Busca em largura

Algoritmos em Grafos

Marco A L Barbosa

Conteúdo

O estudo utilizando apenas este material **não é suficiente** para o entendimento do conteúdo. Recomendamos a leitura das referências no final deste material e a resolução (por parte do aluno) de todos os exercícios indicados.

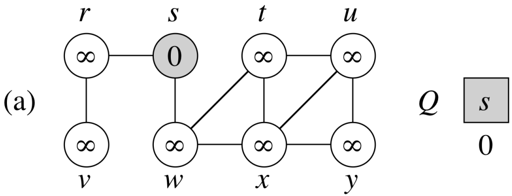
Introdução

Introdução

- Dado um grafo $G = (V, E)$ e um vértice de origem s, a busca em largura explora sistematicamente as arestas de G até descobrir cada vértice acessível a partir de s
- O algoritmo calcula a distância (menor número de arestas) deste s até todos os vértices acessíveis a partir de s
- O algoritmo produz uma árvore primeiro em extensão
- Recebe este nome porque expande a fronteira entre vértices descobertos e não descobertos uniformemente ao longo da extensão da fronteira. Descobre todos os vértices de distância k de s antes de descobrir quaisquer vértices de distância $k + 1$

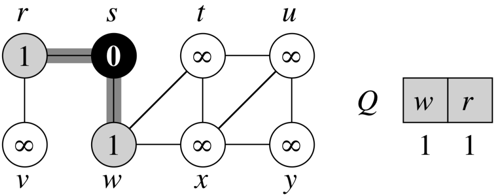
Exemplo de execução

Exemplo de execução

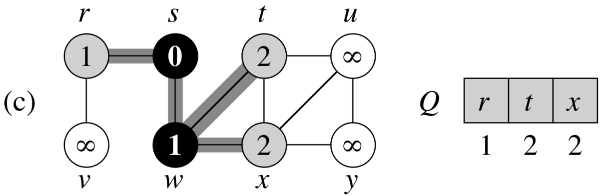


Exemplo de execução

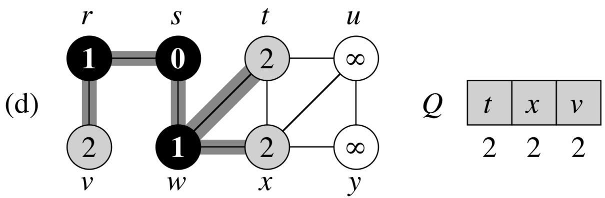
(b)



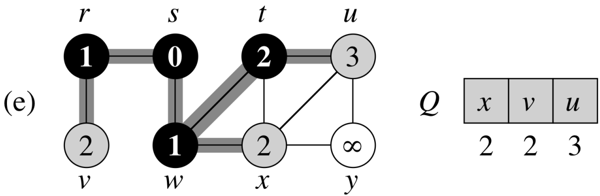
Exemplo de execução



Exemplo de execução

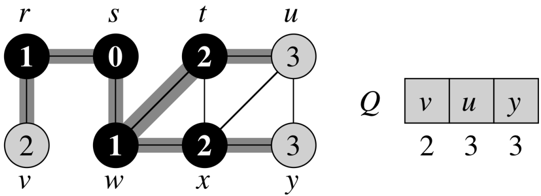


Exemplo de execução

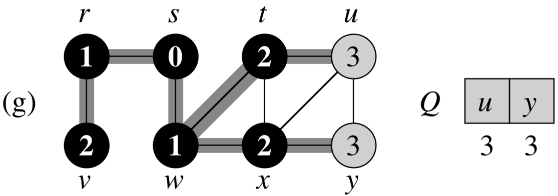


Exemplo de execução

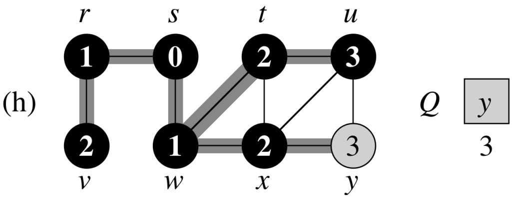
(f)



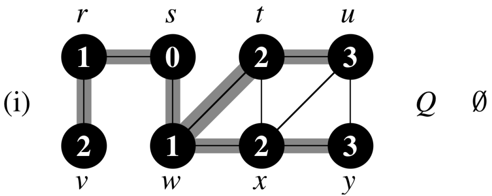
Exemplo de execução



Exemplo de execução



Exemplo de execução



Procedimento `bfs`

Procedimento bfs

```
bfs(G, s)
for cada vértice u em G.V - {s}
  u.d = infinito
  u.pai = nil
  u.cor = branco
s.d = 0
s.pai = nil
s.cor = cinza
Q = {}
Q.add(s)
while Q != {}
  u = Q.remove()
  for cada vértice v em G.adj[u]
    if v.cor == branco
      v.d = u.d + 1
      v.pai = u
      v.cor = cinza
      Q.add(v)
  u.cor = preto
```

Análise do tempo de execução do `bfs`

Análise do tempo de execução do `bfs`

- Análise agregada

Análise do tempo de execução do `bfs`

- Análise agregada
- O teste da linha 13 garante que cada vértice é colocado na fila no máximo uma vez, e portanto, é retirado da fila no máximo uma vez
- As operações de colocar e retirar da fila demoram $O(1)$, assim, o tempo total das operações com filas é $O(V)$
- A lista de adjacência de cada vértice é examinada apenas quando o vértice é retirado da fila, desta forma, no máximo uma vez
- Como a soma dos comprimentos das listas de adjacências é $\Theta(E)$, o tempo para percorrer todas as listas é no máximo $O(E)$
- O tempo de inicialização é $O(V)$
- Tempo total de execução do `bfs` é $O(V + E)$

Árvore primeiro na extensão

Árvore primeiro na extensão

- `bfs` constrói uma árvore primeiro na extensão
- A árvore é definida pelo campo pai (π) em cada vértice
- Para um grafo $G = (V, E)$ e um vértice de origem s, definimos o **subgrafo predecessor** de G como $G_\pi = (V_\pi, E_\pi)$ onde
  - $V_\pi = \{v \in V : v.\pi \neq \text{NIL}\} \cup \{s\}$
  - $E_\pi = \{(v.\pi, v) : v \in V_\pi - \{s\}\}$

Árvore primeiro na extensão

- `bfs` constrói uma árvore primeiro na extensão
- A árvore é definida pelo campo pai (π) em cada vértice
- Para um grafo $G = (V, E)$ e um vértice de origem s, definimos o **subgrafo predecessor** de G como $G_\pi = (V_\pi, E_\pi)$ onde
  - $V_\pi = \{v \in V : v.\pi \neq \text{NIL}\} \cup \{s\}$
  - $E_\pi = \{(v.\pi, v) : v \in V_\pi - \{s\}\}$
- O subgrafo predecessor G_π é uma árvore primeiro na extensão
  - V_π consiste nos vértices acessíveis a partir de s
  - Para todo $v \in V_\pi$, existe um caminho único simples desde s até v em G_π, que também é um caminho mais curto de s até v em G
- Uma árvore primeiro na extensão é de fato uma árvore, pois é conexa e $|E_\pi| = |V_\pi| - 1$

Árvores primeiro na extensão

```
imprimir-caminho(G, s, v)
if v == s
  imprimir s
else if v.pai == nil
  imprimir "nenhum caminho existente de" s "para" v
else
  imprimir-caminho(G, s, v.pai)
  imprimir v
```

- Executado em tempo

Árvores primeiro na extensão

```
imprimir-caminho(G, s, v)
if v == s
  imprimir s
else if v.pai == nil
  imprimir "nenhum caminho existente de" s "para" v
else
  imprimir-caminho(G, s, v.pai)
  imprimir v
```

- Executado em tempo linear no número de vértices no caminho impresso, pois cada chamada recursiva é feita para um caminho com um vértice menor que o atual

Referências

Referências

- Thomas H. Cormen et al. Introduction to Algorithms. 3rd edition. Capítulo 22.2.