Grafos: busca em largura

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados II (AE23CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

Sumário

- Busca em Largura
- Implementação da Busca em Largura
- Análise de Complexidade da Busca em Largura
- Caminho Mais Curto

Introdução

- Grafos são estruturas mais complexas em comparação com listas, vetores e árvores binárias
- É necessário o desenvolvimento de métodos para explorar/percorrer um grafo
- Busca em grafos: processo de seguir sistematicamente pelas arestas a fim de visitar os vértices do grafo
  - Busca em profundidade
  - Busca em largura

Introdução

- Notações
  - Para um grafo G (orientado ou não) denotamos por V ($V[G]$) seu conjunto de vértices e por E ($E[G]$) seu conjunto de arestas
  - A complexidade de algoritmos para grafos é dada em termos de V e/ou E
- Informações relevantes sobre a estrutura do grafo podem ser extraídas
  - Podem ser úteis para projetar algoritmos eficientes para determinados problemas

Sumário

Busca em Largura

Busca em Largura

- Um vértice v é alcançável a partir de um vértice s em um grafo G se existe um caminho de s a v em G

- **Definição**: a distância de s a v é o **comprimento** do **caminho mais curto** de s a v

- Se v **não é alcançável** a partir de s, então dizemos que a distância entre ambos vértices é ∞ (infinita)
  - No início da aplicação do algoritmo, o vértice s começa com o valor de distância igual a zero e os demais com ∞

Busca em Largura

- Um algoritmo de busca em largura recebe um grafo $G = (V, E)$ e um vértice especificado s chamado fonte (*source*)

- Percorre todos os vértices alcançáveis a partir de s em ordem de distância

- O algoritmo constrói uma **árvore de busca em largura** com raiz s
  - Cada caminho entre s e v nessa árvore corresponde a um caminho mais curto entre ambos vértices

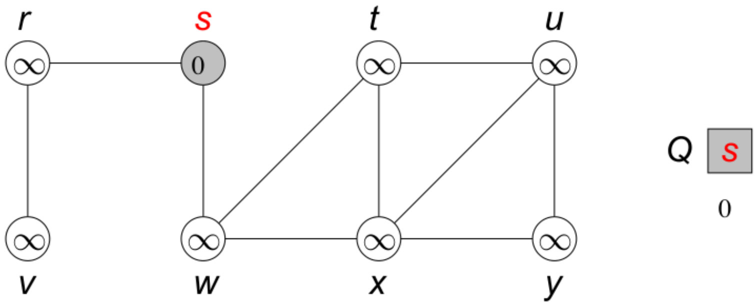
  - O algoritmo descobre todos os vértices a uma distância k do vértice origem antes de descobrir qualquer vértice a uma distância $k + 1$

Busca em Largura

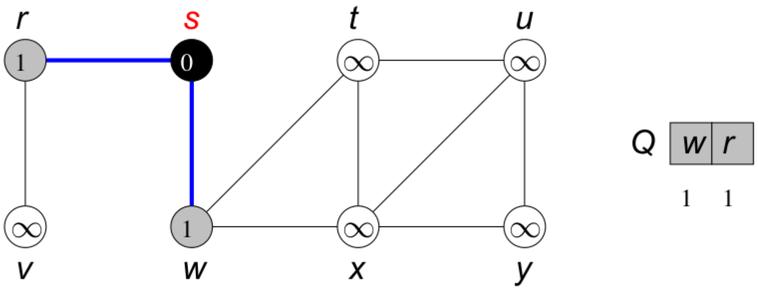
- Inicialmente a árvore de busca em largura contém apenas o vértice fonte s
- Para cada vizinho v de s, o vértice v e a aresta (s, v) são acrescentadas à árvore
- O processo é repetido para os vizinhos dos vizinhos de s
  - Isso é feito até que todos os vértices alcançáveis por s sejam adicionados na árvore
  - O algoritmo de busca em largura também pode formar uma floresta
- O processo de busca é implementado através de uma fila Q

Busca em Largura

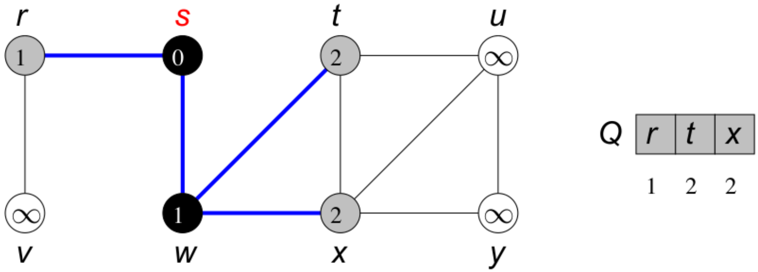
- Durante a aplicação do algoritmo de busca em largura, cada vértice pode ser colorido por meio das seguintes cores
  - Branca: não visitado (inicialmente, todos os vértices são brancos)
  - Cinza: visitado pela primeira vez
  - Preta: todos os seus vizinhos foram visitados
- Vértices de cinza podem ter alguns vértices adjacentes brancos, e eles representam a fronteira entre vértices descobertos e não descobertos



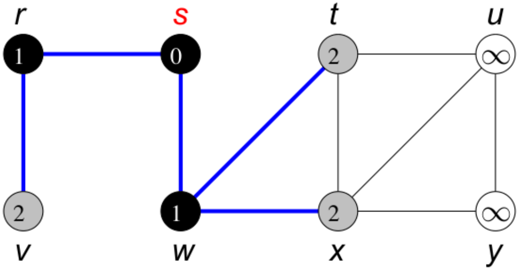
r
s
t
u
∞
0
∞
∞
∞
∞
∞
∞
v
w
x
y
Q
s
0



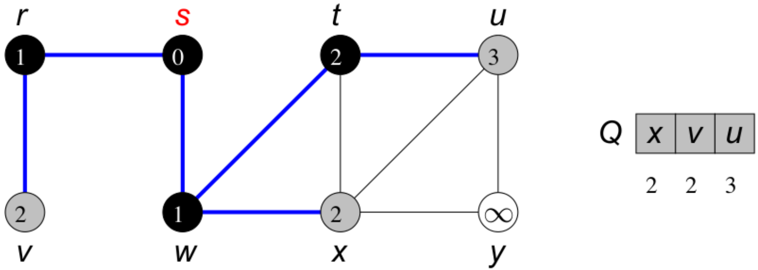
r
s
t
u
1
0
∞
∞
∞
1
∞
∞
v
w
x
y
Q
w
r
1
1

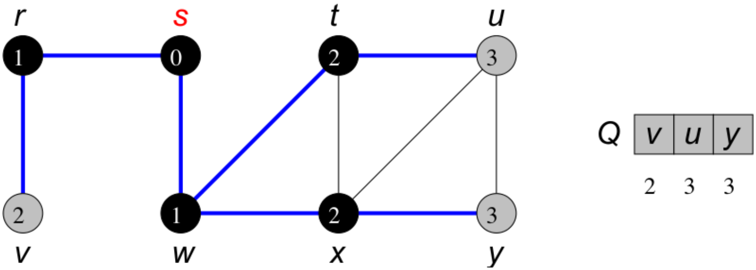


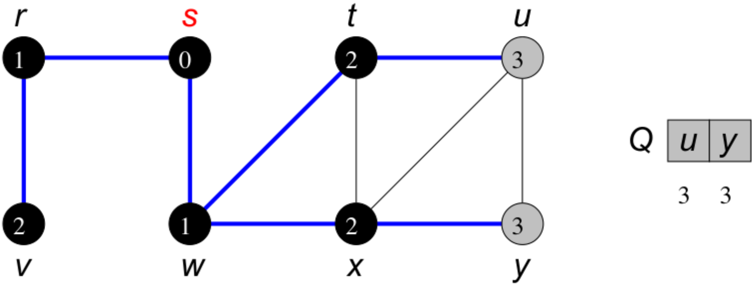
r
s
t
u
1
0
2
∞
∞
1
2
∞
v
w
x
y
Q
r
t
x
1
2
2



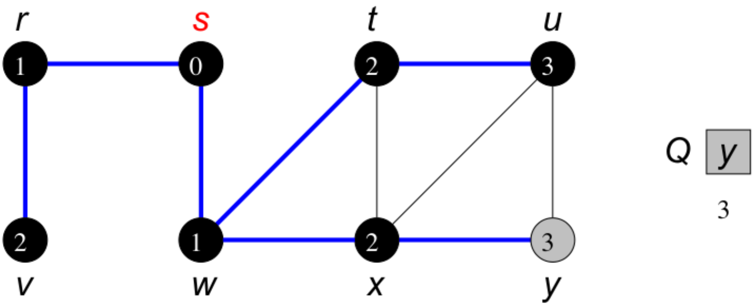
Q	t	x	v
	2	2	2


r
s
t
u
1
0
2
3
2
1
2
∞
v
w
x
y
Q
x
v
u
2
2
3

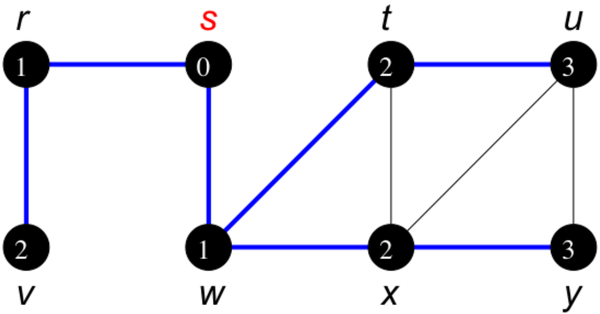

r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
v
u
y
2
3
3



r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
u
y
3
3



r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
y
3



$Q \ \emptyset$

Implementação da Busca em Largura

Implementação da Busca em Largura

- Cores
  - Para cada vértice v, a cor atual é guardada no vetor $cor[v]$, que pode ser branco, cinza ou preto
  - Para o efeito de implementação, o uso da cor não é realmente necessário, mas facilita a compreensão do algoritmo
- A raiz da árvore de busca em largura é s
- Cada vértice v (exceto a raiz) possui um pai $\pi[v]$
- O caminho de s até v é dado por

$$v, \pi[v], \pi[\pi[v]], \pi[\pi[\pi[v]]], ..., s$$

Implementação da Busca em Largura

- A variável $d[v]$ é usada para armazenar a **distância** de s a v
  - Determinada durante o processo de busca

- O algoritmo de busca em largura recebe um grafo G (na forma de listas de adjacências), e um vértice $s \in V[G]$ e devolve
  1. Para cada vértice v, a distância de s a v em G
  2. Árvore ou floresta de busca em largura

Implementação da Busca em Largura

```
BUSCA-EM-LARGURA(G, s)
  ▷ Inicialização
  para cada u ∈ V[G] − {s} faça
      cor[u] ← branco
      d[u] ← ∞
      π[u] ← NIL
  cor[s] ← cinza
  d[s] ← 0
  π[s] ← NIL
  Q ← ∅
  ENQUEUE(Q, s)
```

Implementação da Busca em Largura

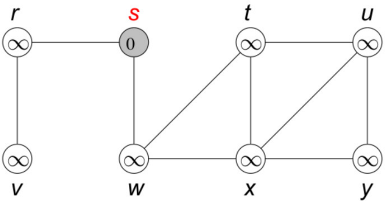
```
enquanto Q ≠ ∅ faça
    u ← DEQUEUE(Q)
    para cada v ∈ Adj[u] faça
        se cor[v] = branco então
            cor[v] ← cinza
            d[v] ← d[u] + 1
            π[v] ← u
            ENQUEUE(Q, v)
    cor[u] ← preto
```

- π é a árvore de busca em largura

BUSCA-EM-LARGURA(G, s)

```
▷ Inicialização
para cada u ∈ V[G] − {s} faça
    cor[u] ← branco
    d[u] ← ∞
    π[u] ← NIL
cor[s] ← cinza
d[s] ← 0
π[s] ← NIL
Q ← ∅
ENQUEUE(Q, s)
```

	r	s	t	u	v	x	y	w
d	b	0	b	b	b	b	b	b
pi	a	a	a	a	a	a	a	a



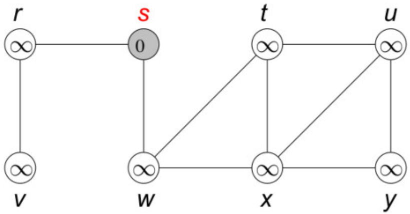
Q: s

0

Implementação da Busca em Largura
Exemplo

```
enquanto Q ≠ ∅ faça
    u ← DEQUEUE(Q)
    para cada v ∈ Adj[u] faça
        se cor[v] = branco então
            cor[v] ← cinza
            d[v] ← d[u] + 1
            π[v] ← u
            ENQUEUE(Q, v)
    cor[u] ← preto
```

	r	s	t	u	v	x	y	w
d	b	0	b	b	b	b	b	b
pi	a	a	a	a	a	a	a	a

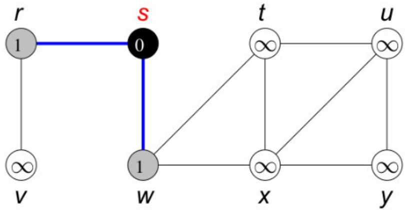


Q | s
0

enquanto $Q \neq \emptyset$ **faça**
  $u \leftarrow$ DEQUEUE(Q)
  para cada $v \in \mathrm{Adj}[u]$ **faça**
    se $cor[v] = \mathrm{branco}$ **então**
      $\mathrm{cor}[v] \leftarrow \mathrm{cinza}$
      $d[v] \leftarrow d[u] + 1$
      $\pi[v] \leftarrow u$
      ENQUEUE(Q, v)
  $\mathrm{cor}[u] \leftarrow \mathrm{preto}$

u = s

	r	s	t	u	v	x	y	w
d	1	0	b	b	b	b	b	1
pi	s	a	a	a	a	a	a	s

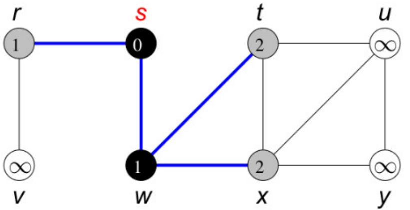


Q | w | r |
1 1

enquanto $Q \neq \emptyset$ **faça**
  $u \leftarrow$ DEQUEUE(Q)
  para cada $v \in \text{Adj}[u]$ **faça**
    se $cor[v] = \text{branco}$ **então**
      $\text{cor}[v] \leftarrow \text{cinza}$
      $d[v] \leftarrow d[u] + 1$
      $\pi[v] \leftarrow u$
      ENQUEUE(Q, v)
  $\text{cor}[u] \leftarrow \text{preto}$

u = w

	r	s	t	u	v	x	y	w
d	1	0	2	b	b	2	b	1
pi	s	a	w	a	a	w	a	s

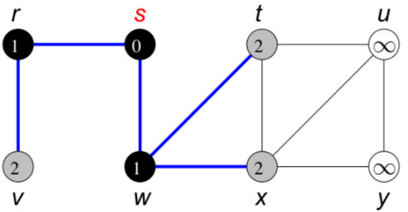


Q	r	t	x
	1	2	2

```
enquanto Q ≠ ∅ faça
    u ← DEQUEUE(Q)
    para cada v ∈ Adj[u] faça
        se cor[v] = branco então
            cor[v] ← cinza
            d[v] ← d[u] + 1
            π[v] ← u
            ENQUEUE(Q, v)
    cor[u] ← preto
```

u = r

	r	s	t	u	v	x	y	w
d	1	0	2	b	2	2	b	1
pi	s	a	w	a	r	w	a	s

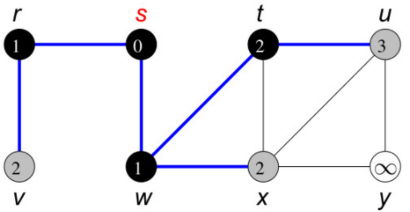


Q	t	x	v
	2	2	2

```
enquanto Q ≠ ∅ faça
    u ← DEQUEUE(Q)
    para cada v ∈ Adj[u] faça
        se cor[v] = branco então
            cor[v] ← cinza
            d[v] ← d[u] + 1
            π[v] ← u
            ENQUEUE(Q, v)
    cor[u] ← preto
```

u = t

	r	s	t	u	v	x	y	w
d	1	0	2	3	2	2	b	1
pi	s	a	w	t	r	w	a	s

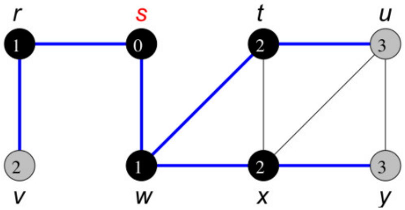


Q	x	v	u
	2	2	3

enquanto $Q \neq \emptyset$ **faça**
  $u \leftarrow$ DEQUEUE(Q)
  para cada $v \in \mathrm{Adj}[u]$ **faça**
    se $cor[v] = \mathrm{branco}$ **então**
      $\mathrm{cor}[v] \leftarrow \mathrm{cinza}$
      $d[v] \leftarrow d[u] + 1$
      $\pi[v] \leftarrow u$
      ENQUEUE(Q, v)
  $\mathrm{cor}[u] \leftarrow \mathrm{preto}$

u = x

	r	s	t	u	v	x	y	w
d	1	0	2	3	2	2	3	1
pi	s	a	w	t	r	w	x	s

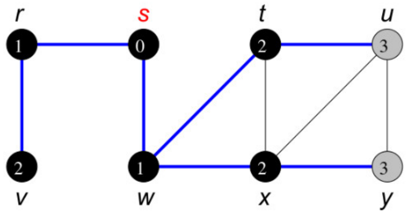


Q	v	u	y
	2	3	3

enquanto $Q \neq \emptyset$ **faça**
  $u \leftarrow \text{DEQUEUE}(Q)$
  para cada $v \in \text{Adj}[u]$ **faça**
    se $cor[v] = \text{branco}$ **então**
      $\text{cor}[v] \leftarrow \text{cinza}$
      $d[v] \leftarrow d[u] + 1$
      $\pi[v] \leftarrow u$
      $\text{ENQUEUE}(Q, v)$
  $\text{cor}[u] \leftarrow \text{preto}$

u = v

	r	s	t	u	v	x	y	w
d	1	0	2	3	2	2	3	1
pi	s	a	w	t	r	w	x	s

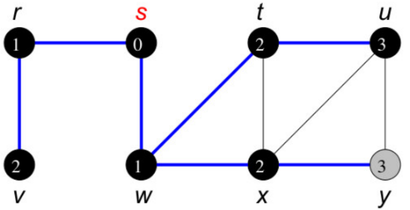


Q	u	y
	3	3

```
enquanto Q ≠ ∅ faça
    u ← DEQUEUE(Q)
    para cada v ∈ Adj[u] faça
        se cor[v] = branco então
            cor[v] ← cinza
            d[v] ← d[u] + 1
            π[v] ← u
            ENQUEUE(Q, v)
    cor[u] ← preto
```

u = u

	r	s	t	u	v	x	y	w
d	1	0	2	3	2	2	3	1
pi	s	a	w	t	r	w	x	s



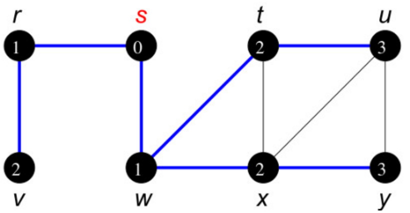
Q | y

3

```
enquanto Q ≠ ∅ faça
    u ← DEQUEUE(Q)
    para cada v ∈ Adj[u] faça
        se cor[v] = branco então
            cor[v] ← cinza
            d[v] ← d[u] + 1
            π[v] ← u
            ENQUEUE(Q, v)
    cor[u] ← preto
```

u = y

	r	s	t	u	v	x	y	w
d	1	0	2	3	2	2	3	1
pi	s	a	w	t	r	w	x	s



Q ∅

Análise de Complexidade da Busca em Largura

Análise de Complexidade da Busca em Largura

- A inicialização consome tempo $O(|V|)$
- Depois que um vértice deixa de ser branco, ele não volta a ser branco novamente
  - Cada vértice é adicionado na fila apenas uma vez
  - Cada operação sobre a fila consome tempo $O(1)$, resultando em um total de $O(|V|)$
- Em uma lista de adjacência, cada vértice é percorrido apenas uma vez
  - A soma dos comprimentos das listas de adjacência é $O(|E|)$
  - Logo, o tempo gasto para percorrer as listas é $O(|E|)$

Análise de Complexidade da Busca em Largura

- **Conclusão**

A complexidade de tempo do algoritmo
BUSCA-EM-LARGURA é $O(|V| + |E|)$

Caminho Mais Curto

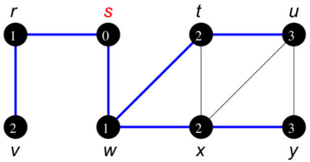
Caminho Mais Curto

```
Print-Path(G, s, v)
    se v = s então
        imprime s
    senão
        se π[v] = NIL então
            imprime não existe caminho de s a v.
        senão
            Print-Path(G, s, π[v])
            imprime v.
```

Caminho Mais Curto

- Imprime um caminho mais curto de s a u

```
Print-Path(G, s, v)
se v = s então
    imprime s
senão
    se π[v] = NIL então
        imprime não existe caminho de s a v.
    senão
        Print-Path(G, s, π[v])
        imprime v.
```



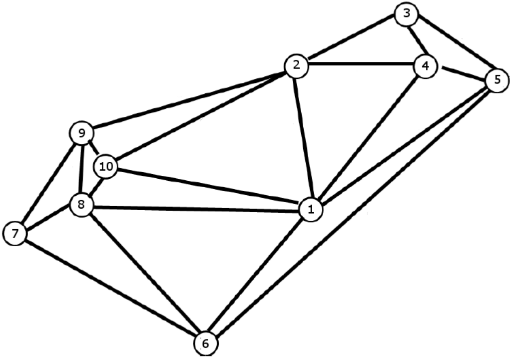
Print-Path(G, s, u)

- Print-Path(G, s, t)
  - Print-Path(G, s, w)
    - Print-Path(G, s, s)
      - imprime s
    - imprime w
  - imprime t
- imprime u

	r	s	t	u	v	x	y	w
pi	s	a	w	t	r	w	x	s

Exercício

- Considere o grafo abaixo e faça:



- Execute a busca em largura considerando o vértice 1 como origem. Para este caso, considere que todas as listas de adjacência estejam ordenadas crescentemente.

Referências I

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.
Introduction to Algorithms.
Third edition, The MIT Press, 2009.

Marin, L. O.
Grafos – Algoritmos de Busca: Busca em Largura. AE23CP – Algoritmos e Estrutura de Dados II.
Slides. Engenharia de Computação. Dainf/UTFPR/Pato Branco, 2017.

Ziviani, N.
Projeto de Algoritmos - com implementações em Java e C++.
Thomson, 2007.