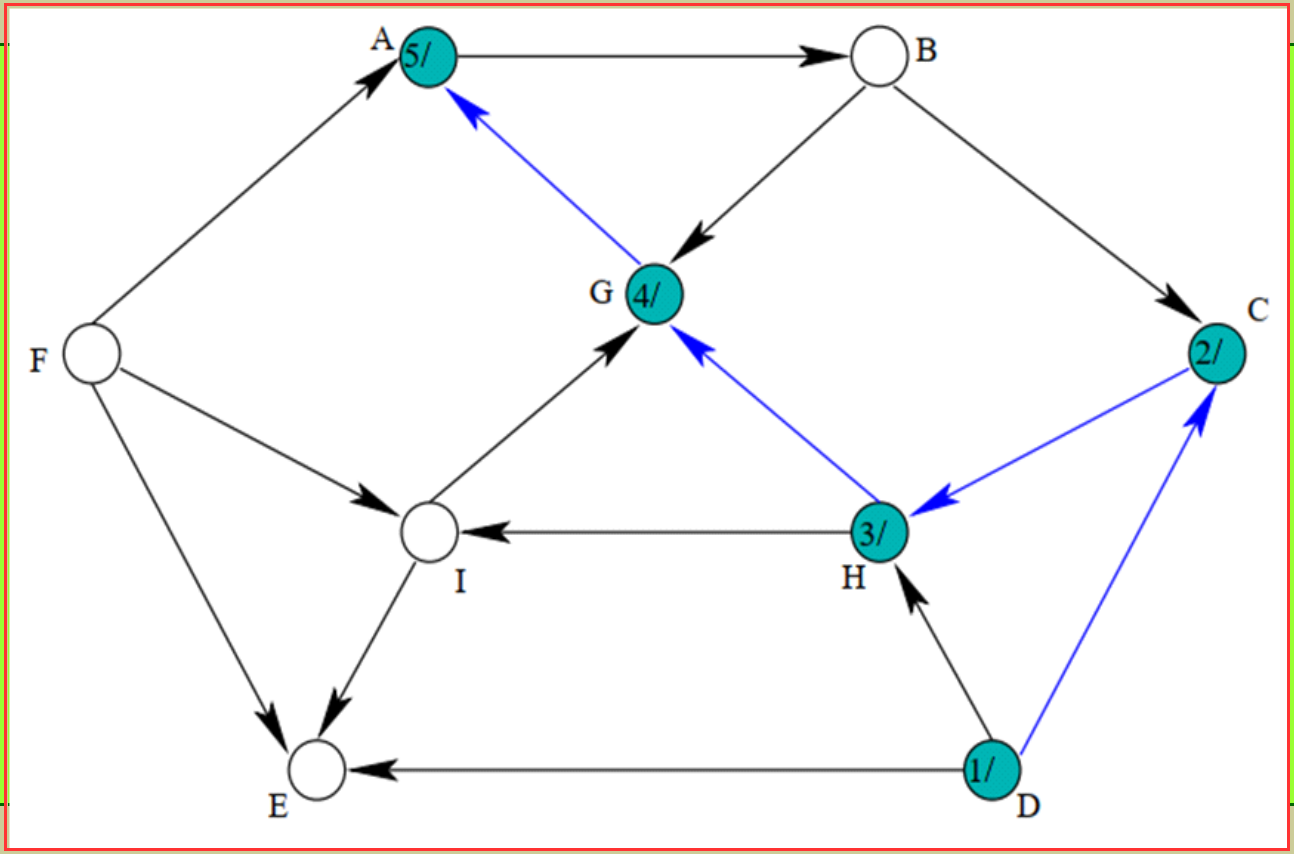
Cor[G] = cinzento; tempo = 4; d[G] = 4
vértice adjacente A (Cor[A] = branco):
    Pai[A] = G
    DFS(G, A, 4)




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
     se (Cor[v] = branco) então
        Pai[v] ← S
        DFS(G, v, tempo)
     fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: chamada de DFS(G, A, 4)

Cor[A] = cinzento; tempo = 5; d[A] = 5
vértice adjacente B (Cor[B] = branco):
    Pai[B] = A
    DFS(G, B, 5)



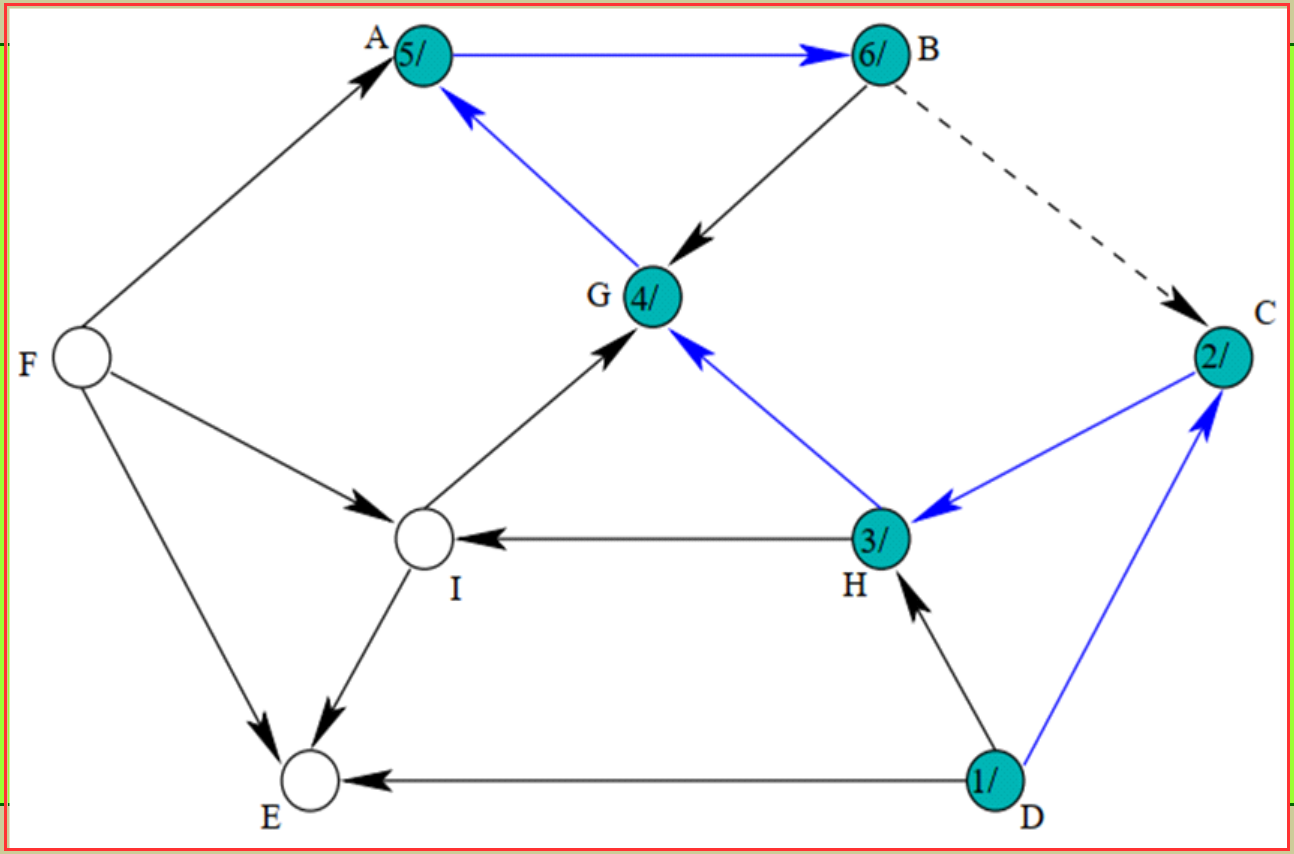

```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
     se (Cor[v] = branco) então
        Pai[v] ← S
        DFS(G, v, tempo)
     fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: chamada de DFS(G, B, 5)

Cor[B] = cinzento; tempo = 6; d[B] = 6

vértice adjacente C (Cor[C] = cinzento (não branco)):

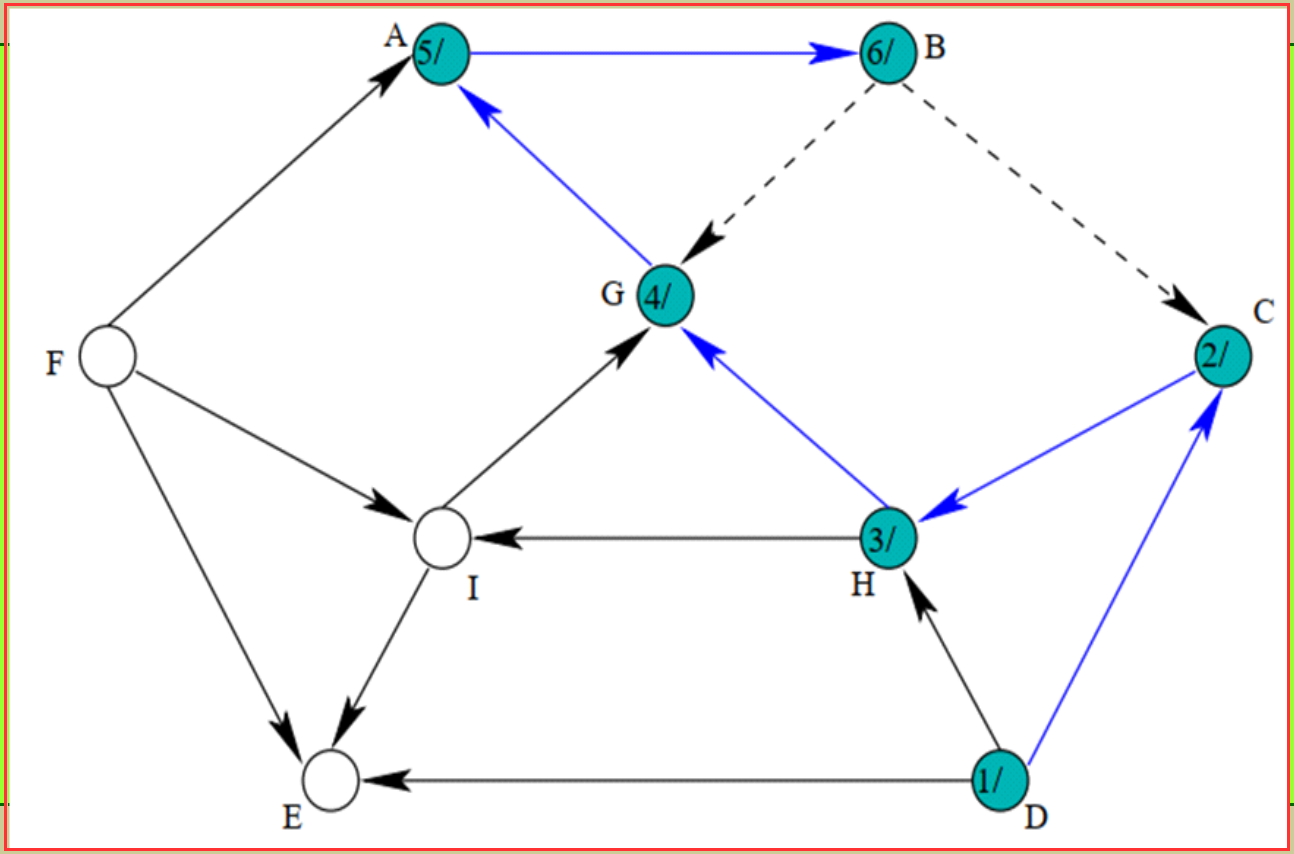
passar ao próximo adjacente




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: chamada de DFS(G, B, 6)

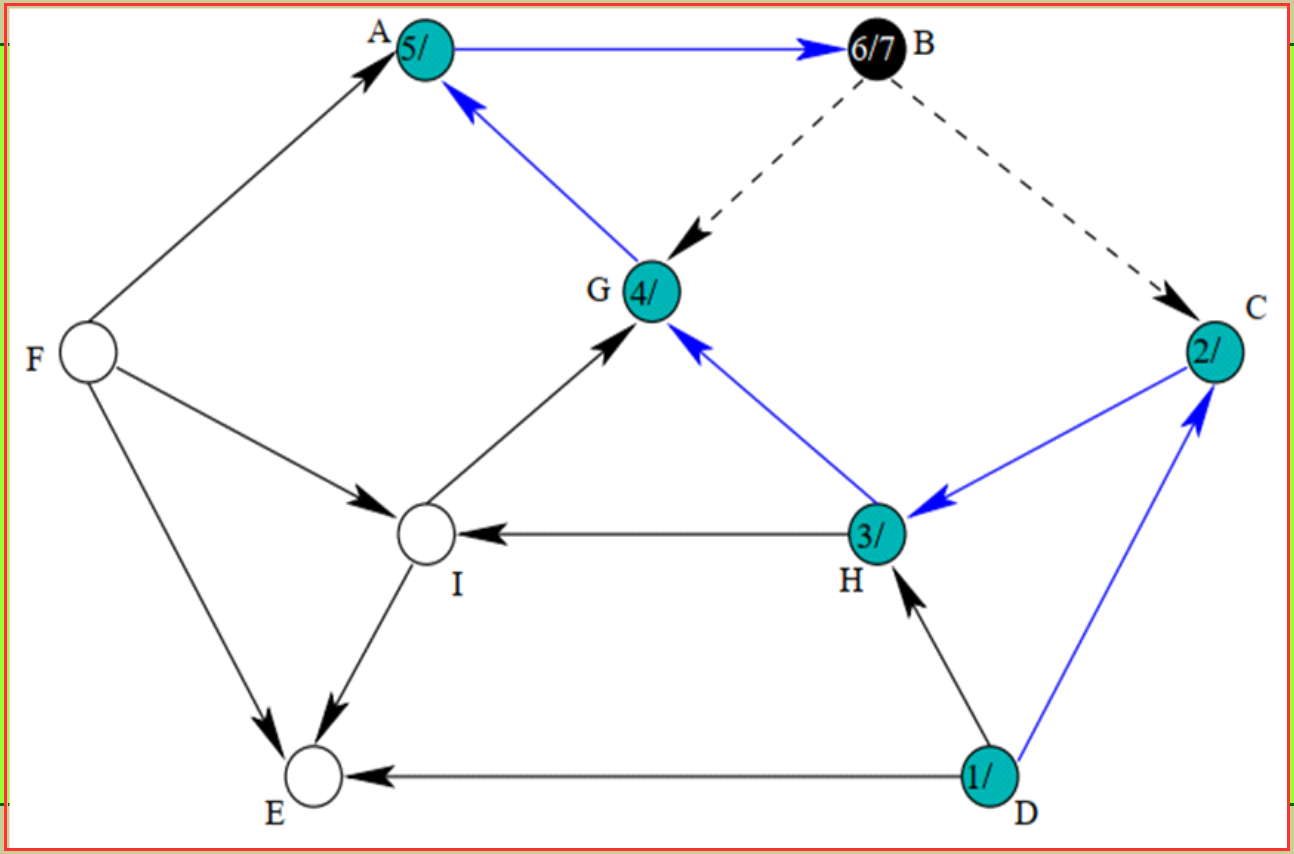
vértice adjacente G (Cor[G] = cinzento (não branco)):
    passar ao próximo adjacente




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: chamada de DFS(G, B, 6)

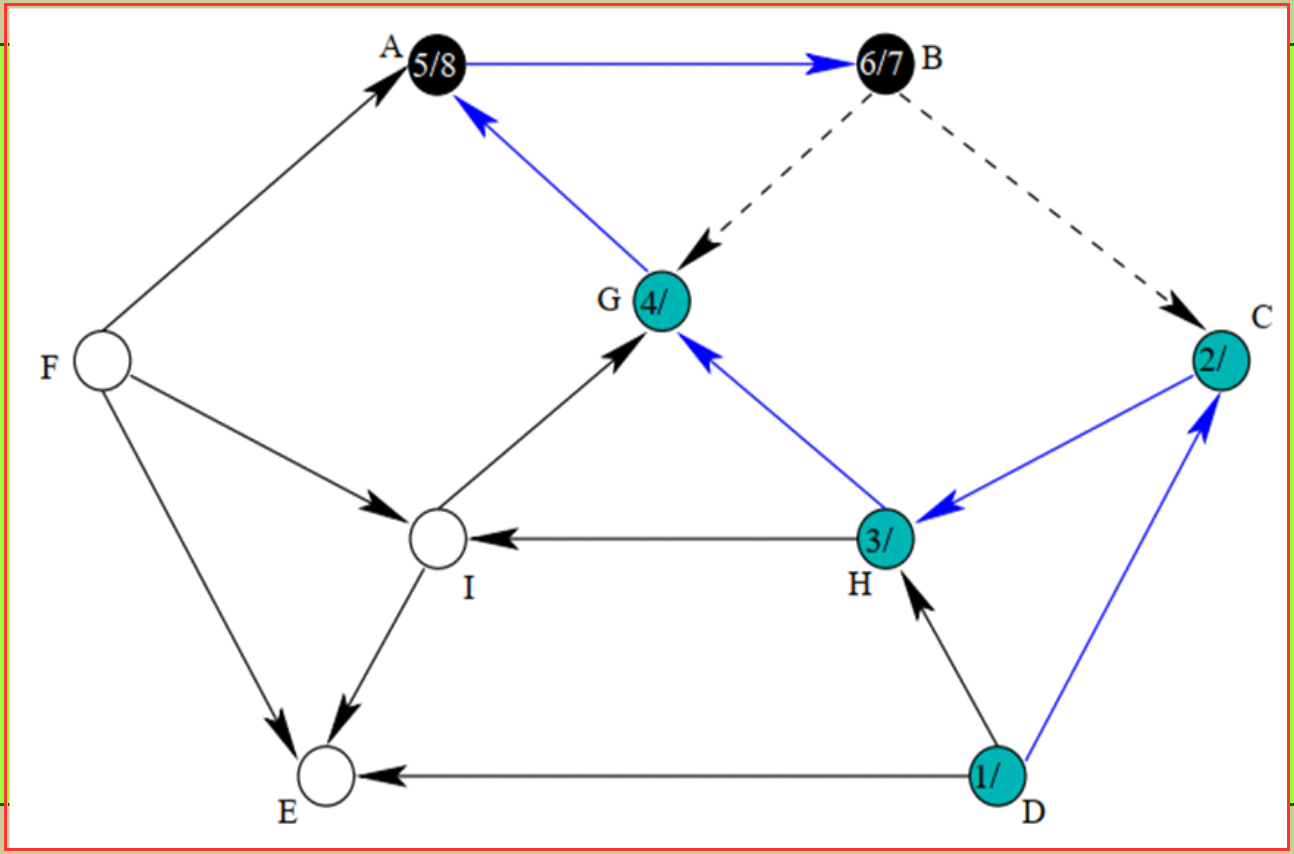
todos os vértices adjacentes de B tratados
Cor[B] = preto
tempo = 7
f[B] = 7




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: recuar ao vértice A (pai de B na APP) - DFS(G, A, 7)

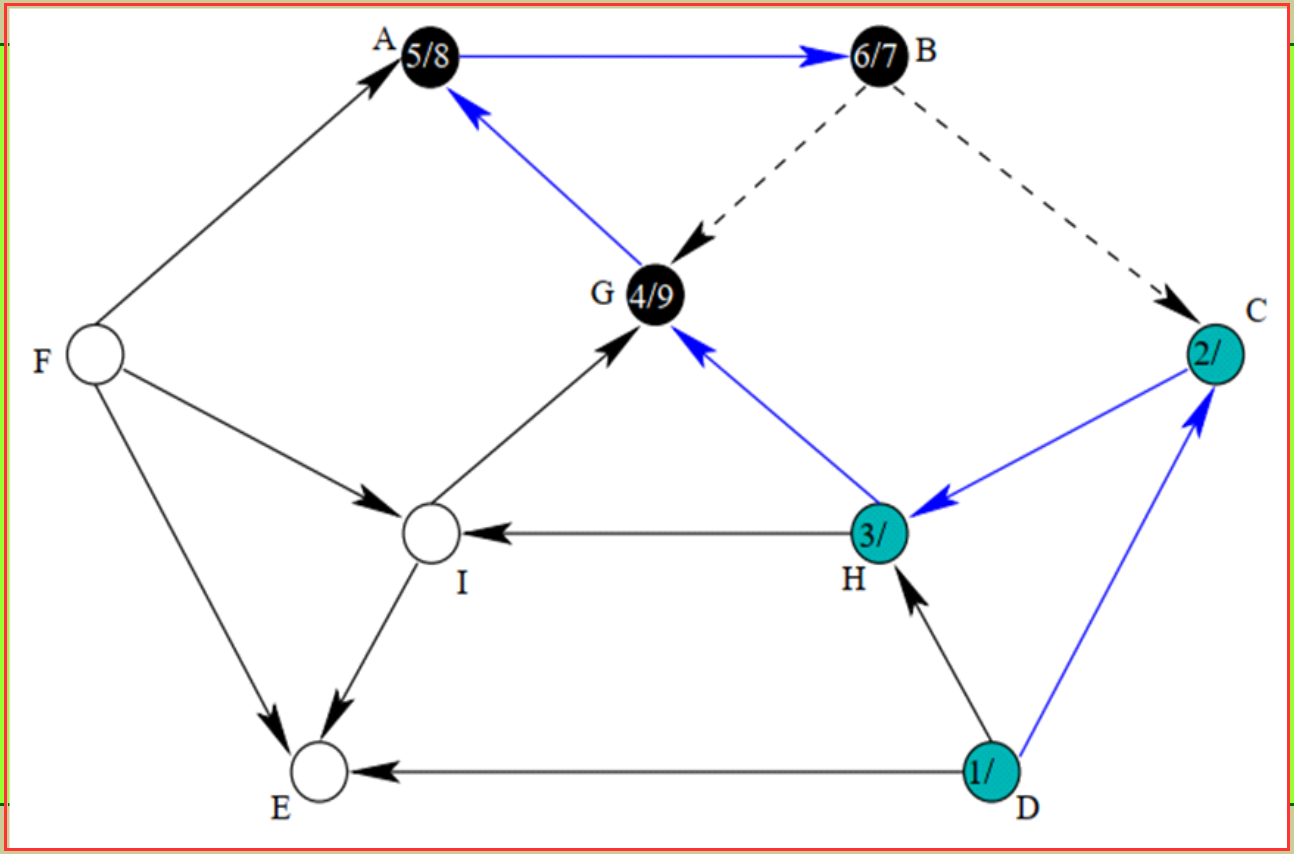
todos os vértices adjacentes de A tratados
Cor[A] = preto
tempo = 8
f[A] = 8




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: recuar ao vértice G (pai de A na APP) - DFS(G, G, 8)

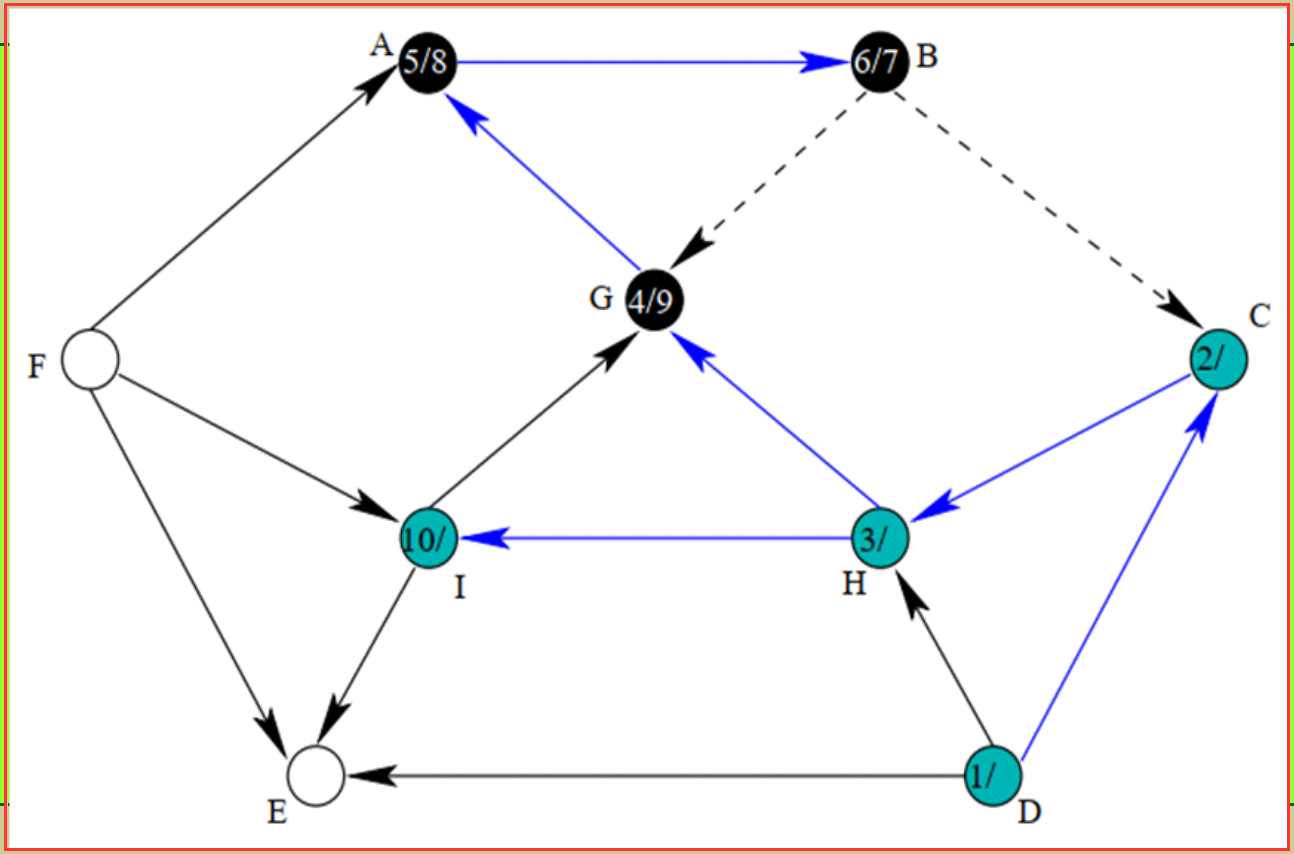
todos os vértices adjacentes de G tratados
Cor[G] = preto
tempo = 9
f[G] = 9




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: recuar ao vértice H (pai de G na APP) - DFS(G, I, 9)

Cor[I] = cinzento; tempo = 10; d[I] = 10
vértice adjacente E (Cor[E] = branco):
    Pai[E] = I
    DFS(G, E, 10)

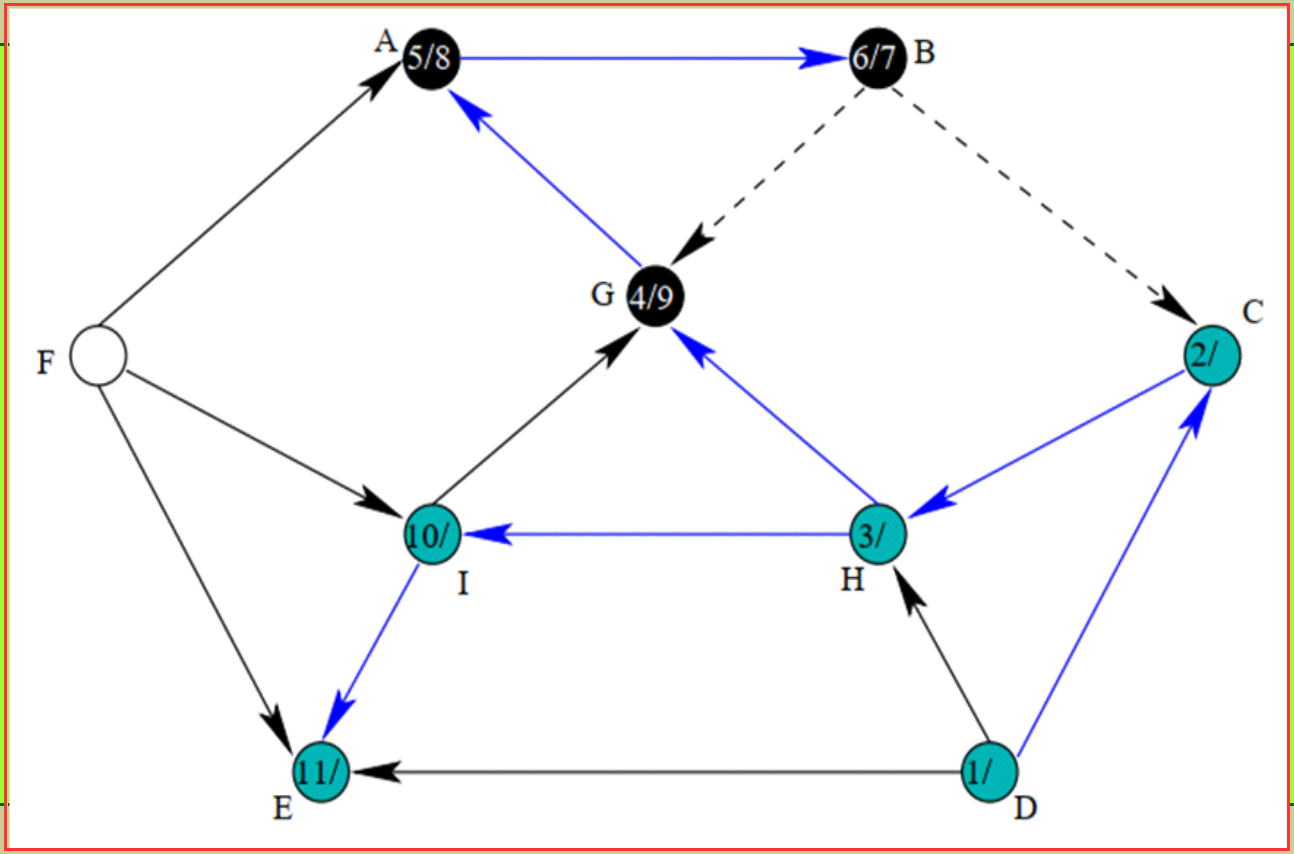



```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: chamada de DFS(G, E, 10)

Cor[E] = cinzento; tempo = 11; d[E] = 11

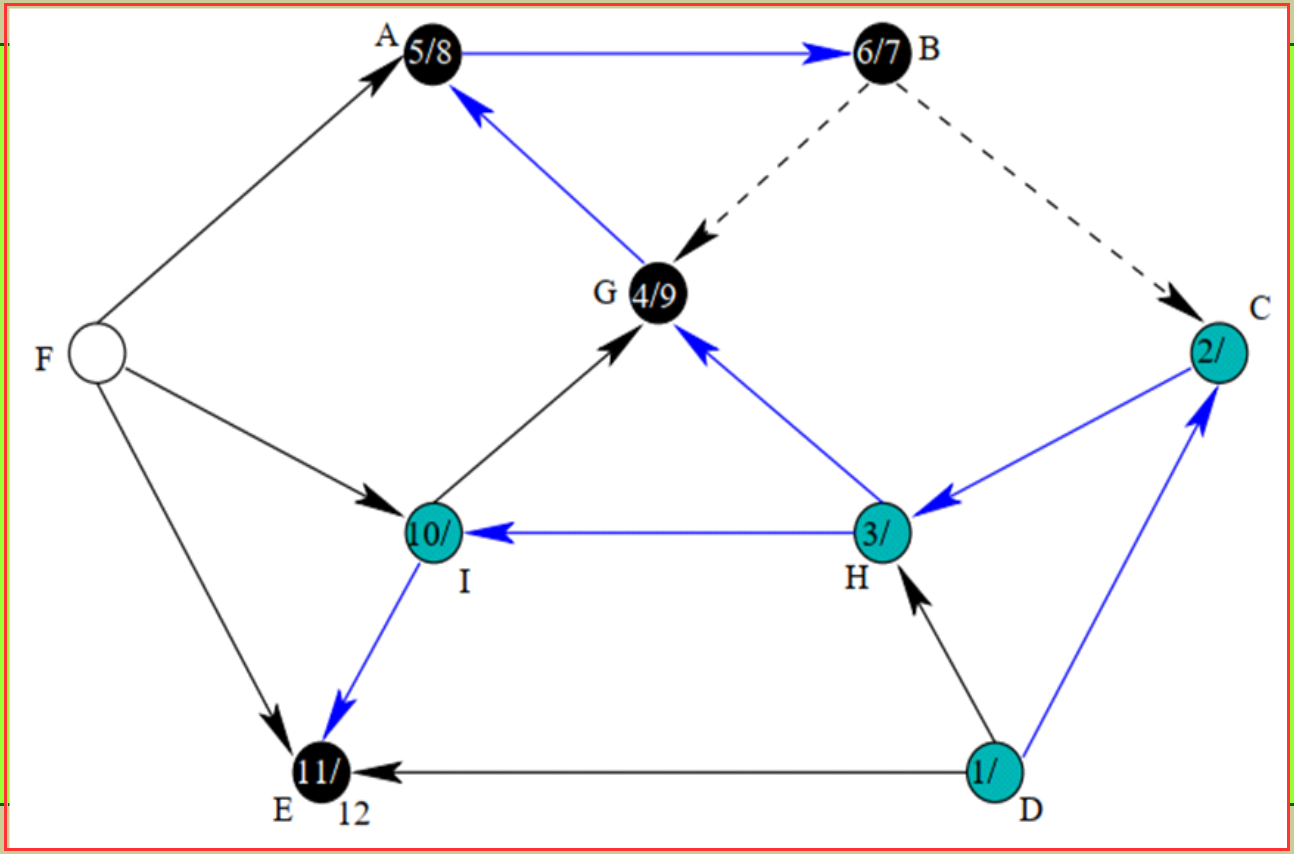
todos os vértices adjacentes têm Cor ≠ branco




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: chamada de DFS(G, E, 11)

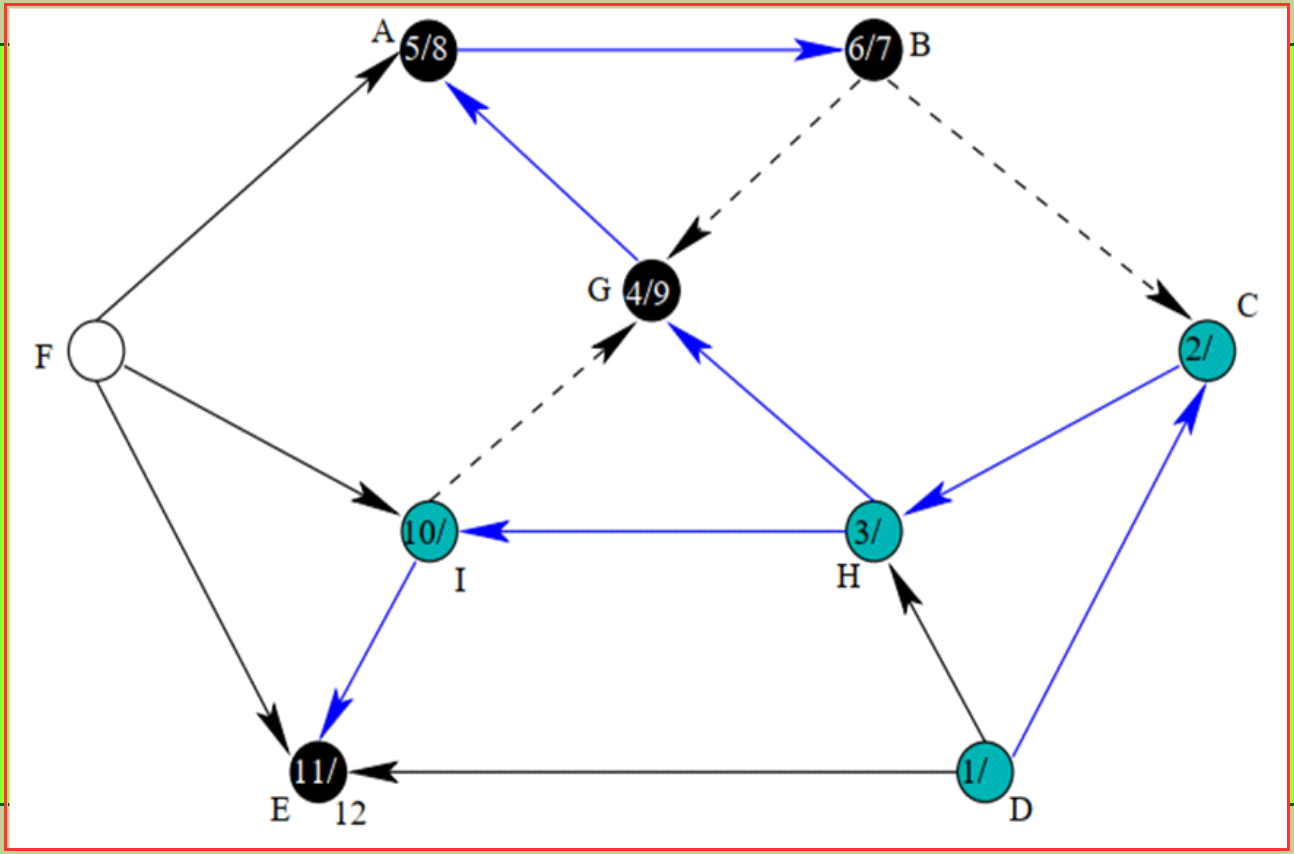
todos os vértices adjacentes de E tratados
Cor[E] = preto
tempo = 12
f[A] = 12




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: recuar ao vértice I (pai de E na APP) - DFS(G, I, 12)

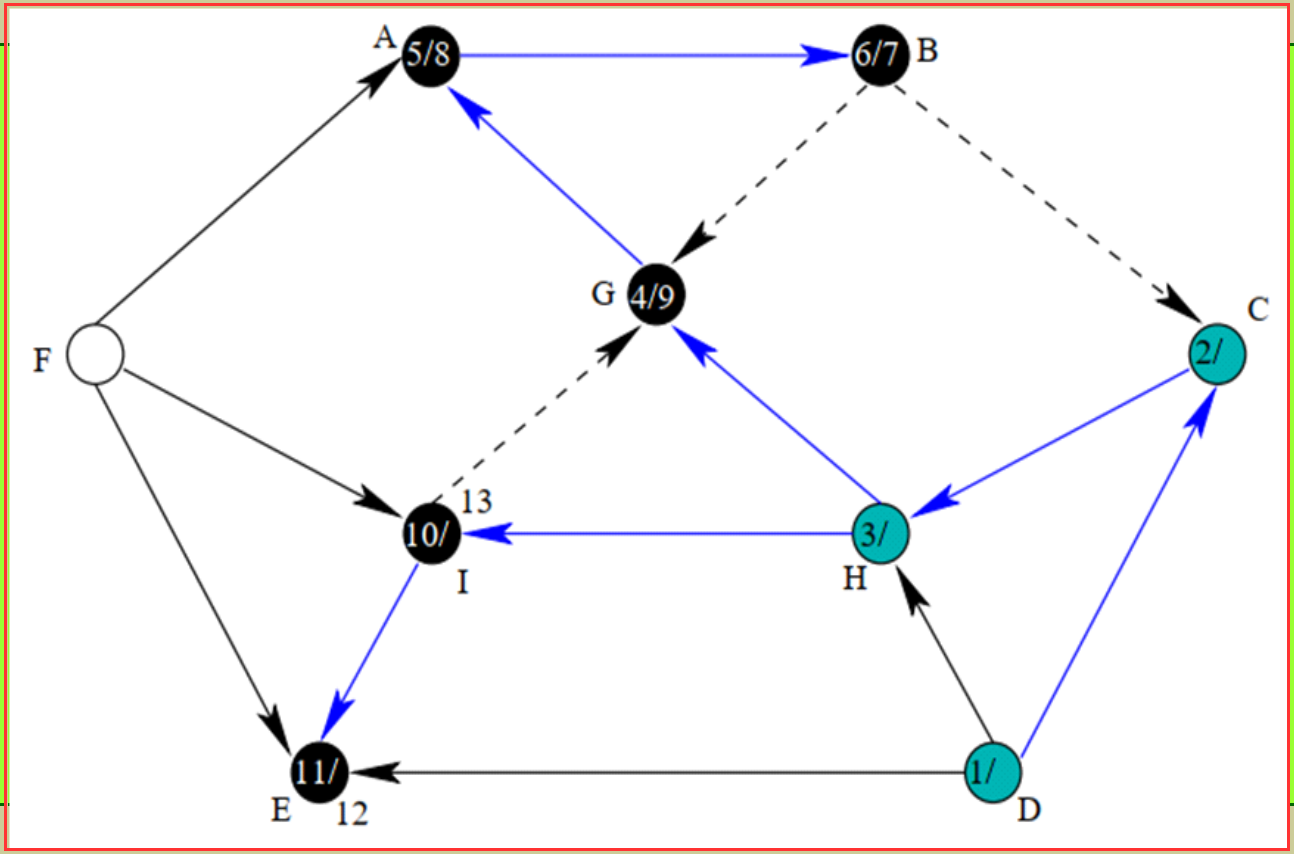
vértice adjacente G (Cor[G] = preto (não branco))
    passar ao próximo adjacente




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: DFS(G, I, 12)

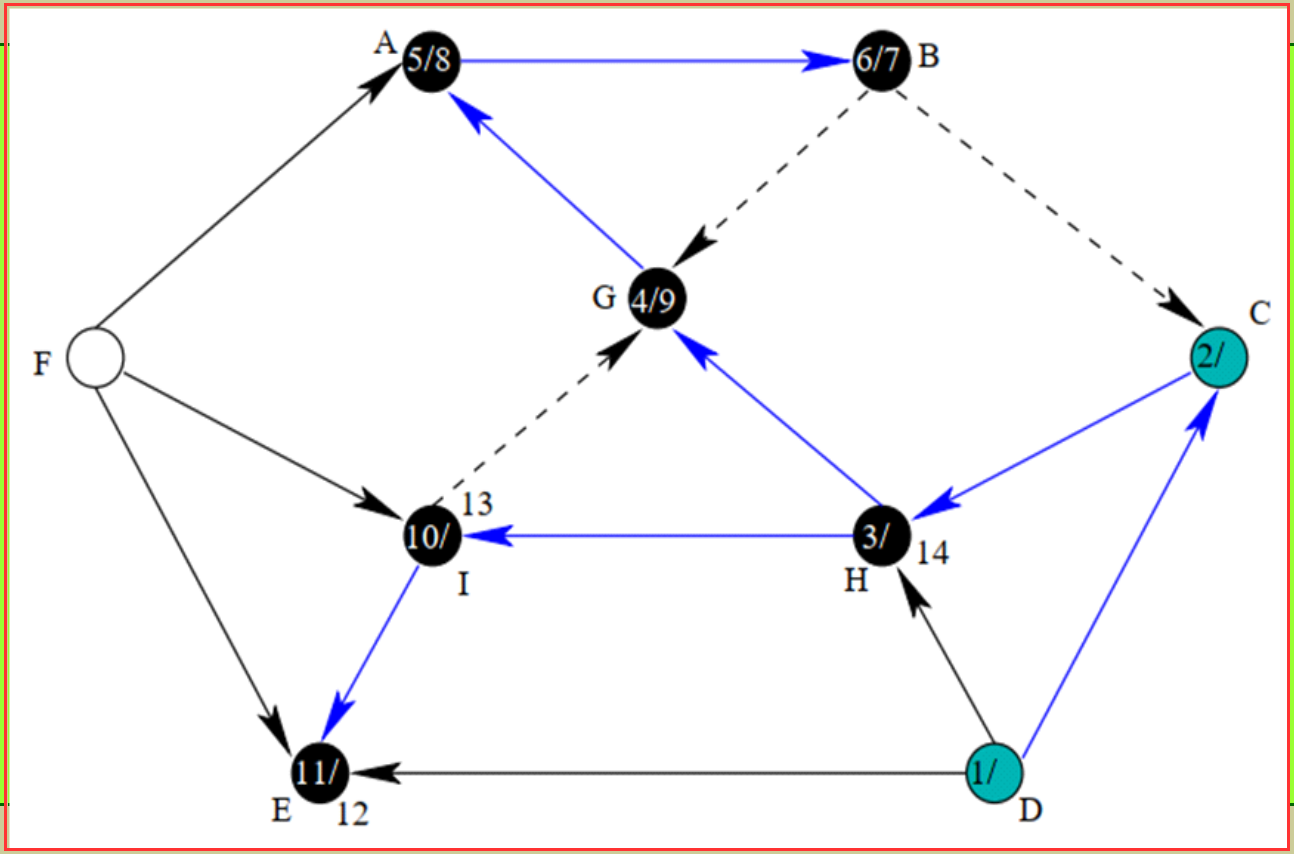
todos os vértices adjacentes de I tratados
Cor[I] = preto
tempo = 13
f[I] = 13




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: recuar ao vértice H (pai de I na APP) - DFS(G, H, 13)

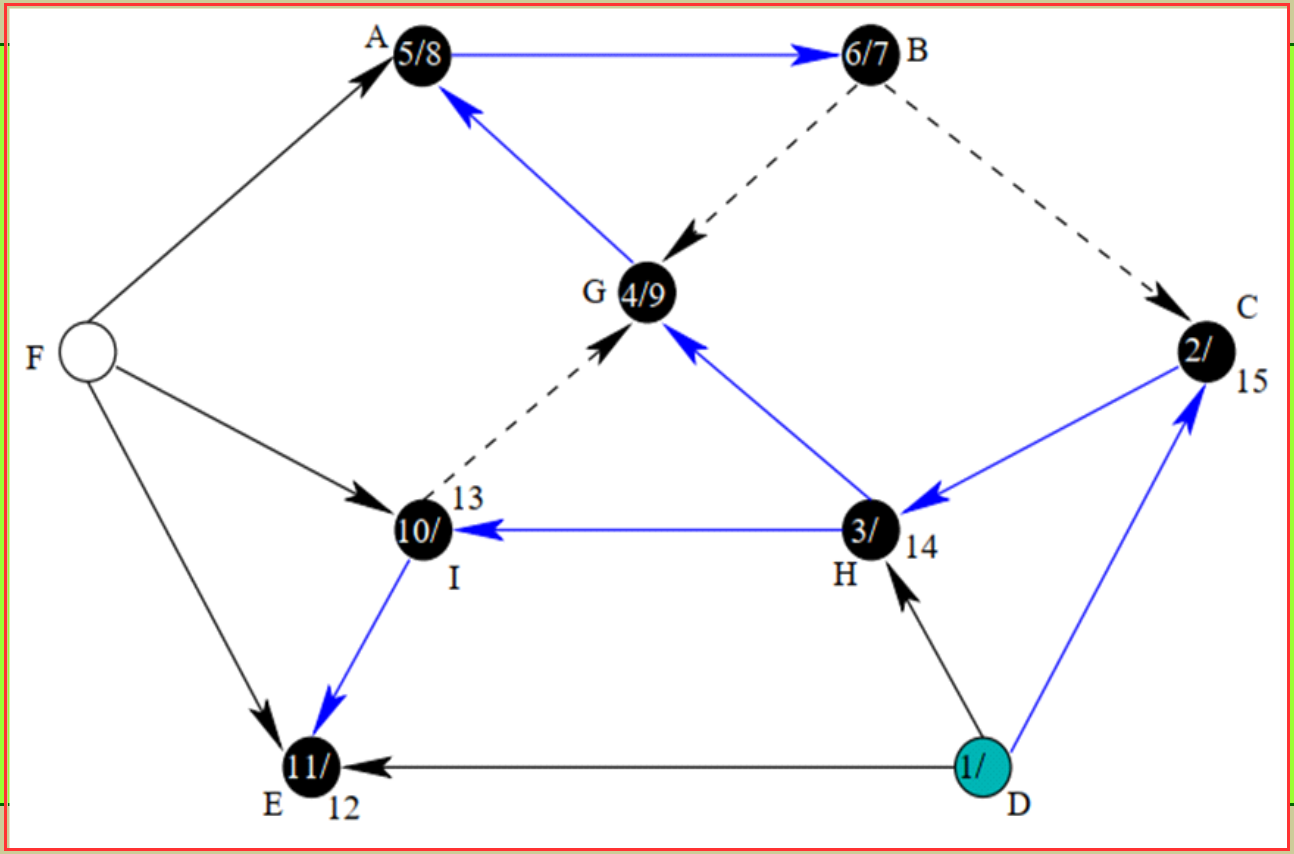
todos os vértices adjacentes de H tratados
Cor[H] = preto
tempo = 14
f[H] = 14




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: recuar ao vértice C (pai de H na APP) - DFS(G, C, 14)

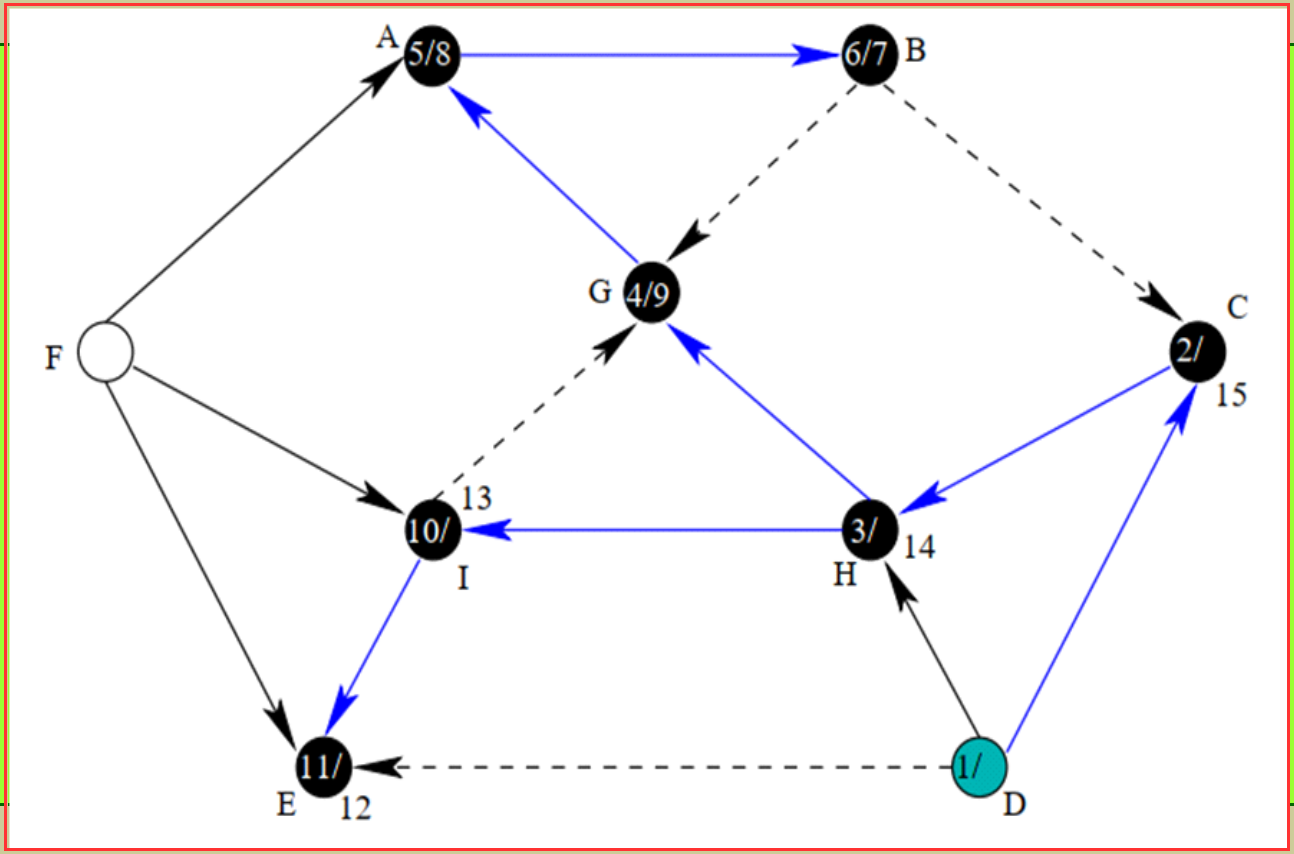
todos os vértices adjacentes de C tratados
Cor[C] = preto
tempo = 15
f[C] = 15




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: recuar ao vértice D (pai de C na APP) - DFS(G, D, 15)

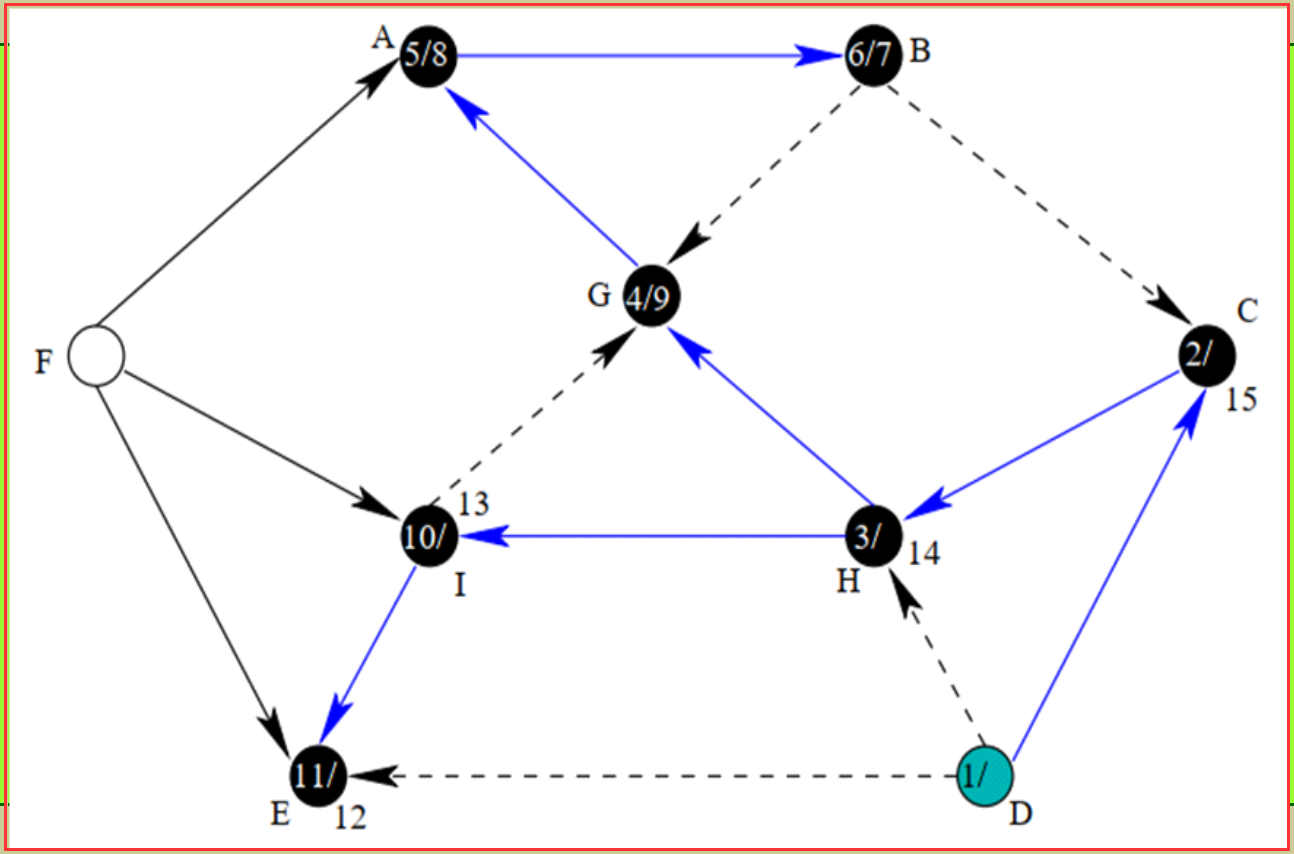
vértice adjacente E (Cor[E] = preto (não branco)):
passar ao próximo adjacente




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: DFS(G, D, 15)

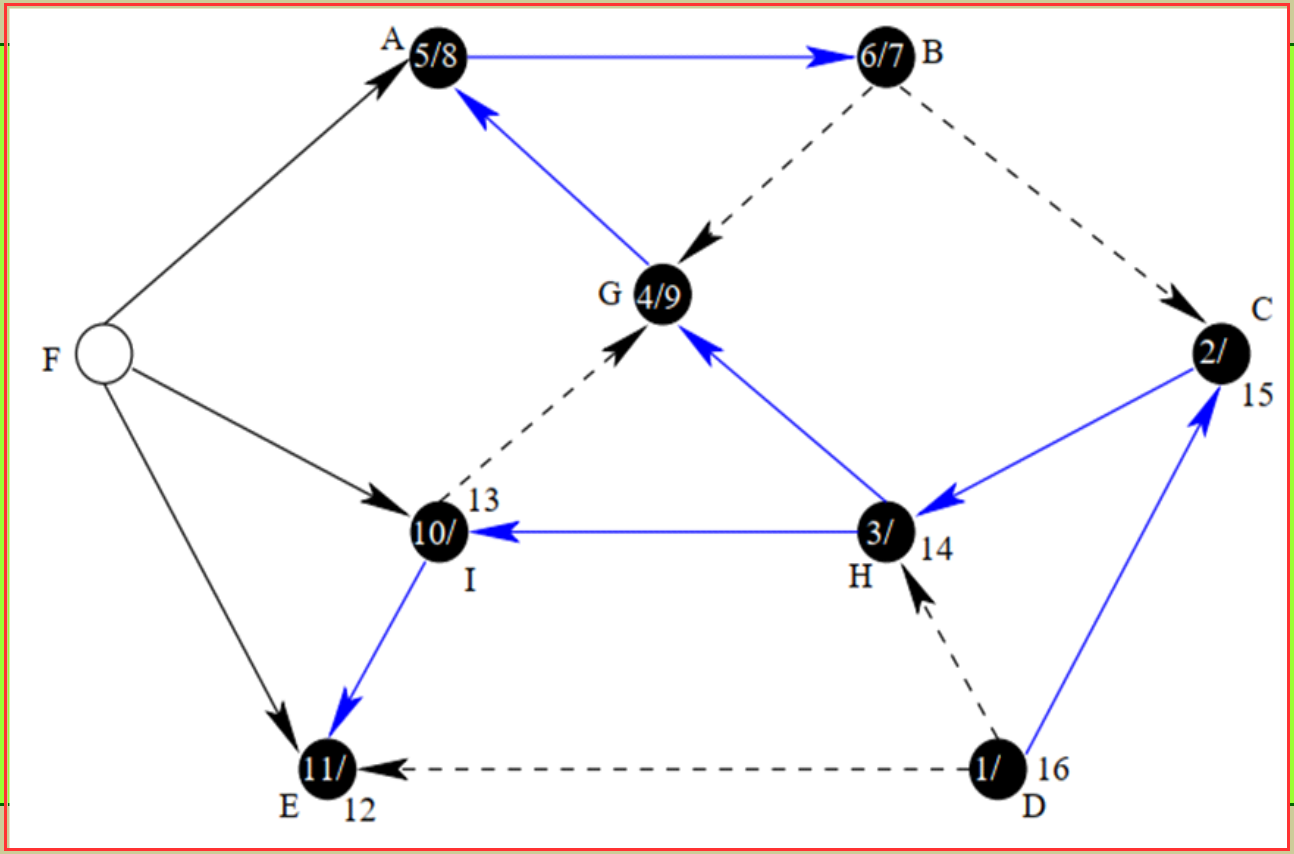
vértice adjacente H (Cor[H] = preto (não branco)):
    passar ao próximo adjacente




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: DFS(G, D, 15)

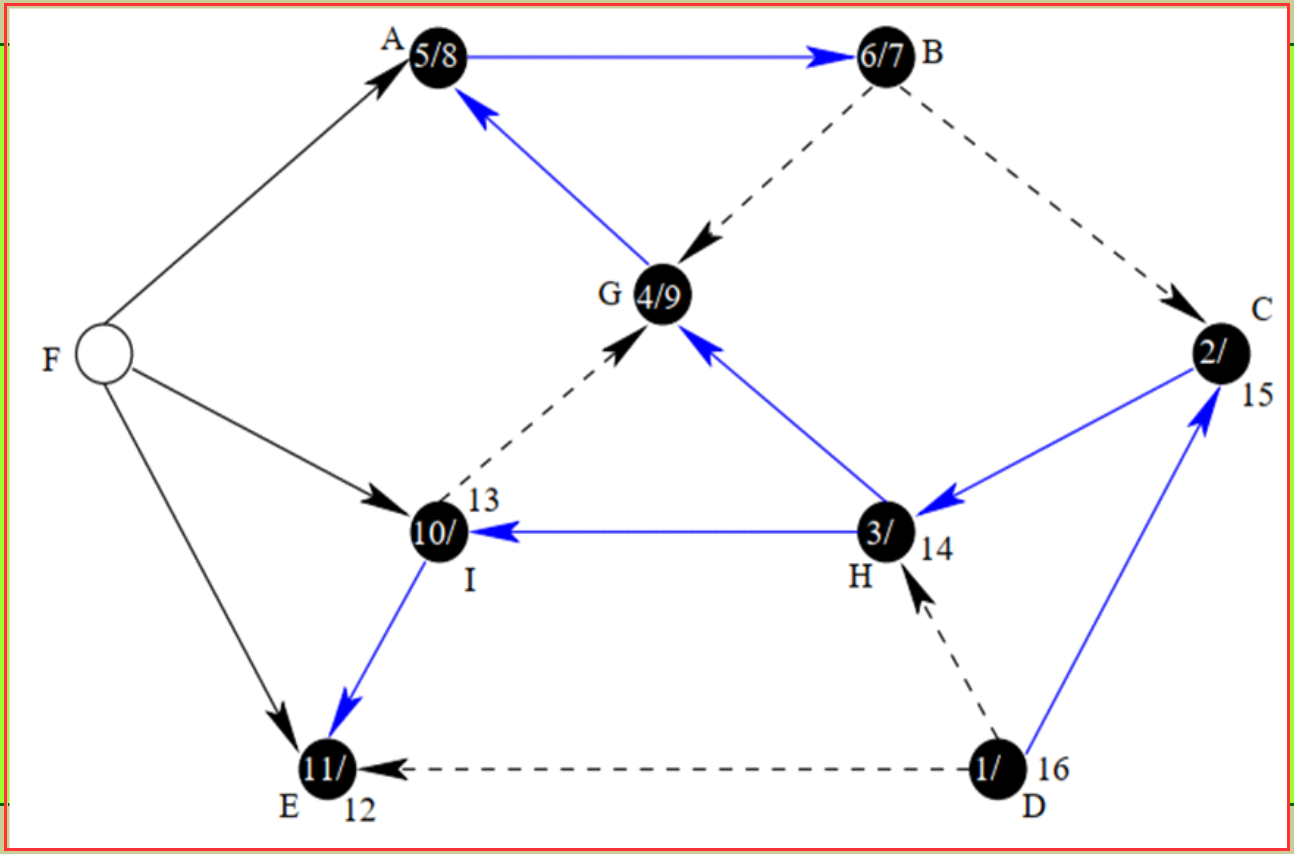
todos os vértices adjacentes de D tratados
Cor[D] = preto
tempo = 16
f[D] = 16




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: CDFS(G, D)

Cor[F] = branco, então
chamada de **DFS(G, F, 16)**



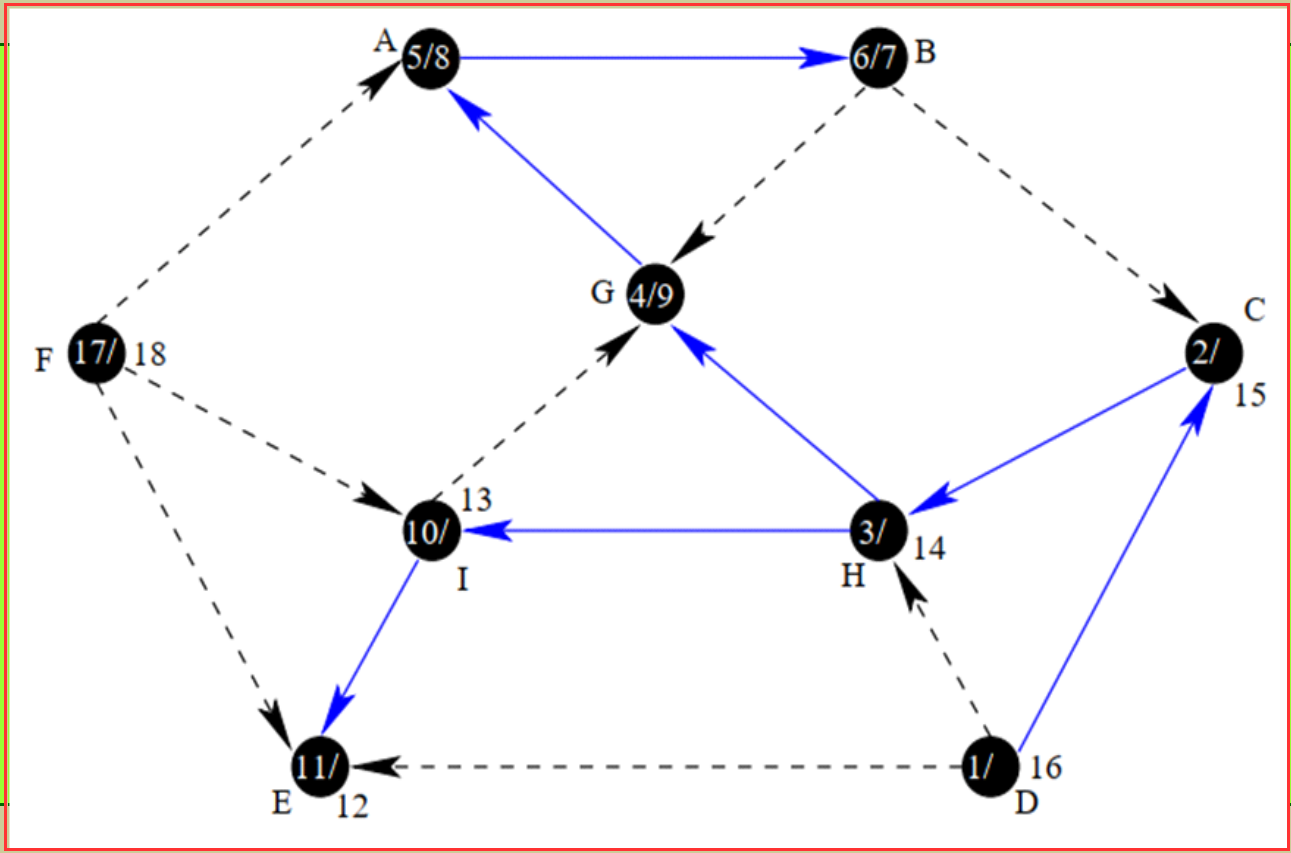

```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: DFS(G, F, 16)

Cor[F] = cinzento; tempo = 17; d[F] = 17

vértices adjacentes A, E e I com Cor = preto (não branco)

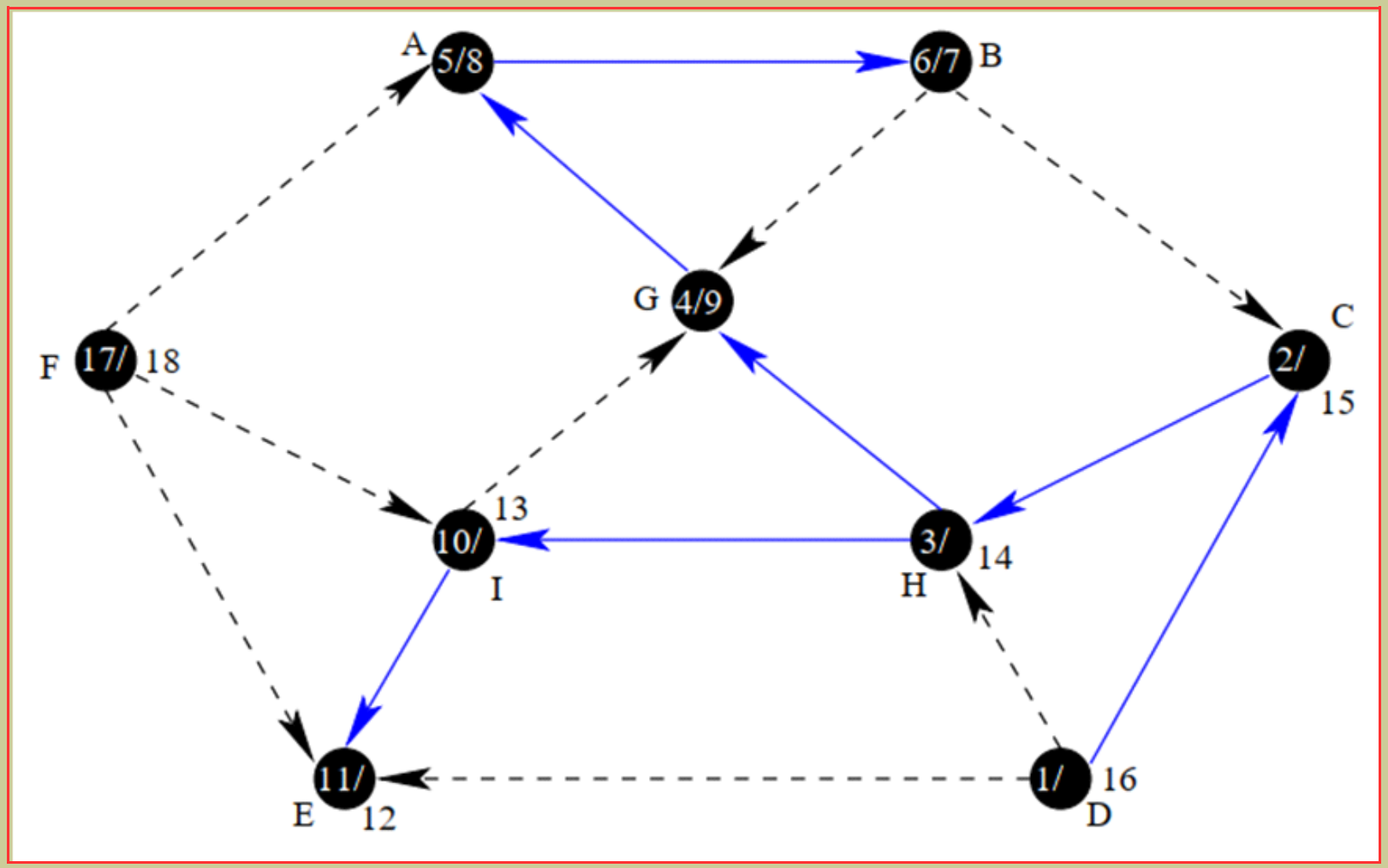
Cor[F] = preto; tempo = 18; f[F] = 18




```
algoritmo DFS(G, S, tempo)
  Cor[S] ← cinzento
  tempo ← tempo + 1
  d[S] ← tempo
  para cada vértice v adjacente a S fazer
    se (Cor[v] = branco) então
      Pai[v] ← S
      DFS(G, v, tempo)
    fim_se
  fim_para
  Cor[S] ← preto
  tempo ← tempo + 1
  f[S] ← tempo
fim_algoritmo
```

Exemplo: ordem das chamadas recursivas

(D, 0) → (C, 1) → (H, 2) → (G, 3) → (A, 4) → (B, 5) → (B, 6) → (A, 7) → (G, 8) → (I, 9) →
→ (E, 10) → (E, 11) → (I, 12) → (H, 13) → (C, 13) → (D, 15) → (F, 16)



Resultados

- Teorema. [Parêntesis]

  Em qualquer pesquisa em profundidade de G = (V, A), tem-se para qualquer par de vértices (**u**, **v**) que uma e uma só das seguintes situações são válidas

  - o intervalo [d[u], f[u]] está contido em [d[v], f[v]] e **u** é descendente de **v** numa APP
  - o intervalo [d[v], f[v]] está contido em [d[u], f[u]] e **v** é descendente de **u** numa APP
  - os dois intervalos são disjuntos e nem **u** é descendente de **v** nem o contrário

- Teorema. [Caminhos Brancos]

  Na floresta de pesquisa em profundidade de G = (V, A), o vértice **v** é um descendente de **u** se e só se no instante d[u] de descoberta de **u**, **v** é alcançável a partir de **u** por um caminho inteiramente constituído por vértices brancos

- Exemplo:

  - B é descendente de G, mas E não o é (apesar de haver um caminho de G para E, este caminho não é inteiramente branco no momento em que G passa a cinzento)