UFPA

Algoritmos de Busca em Grafos

November 18, 2020

Algoritmos de Busca

Problema fundamental em grafos:

Como explorar um grafo de forma sistemática ?

Algoritmos de Busca

Problema fundamental em grafos:

Como explorar um grafo de forma sistemática ?

Utilizando um algoritmo de busca :-)

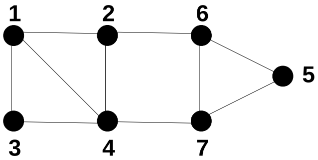
Algoritmos de Busca

Problema fundamental em grafos:

Como explorar um grafo de forma sistemática ?

Utilizando um algoritmo de busca :-)

Como o seguinte grafo pode ser explorado ?



Algoritmos de Busca

Como percorrer o grafo, visitando todos os seus vértices e arestas, e evitando repetições desnecessárias a um mesmo vértice ou aresta ?

Algoritmos de Busca

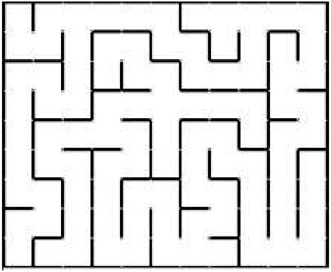
Como percorrer o grafo, visitando todos os seus vértices e arestas, e evitando repetições desnecessárias a um mesmo vértice ou aresta ?

Ideia:

Evitar explorar vértice já explorados $\rightarrow$ marcar os vértices.

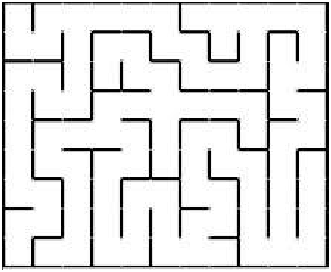
Alg. de Busca em Profundidade (DFS)

Como o seguinte labirinto poderia ser explorado ?



Alg. de Busca em Profundidade (DFS)

Como o seguinte labirinto poderia ser explorado ?

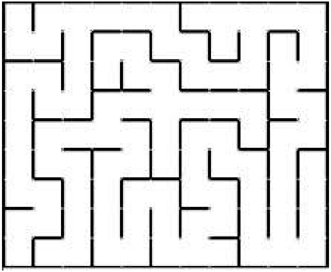


DFS

"Dentre todos os vértices marcados e incidentes a alguma aresta ainda não explorada, escolher aquele mais recentemente alcançado na busca"

Alg. de Busca em Profundidade (DFS)

Como o seguinte labirinto poderia ser explorado ?



DFS

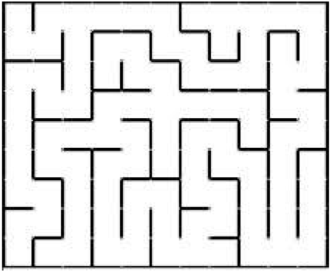
"Dentre todos os vértices marcados e incidentes a alguma aresta ainda não explorada, escolher aquele mais recentemente alcançado na busca"

Ideia:

Explorar o vértice descoberto mais recentemente primeiro.

Alg. de Busca em Profundidade (DFS)

Como o seguinte labirinto poderia ser explorado ?



DFS

"Dentre todos os vértices marcados e incidentes a alguma aresta ainda não explorada, escolher aquele mais recentemente alcançado na busca"

Ideia:

Explorar o vértice descoberto mais recentemente primeiro.

Alg. de Busca em Profundidade (DFS)

Exemplos de Aplicações

- Verificar se existe um percurso de um vértice para outro;
- Verificar se o grafo é conectado ou não;
- . . .

Obs.: Pode ser aplicado a grafos direcionados e não direcionados.

Alg. de Busca em Profundidade (DFS) - Cormem et al.

Particularidades do Algoritimo de Cormem

- $\pi[v]$ - salva o antecessor de um vértice. Sempre que um vértice v for descoberto durante a leitura da lista de adj. de um vértice já descoberto u, $\pi[v] = u$;
- Cada vértice é colorido conforme o seu estado:

  Branco Estado inicial de um vértice;
  Cinza O vértice foi descoberto;
  Preto a lista de adjacências do vértice foi finalizada;

Alg. de Busca em Largura (DFS) - Cormem et al.

Particularidades do Algoritimo de Cormem (cont.)

- $cor[u]$ - mantém a cor do vértice $u \in V$
- Selo de tempo (*timestamp*) - Cada vértice possui dois selos de tempo:

  $d[v]$ - salva o tempo de quando o vértice v foi descoberto e colorido como cinza;

  $f[v]$ - salva o tempo de quando de quando a busca finaliza o exame da lista de adjacências do vértice v e o colore como preto.

Alg. de Busca em Profundidade (DFS) - Cormem et al.

Seja $G = (V, E)$ um grafo:

```
DFS(G)
  // Inicializacao - vertices coloridos de branco;
  //                π[v] iniciado com nulo.
  for cada vertice u ∈ V[G]
     do cor[u] = BRANCO
        π[u] = NULL
  time = 0
  for cada vertice u ∈ V[G]
      do if cor[u] == BRANCO
         then DFS-VISIT(u)
```

Alg. de Busca em Profundidade (DFS) - Cormem et al.

```
DFS-VISIT(u)
  cor[u] = CINZA
  tempo = tempo + 1
  d[u] = tempo
  for cada v ∈ Adj[u] // Explorar a aresta (u,v)
      do if cor[v] = BRANCO
         then π[v] = u // O antecessor de 'v' = 'u'
              DFS-VISIT(v)
  cor[u] = PRETO // vertice 'u' finalizado
  f[u] = tempo = tempo + 1
```

Alg. de Busca em Profundidade (DFS) - Cormem et al.

```
DFS-VISIT(u)
  cor[u] = CINZA
  tempo = tempo + 1
  d[u] = tempo
  for cada v ∈ Adj[u] // Explorar a aresta (u,v)
      do if cor[v] = BRANCO
          then π[v] = u // O antecessor de 'v' = 'u'
              DFS-VISIT(v)
  cor[u] = PRETO // vertice 'u' finalizado
  f[u] = tempo = tempo + 1
```

Obs. 1: Esse algoritmo foi originalmente proposto para um grafo direcionado, mas pode ser utilizado em grafos não direcionados (Verifique).
Obs. 2: A ordem em que as arestas/vértices são visitados depende da representação do grafo.

Complexidade

- Laços DFS(G): [for cada vertice $u \in V[G]$] $\rightarrow \Theta(V)$

Complexidade

- Laços DFS(G): [for cada vertice $u \in V[G]$] $\rightarrow \Theta(V)$
- DFS-Visit(u): executado uma vez para cada vértice (branco). Laço [for cada $v \in Adj[u]$ do] executado $|Adj[v]|$ vezes $\rightarrow \sum_{v \in V} |Adj[v]| = \Theta(E)$

Complexidade

- Laços DFS(G): [for cada vertice $u \in V[G]$] $\rightarrow \Theta(V)$
- DFS-Visit(u): executado uma vez para cada vértice (branco). Laço [for cada $v \in Adj[u]$ do] executado $|Adj[v]|$ vezes $\rightarrow \sum_{v \in V} |Adj[v]| = \Theta(E)$
- Tempo de Execução do DFS: $\Theta(V + E)$

Classificação das Arestas

Seja uma aresta (u, v):

Aresta de Árvore São arestas que estão na floresta DFS. A aresta (u, v) é uma aresta de árvore se v foi primeiro descoberto explorando a aresta (u, v);

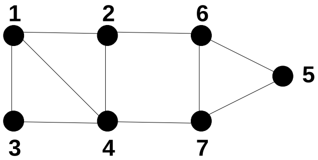
Aresta de Retorno Aresta conectando o vértices u ao ancestral v na floresta DFS. Laços são considerados arestas de retorno;

Aresta de Avanço São arestas que não fazem parte da floresta, mas conectam o vértice u ao descendente v na floresta DFS;

Aresta de Cruzamento São arestas que conectam vértices na mesma árvore DFS em que um vértice não é ancestral de outro OU conecta vértices em diferentes árvores DFS

Exemplo

Vamos empregar o alg. DFS no seguinte grafo, iniciando pelo vértice 2



Uma implementação do Alg. DFS (Sedgewick)

```
public class BuscaEmProfundidade {
  private boolean[] marcado;
  private int cont;
  public BuscaEmProfundidade(Grafo G, int s) {
     marcado = new boolean[G.V()];
     dfs(G,s);
  }
  private void dfs(Grafo G, int v) {
     marcado[v] = true;
     cont++;
     for (int w: G.adj())
          if (!marcado[w]) dfs(G,w);
  }
  public boolean marcado(int w) {
     return marcado[w];
  }
  public int cont() {
     return cont;
} }
```

Trabalho

Adapte e implemente o alg. DFS de Sedgewick para funcionar de acordo com o algoritmo de Cormem et al.

Alg. de Busca em Largura (BFS)

BFS

"Dentre todos os vértices marcados e incidentes a alguma aresta não explorada, escolher aquele *menos recentemente* alcançado na busca."

Alg. de Busca em Largura (BFS)

BFS

"Dentre todos os vértices marcados e incidentes a alguma aresta não explorada, escolher aquele *menos recentemente* alcançado na busca."

Ideia:

A partir de um vértice s, explorar todos os vértices a partir de s.

Alg. de Busca em Largura (BFS)

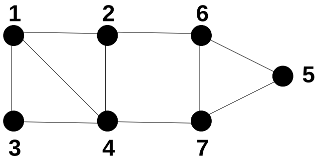
BFS

"Dentre todos os vértices marcados e incidentes a alguma aresta não explorada, escolher aquele *menos recentemente* alcançado na busca."

Ideia:

A partir de um vértice s, explorar todos os vértices a partir de s.

Como o seguinte grafo pode ser explorado ?



Alg. de Busca em Largura (BFS)

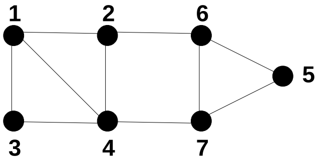
BFS

"Dentre todos os vértices marcados e incidentes a alguma aresta não explorada, escolher aquele *menos recentemente* alcançado na busca."

Ideia:

A partir de um vértice s, explorar todos os vértices a partir de s.

Como o seguinte grafo pode ser explorado ?



A expansão assim é realizada em "camadas".

Obs.: Funciona em grafos direcionados e não direcionados.

Alg. de Busca em Largura (BFS) - Cormem et al.

Particularidades do Algoritimo de Cormem

- Seja um grafo $G = (V, E)$ e um vértice (fonte) s;
- $\pi[u]$ é um vetor utilizado para salvar o antecessor de um vértice u. Se o vértice u não tem antecessor, $\pi[u] = nulo$;
- Cada vértice é colorido conforme o seu estados:

  Branco Estado inicial de um vértice;
  Cinza O vértice foi descoberto;
  Preto As adjacências do vértice foram finalizada;

- Se $(u, v) \in E$ e o vértice $u = preto$, então o vértice $v =$ cinza or preto
- Vértices cinza podem ter vértices brancos adjacentes; eles representam uma fronteira entre vértices descobertos e não descobertos.

Alg. de Busca em Largura (BFS) - Cormem et al.

Particularidades do Algoritimo de Cormem (cont.)

- $cor[u]$ - mantém a cor do vértice $u \in V$;
- $d[v]$ - mantém a distância entre o vértice fonte s e o vértice u;
- Fila Q para gerenciar o conjunto dos vértices cinzas.

Alg. de Busca em Largura (BFS) - Cormem et al.

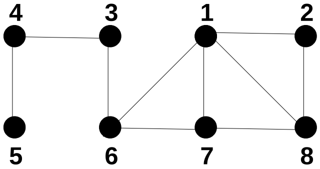
```
BFS(G,s)
  for cada u ∈ V[G] - {s} //Todos exceto s=fonte
      do cor[u] = BRANCO
         d[u] = ∞
         π[u] = NULO
  cor[s] = CINZA
  d[s] = 0
  π[s] = NULO // vertice s nao possui antecessor
  Q = ∅
  ENFILEIRAR(Q,s) // inicializa a fila Q com o vert. s
```

Alg. de Busca em Largura (BFS) (cont.)

```
while (Q ≠ 0 )
   do u = DESENFILEIRAR(Q)
      for cada v ∈ Adj[u] // Explorar (u,v)
          do If cor[v] = BRANCO
               then cor[v] = CINZA
                    d[v] = d[u] + 1
                    π[v] = u
                    ENFILEIRAR(Q,v)
     cor[u] = PRETO
```

Alg. de Busca em Largura (BFS)

Exemplo:



Uma implementação do Alg. BFS (Sedgewick)

```
public class BuscaEmLargura {
  private boolean[] marcado;
  public BuscaEmLargura(Grafo G, int s) {
     marcado = new boolean[G.V()];
     bfs(G,s);
  }
  private void bfs(Grafo G, int s) {
     Queue<Integer> queue = new Queue<Integer>();
     marcado[s] = true;
     queue.enqueue(s);
     while( !q.vazio() ) {
        int v = queue.dequeue();
        for (int w: G.adj(v))
          if ( !marcado[w] ) {
             marcado[w] = true;
             queue.enqueue(w);
  } } } }
  public boolean marcado(int w) {
     return marcado[w];
```

Trabalho

Adapte e implemente o alg. BFS de Sedgewick para funcionar de acordo com o algoritmo de Cormem et al.