Grafos [Buscas e Conexidade]

Referência

Buscar um grafo significa visitar seus vértices, percorrendo sistematicamente suas arestas.

Busca em Largura (BFS - Breadth First Search)

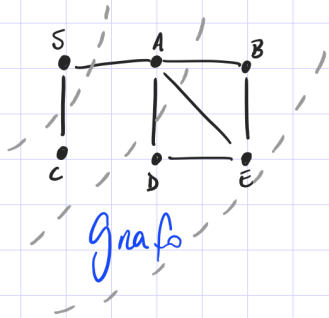
Explora o grafo a partir de um vértice inicial s arbitrário, visitando vértices mais próximos de s antes antes de vértices mais distantes de s.

↳ Considerando o número de arestas como medida de distância.

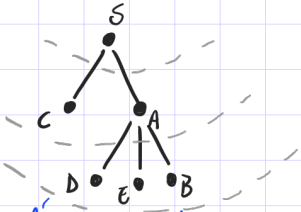
↳ Consequência: Os vértices são visitados pelo caminho mais curto a partir da origem.

↳ Esses caminhos formam a árvore de busca.

Busca por camadas



Grafo



Árvore da busca em largura

Vetor de Roteamento: (vetor de predecessores)

Vértice →	v	s	A	B	C	D	E
Predecessor →	p(v)	∅	S	A	S	A	A

↳ Forma de representar/armazenar a árvore de busca (caminhos)

BFS $(G=(V,E), s)$:

PARA CADA $v \in V$ FAÇA:
- Visitado$(v) \leftarrow$ FALSO
- $p(v) \leftarrow \emptyset$
- $d(v) \leftarrow \infty$

$O(n)$

$d(s) \leftarrow 0$

Visitado$(s) \leftarrow$ VERDADEIRO

$Q \leftarrow \emptyset$

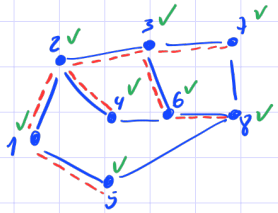
Q.enqueue(s)

ENQUANTO $Q \neq \emptyset$ FAÇA
- $u \leftarrow$ Q.DEQUEUE()
- PARA CADA $v \in$ ADJ(u) FAÇA
  - SE !Visitado(v) ENTÃO
    - Q.Enqueue(v)
    - Visitado$(v) \leftarrow$ VERDADEIRO
    - $p(v) \leftarrow u$
    - $d(v) \leftarrow d(u)+1$

$2m$ vezes $\sim O(m)$

Total: $O(n+m)$

EXEMPLO:



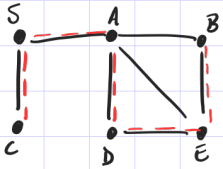
$s=3$

	1	2	3	4	5	6	7	8
Visitado	T	T	T	T	T	T	T	T
p	2	3	$\emptyset$	2	1	3	3	6
d	2	1	0	2	3	1	1	2

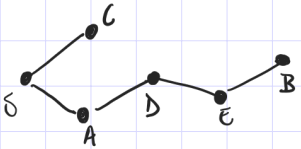
Busca em Profundidade (DFS - Depth First Search)

Explora o grafo procurando sempre entrar mais profundamente na sua estrutura, i.e. tenta aprofundar camadas, antes de visitar vértices da mesma camada.

↳ BACKTRACKING: retorna após não poder mais aprofundar.



Grafo



Árvore da busca em Profundidade

v	S	A	B	C	D	E
p(v)	$\emptyset$	S	E	S	A	D

Vetor de Roteamento

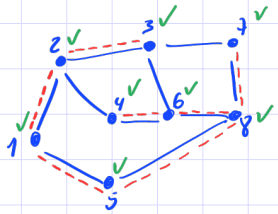
DFS $(G=(V,E), s)$:

PARA cada $v \in V$ FAÇA
- Visitado$(v) \leftarrow$ FALSO
- $p(v) \leftarrow \emptyset$

Visita (s)

$O(n)$

EXEMPLO:



$\boxed{s=3}$

	1	2	3	4	5	6	7	8
Visitado	T	T	T	T	T	T	T	T
p	2	3	$\emptyset$	6	1	8	8	5

Visita (u):

Visitado$(u) \leftarrow$ VERDADEIRO

PARA CADA $v \in$ Adj(u) FAÇA
- SE !Visitado(v) ENTÃO
  - $p(v) \leftarrow u$
  - Visita (v)

$2m$ vezes $\sim O(m)$

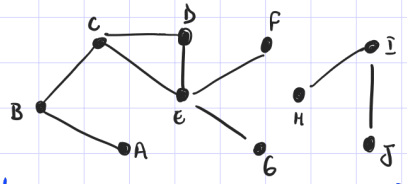
Total: $O(n+m)$

CONEXIDADE

GRAFO CONEXO (CONECTADO): Um grafo G é conexo se, para todo par de vértices u e v do grafo, existe pelo menos um caminho de u para v. Caso contrário, o grafo é não conexo.

COMPONENTE CONEXA: Um subgrafo conexo.

↳ Componente conexa de s: Conjunto de todos os vértices alcançáveis a partir de s.



GRAFO G

→ G é não conexo

→ G possui duas componentes conexas: $\{A,B,C,D,E,F,G\}$, $\{H,I,J\}$

→ Componente conexa do I: $\{H,I,J\}$

Para Dígrafos

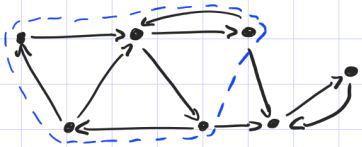
Grafo fracamente conexo: Um dígrafo G é dito fracamente conexo quando seu grafo subjacente é conexo.

Grafo fortemente conexo: Um dígrafo G é dito fortemente conexo quando, para todo par u, v de vértices de G, existe um caminho de u para v e um caminho de v para u.

Componente fortemente conexa: Subgrafo maximal fortemente conexo.

Dígrafo fortemente conexo

Dígrafo fracamente conexo



Componente fortemente conexa

Para encontrar o componente conexo de s:
- Busca (largura/profundidade) a partir de s. $O(n+m)$

Para encontrar todos os componentes conexos:
- Repete a busca enquanto há vértices não marcados. $O(n+m)$

Para encontrar o maior componente conexo:
- Encontra todos, contando os vértices e armazenando o maior. $O(n+m)$