TÉCNICO LISBOA

Análise e Síntese de Algoritmos
BFS
CLRS Cap. 22

Prof. Pedro T. Monteiro

IST - Universidade de Lisboa

2024/2025

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica [CLRS, Cap.15]
  - Algoritmos greedy [CLRS, Cap.16]
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares [CLRS, Cap.22]
  - Caminhos mais curtos [CLRS, Cap.22,24-25]
  - Árvores abrangentes [CLRS, Cap.23]
  - Fluxos máximos [CLRS, Cap.26]
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais
  - Complexidade Computacional [CLRS, Cap.34]

Procura em Largura Primeiro (BFS)

Procura em Largura Primeiro (BFS)
Dado um grafo $G = (V, E)$ e um vértice fonte s, o algoritmo BFS explora sistematicamente os vértices de G para descobrir todos os vértices atingíveis a partir de s

Intuição
Fronteira entre nós descobertos e não descobertos é expandida uniformemente

- Nós à distância k descobertos antes de qualquer nó à distância $k+1$

Procura em Largura Primeiro (BFS)

Resultado

- Identificação de árvore Breadth-First (BF): caminho mais curto de s para cada vértice atingível v
- $\delta(u, v)$: distância do caminho mais curto (menor número de arcos) de u para v

Procura em Largura Primeiro (BFS)

TÉCNICO LISBOA

Implementação

- *color*[*v*]: cor do vértice *v*

  branco: não visitado
  cinzento: já visitado, mas:
  . algum dos adjacentes não visitado
  . ou procura em algum dos adjacentes não terminada
  preto: já visitado e procura nos adjacentes já terminada

- $\pi[v]$: predecessor de *v* na árvore BF
- $d[v]$: tempo de descoberta do vértice *v*

Procura em Largura Primeiro (BFS)

TÉCNICO LISBOA

BFS(G, s)

```
for each u ∈ G.V \ {s} do
  color[u] ← white; d[u] ← ∞; π[u] ← NIL
end for
color[s] ← gray; d[s] ← 0; π[s] ← NIL
Q ← { s }
while Q ≠ ∅ do
  u ← dequeue(Q)
  for each v ∈ G.Adj[u] do
    if color[v] == white then
      color[v] ← gray; d[v] ← d[u] + 1; π[v] ← u
      enqueue(Q, v)
    end if
  end for
  color[u] ← black
end while
```


Procura em Largura Primeiro (BFS)

TÉCNICO LISBOA

Complexidade

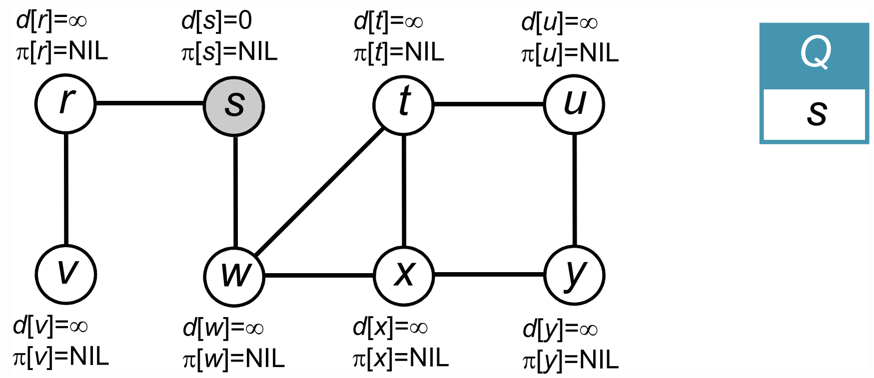
Tempo de execução: $O(V+E)$

- Inicialização: $O(V)$
- Para cada vértice
  - Colocado na fila apenas 1 vez: $O(V)$
  - Lista de adjacências visitada 1 vez: $O(E)$

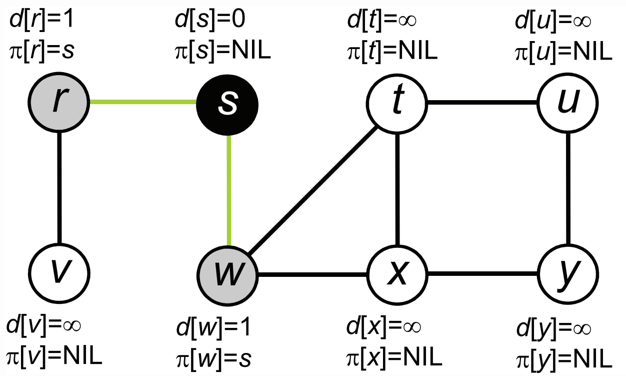
Procura em Largura Primeiro (BFS)

TÉCNICO LISBOA

Exemplo

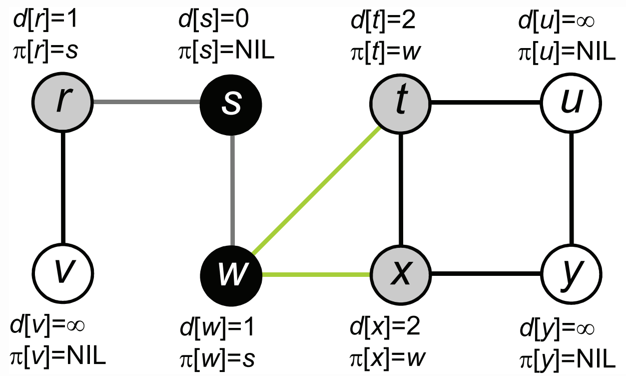


Exemplo



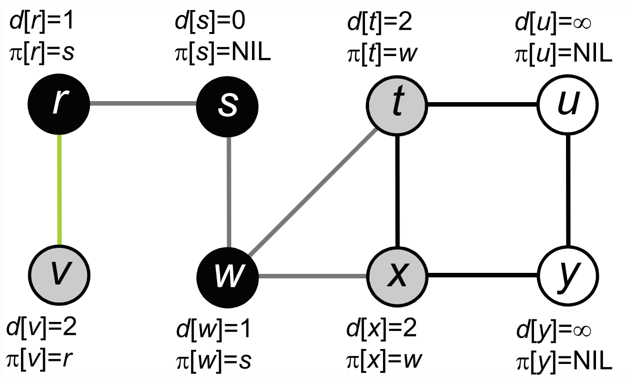
Q
w
r

Exemplo



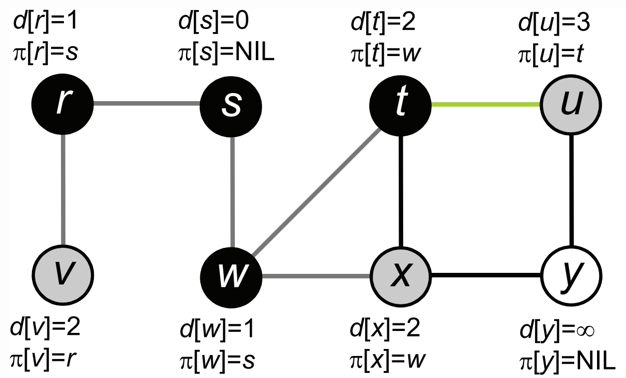
Q
r
t
x

Exemplo



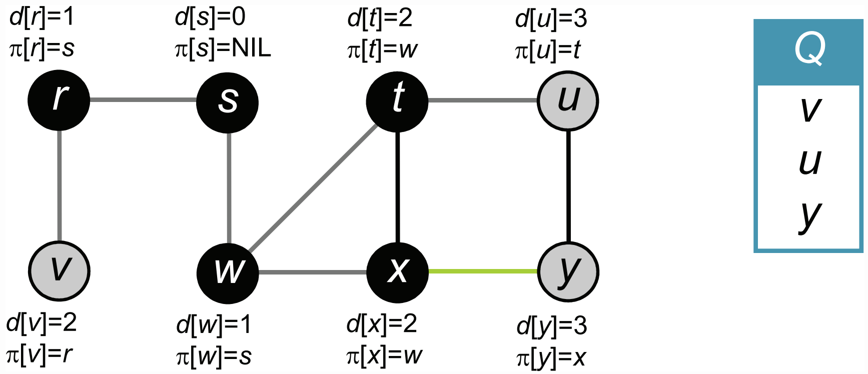
Q
t
x
v

Exemplo

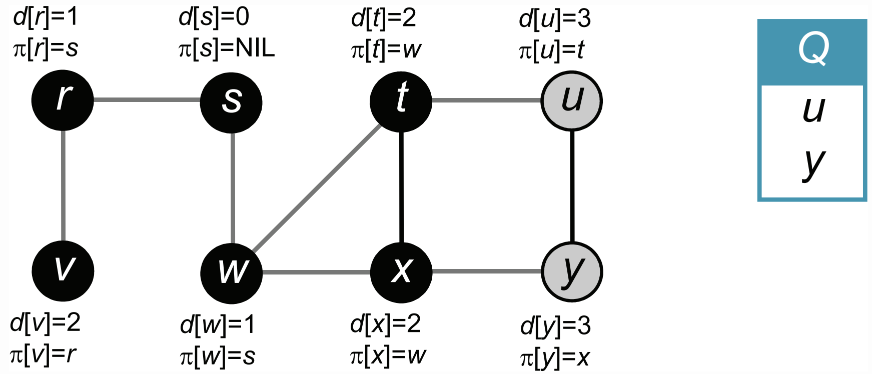


Q
x
v
u

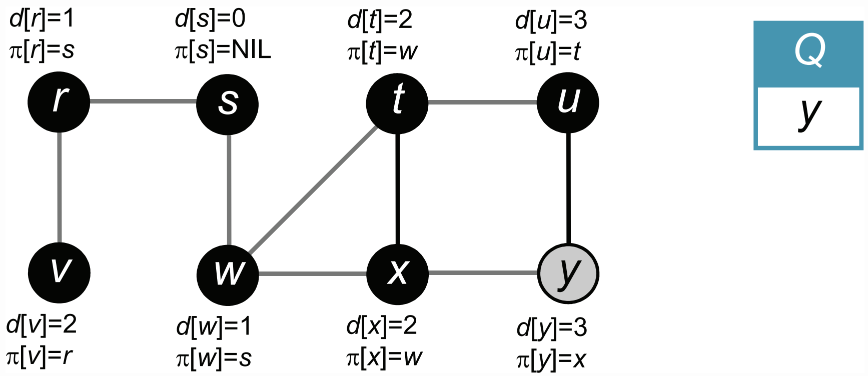
Exemplo



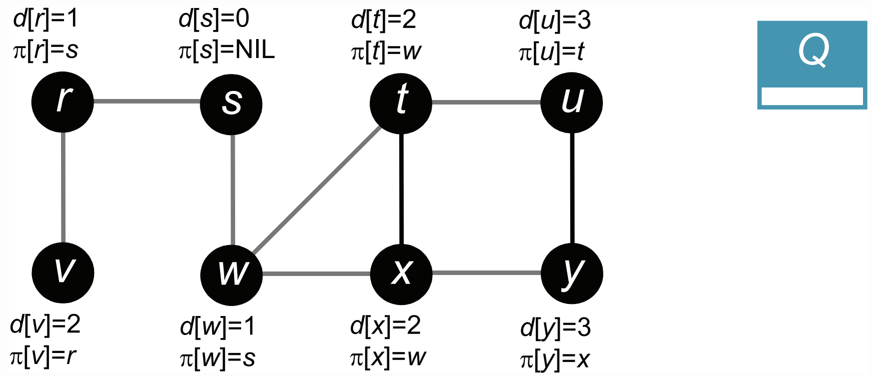
Exemplo



Exemplo



Exemplo



Procura em Largura Primeiro (BFS)

Resultados

Para qualquer arco (u, v):

- Se u atingível, então v atingível
  - caminho mais curto para v não pode ser superior ao caminho mais curto para u mais o arco (u, v)
  - $\delta(s, v) \leq \delta(s, u) + 1$
- Se u não atingível, então resultado é válido (independentemente de v ser atingível)

No final da BFS: $d[u] = \delta(s, u)$, para todo o vértice $u \in V$

Procura em Largura Primeiro (BFS)

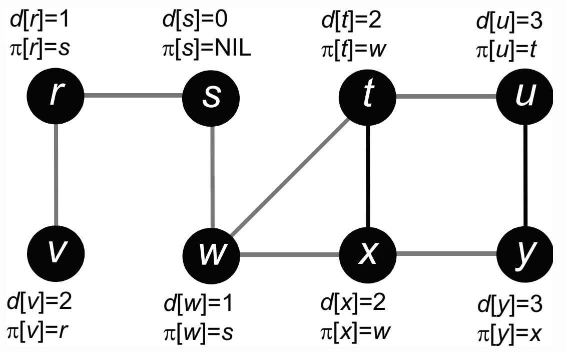
Árvore Breadth-First

- $G_\pi = (V_\pi, E_\pi)$
- $V_\pi = \{v \in V : \pi[v] \neq NIL\} \cup \{s\}$
- $E_\pi = \{(\pi[v], v) \in E : v \in V_\pi \setminus \{s\}\}$

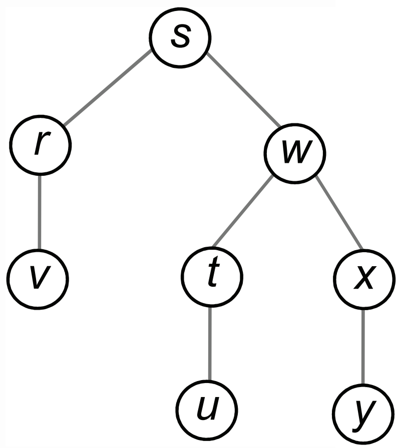
Características

- Árvore BF é sub-grafo de G
- Vértices atingíveis a partir de s
- Arcos que definem a relação predecessor da BFS

Procura em Largura Primeiro (BFS)

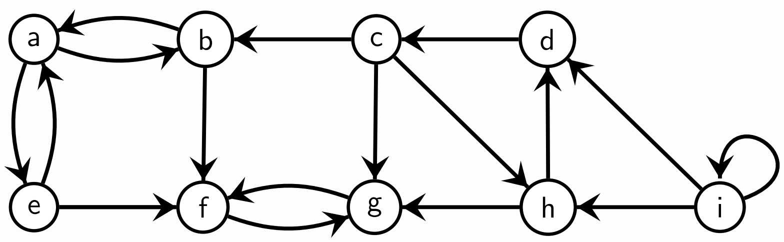


Árvore BF



Procura em Largura Primeiro (BFS)

Exercício: (Aplicar BFS a partir de um vértice qualquer)



Problema 1

Grafo Semi-Ligado

Um grafo dirigido G diz-se semi-ligado se para qualquer par de vértices (u, v), u é atingível a partir de v ou v é atingível a partir de u

Problema

Usando os resultados dos algoritmos elementares (DFS&BFS), indique um algoritmo eficiente para determinar se um grafo $G = (V, E)$ é semi-ligado

Problema 2

Pontos de Articulação

- Um grafo não dirigido diz-se ligado se para qualquer par de vértices $u, v \in V$, existe pelo menos um caminho entre u e v.
- Um vértice $u \in V$ diz-se um ponto de articulação de um grafo se a remoção do vértice u implicar que o grafo deixa de ser ligado.

Problema

Usando os resultados dos algoritmos elementares (DFS&BFS), indique um algoritmo eficiente para determinar se um grafo $G = (V, E)$ não dirigido e ligado tem pontos de articulação

Questões?

Dúvidas?