Busca em largura

Marco A L Barbosa
malbarbo.pro.br

Departamento de Informática
Universidade Estadual de Maringá

Introdução

- Dado um grafo $G = (V, E)$ e um vértice de origem s, a **busca em largura** (*Breadth-first search* - BFS, em inglês) explora sistematicamente as arestas de G até descobrir cada vértice acessível a partir de s
  - Calcula a distância (menor número de arestas) de s até todos os vértices acessíveis a partir de s
  - Produz uma árvore de busca em largura
  - É a base para outros algoritmos

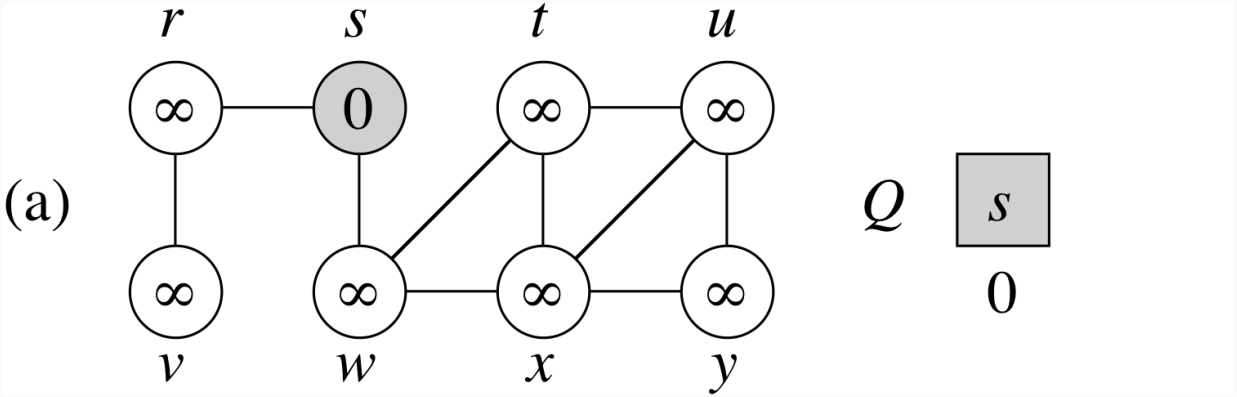
Introdução

- 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 $k + 1$

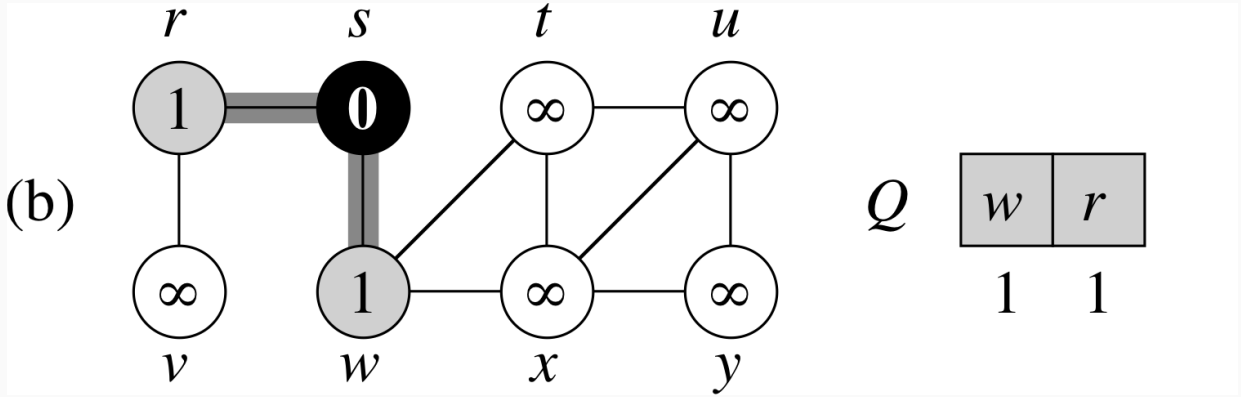
Introdução

- Durante a execução do algoritmo, diversos atributos nos vértices são utilizados
  - Cor (*cor*): cada vértice inicialmente é branco. Quando um vértice é **descoberto** (encontrado pela primeira vez na busca) ele é colorido de cinza. Quando a lista de adjacências de um vértice é totalmente explorada, o vértice é colorido de preto.
  - Pai (π): quando um vértice v é descoberto através da aresta (u, v), dizemos que o vértice u é **predecessor** ou **pai** de v. O atributo π é utilizado para armazenar o predecessor de cada vértice.
  - Distância (d): a distância em relação ao vértice inicial da busca.

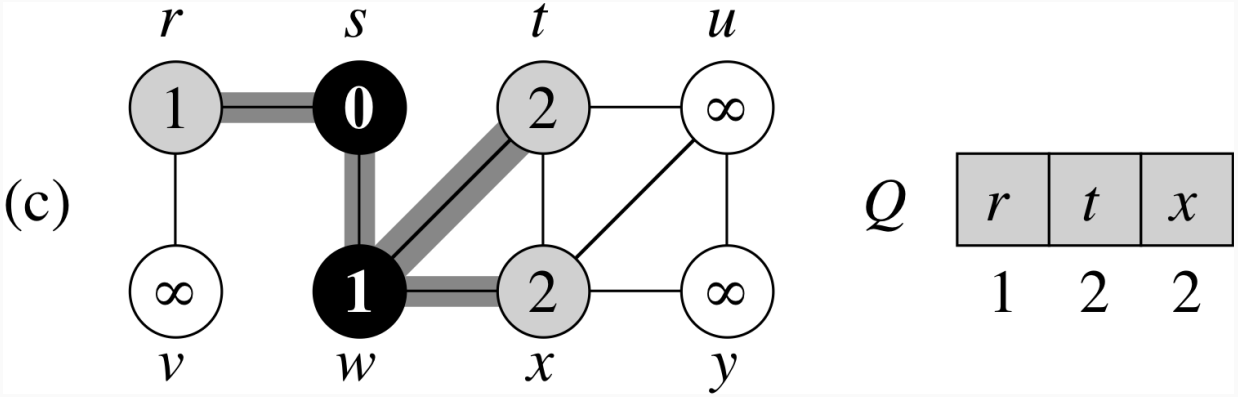
Exemplo de execução



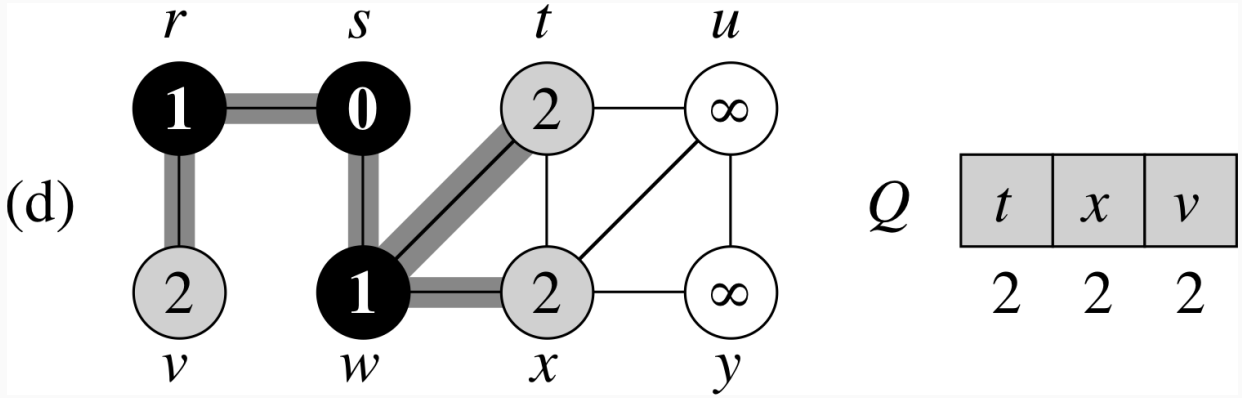
Exemplo de execução



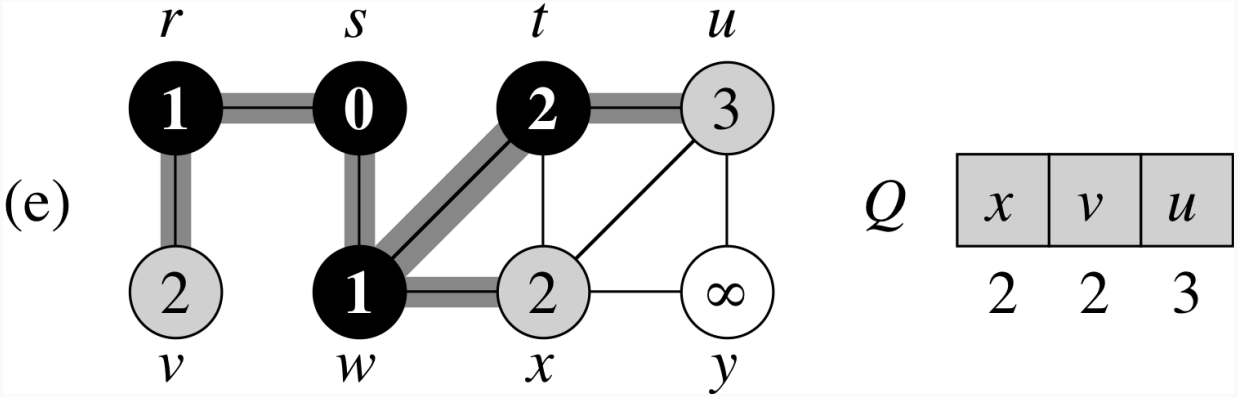
Exemplo de execução



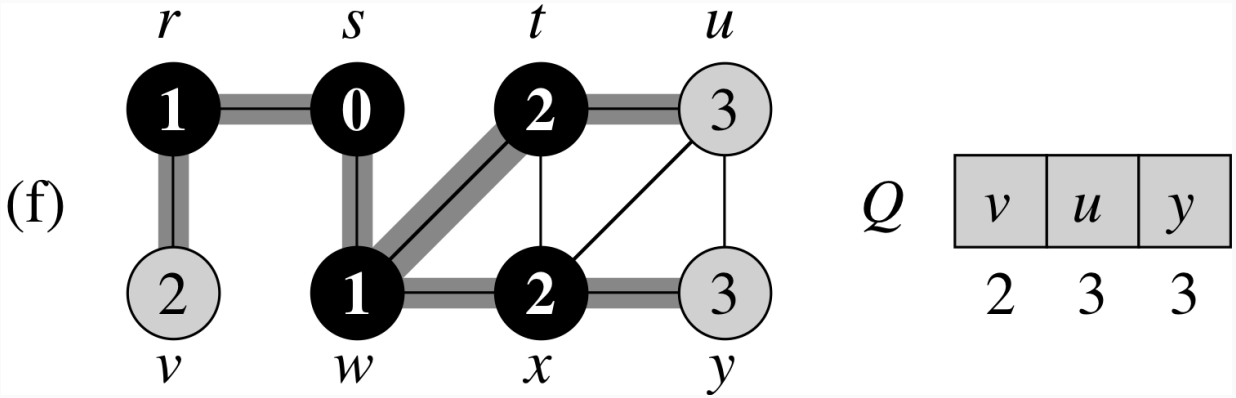
Exemplo de execução



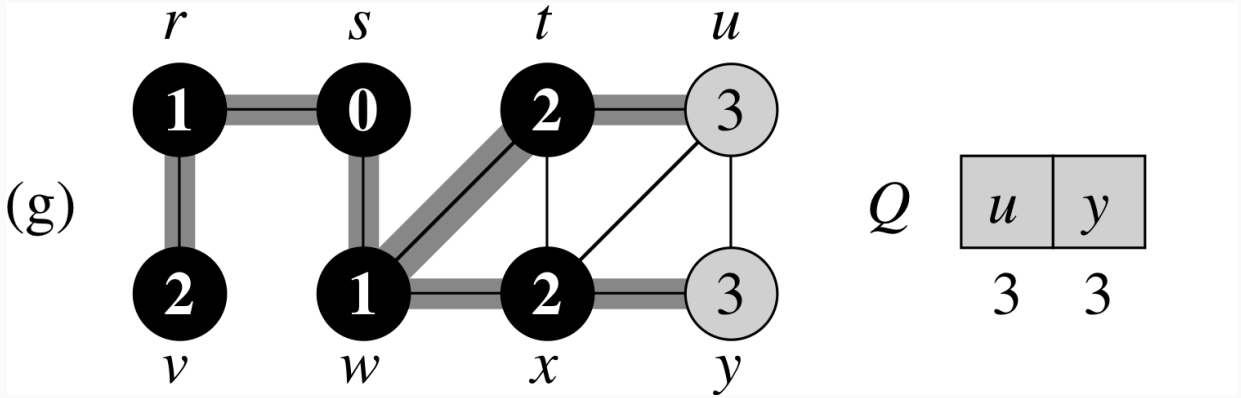
Exemplo de execução



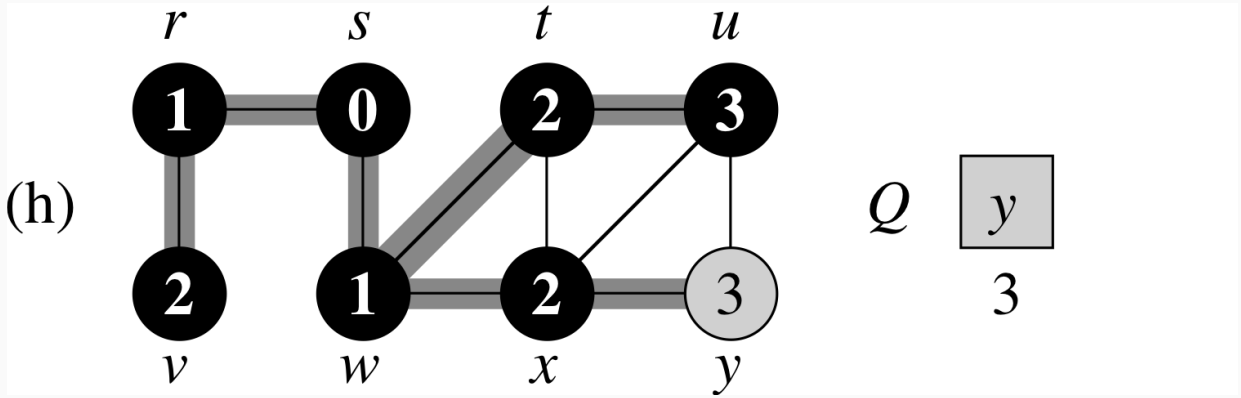
Exemplo de execução



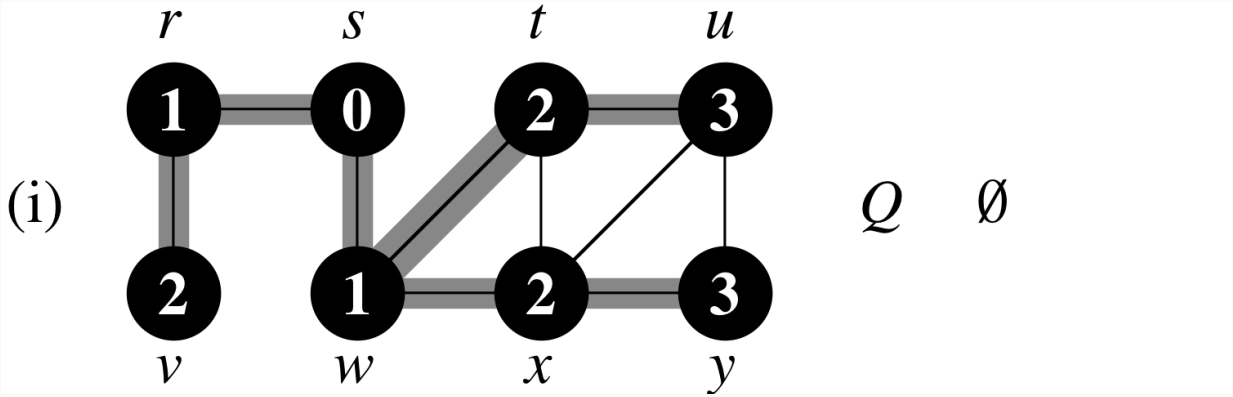
Exemplo de execução



Exemplo de execução



Exemplo de execução



Procedimento BFS

```
BFS(G, s)
 1  for v ∈ G.V − {s}
 2      v.d = ∞
 3      v.π = NIL
 4      v.cor = BRANCO
 5  s.d = 0
 6  s.π = NIL
 7  s.cor = CINZA
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for v ∈ G.Adj[u]
13          if v.cor == BRANCO
14              v.d = u.d + 1
15              v.π = u
16              v.cor = CINZA
17              ENQUEUE(Q, v)
18      u.cor = PRETO
```

Tempo de execução

- 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 tem tempo $O(1)$, 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 $O(V + E)$

Corretude

- Afirmamos no início dos slides que o procedimento BFS encontra as distâncias de s para todos os vértices acessíveis a partir de s. Vamos ver porque isto é verdade.
- Definimos como a **distância do menor caminho** $\delta(s, v)$ de s para v como o número mínimo de arestas de qualquer caminho de s para v, se não existe caminho de s para v então $\delta(s, v) = \infty$.
- Precisamos mostrar que o valor $v.d$ de cada vértice do grafo, produzido pela chamada BFS(G, s), é igual a $\delta(s, v)$.

Corretude

Seja $G = (V, E)$ um grafo orientado ou não orientado e $s \in V$ um vértice arbitrário

Lema 22.1
Se $(u, v) \in E$, então
$\delta(s, v) \leq \delta(s, u) + 1$

Lema 22.2
Se BFS(G, s) é executado, então após a execução
$v.d \geq \delta(s, v)$ para todo vértice $v \in V$

Lema 22.3
Se em um determinado momento da execução de BFS(G, s) a fila Q contém $\langle v_1, v_2, \ldots, v_r \rangle$, então

- $v_r.d \leq v_1.d + 1$; e
- $v_i.d \leq v_{i+1}.d$ para $i = 1, 2, \ldots, r - 1$

Corolário 22.4
Se v_i é inserido antes de v_j na fila durante a execução do BFS, então $v_i.d \leq v_j.d$ no momento que v_j é inserido na fila

Teorema 22.5 - Corretude do BFS
Durante a execução do BFS(G, s), todos os vértices acessíveis a partir de s são descobertos, e depois do término da execução $v.d = \delta(s, v)$ para todo $v \in V$.

Demostração feito em sala. Veja as referências para detalhes.

Árvore de busca em largura

- BFS constrói uma árvore de busca em largura
- 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
  - $V_\pi = \{v \in V : v.\pi \neq \text{NIL}\} \cup \{s\}$ (vértices acessíveis a partir de s)
  - $E_\pi = \{(v.\pi, v) : v \in V_\pi - \{s\}\}$ (arestas que conectam os vértices acessíveis a partir de s - com exceção do próprio s - ao pai)

Árvore de busca em largura

- O subgrafo predecessor G_π é uma árvore de busca em largura
  - 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 de busca em largura é de fato uma árvore, pois é conexa e $|E_\pi| = |V_\pi| - 1$

Árvore de busca em largura

Como exibir o caminho de s para v?

PRINT-PATH(G, S, V)

```
1  if v == s
2      PRINT s
3  elseif v.π == NIL
4      PRINT "não existe nenhum caminho de" s para v
5  else
6      PRINT-PATH(G, s, v.π)
7      PRINT 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

- Thomas H. Cormen et al. Introduction to Algorithms. 3rd edition. Capítulo 22.2.
- Rascunho da prova de corretude do BFS