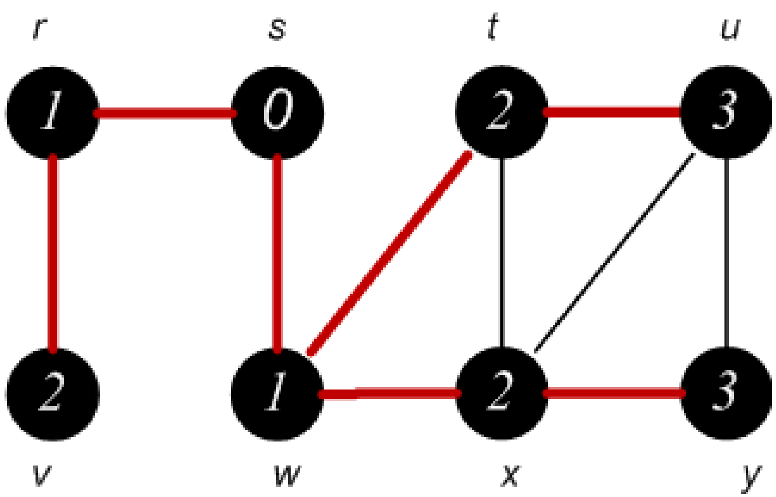
Busca em Largura

Busca em largura

- Um dos algoritmos mais simples da área de grafos;
- Serve de base para vários outros algoritmos:
    - Base para Caminho mais curto (*Dijkstra*);
        - Utilizado para calcular rotas de custo mínimo em um par de localidades em um mapa, por exemplo;
    - Base para Árvore Geradora Mínima - AGM (*Prim*);
        - Utilizado para interligar localidades a um custo mínimo, por exemplo.

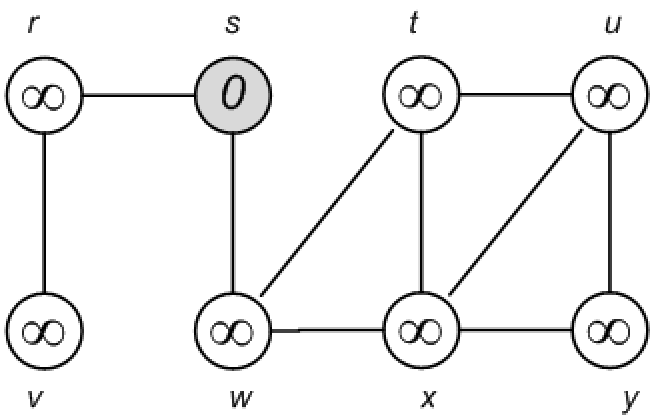
Busca em largura

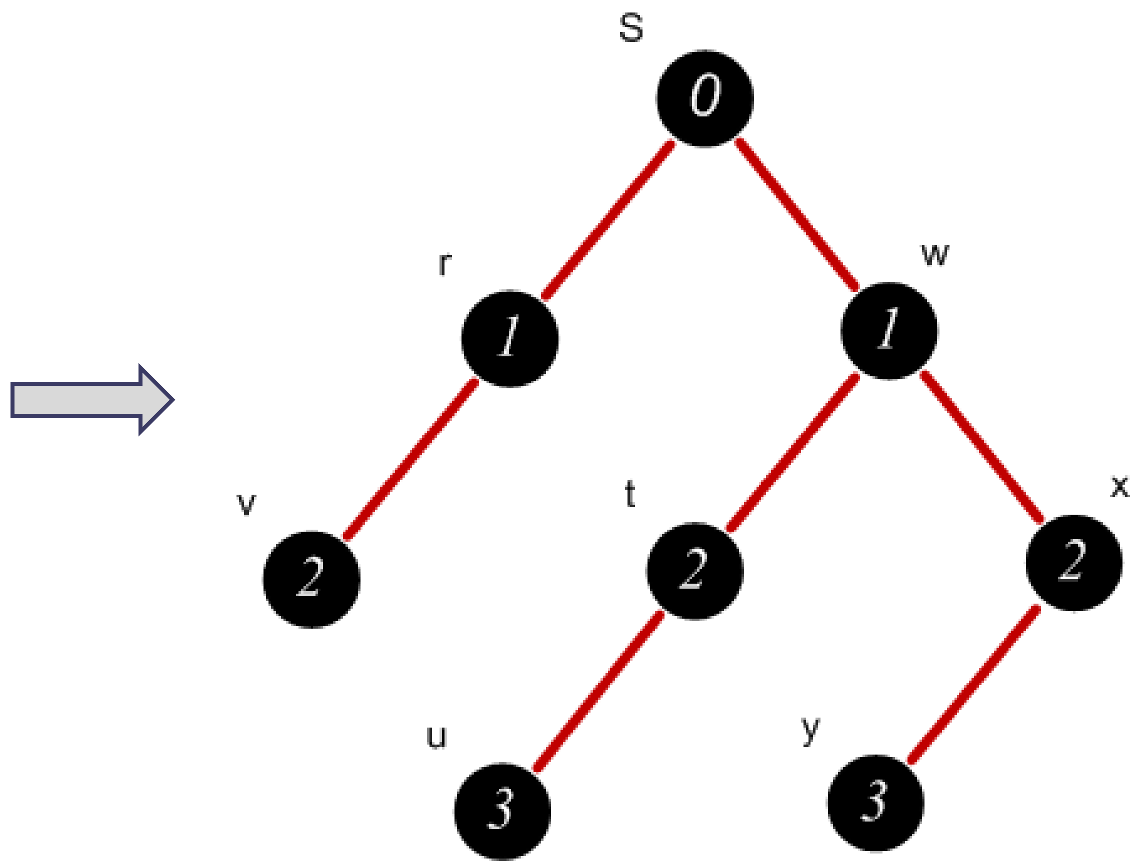
- O algoritmo da Busca em Largura calcula a distância (menor número de arestas) desde o vértice *s* (raiz) até todos os vértices acessíveis;
  - Não considera a distância como o somatório do peso de arestas;
  - Considera a quantidade de saltos necessários mínimos para alcançar outro vértice do grafo;



Busca em largura

- Ele também produz uma "Árvore Primeiro na Extensão", com raiz em no vértice de partida, que contém todos os vértices acessíveis;



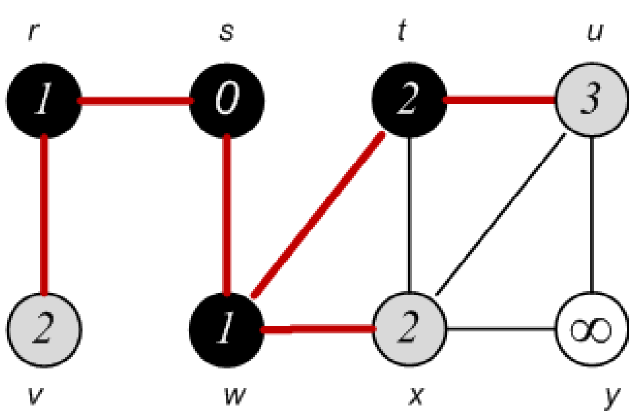


Busca em largura

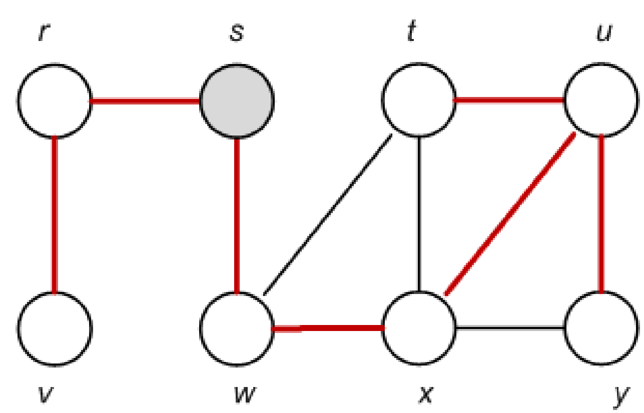
- Para cada vértice *v* acessível a partir de *s*, o caminho na árvore primeiro na extensão de *s* até *v* corresponde a um "caminho mais curto" de *s* até *v*, ou seja, um caminho que contém um número mínimo de arestas;
  - Só é possível porque a busca é "guiada de nível em nível";

- *Observação: Esta informação não é possível ser obtida na busca em profundidade:*

`d(s,t)=2`
na BFS

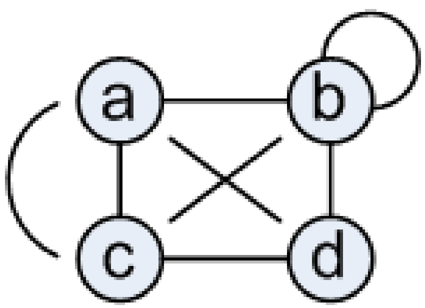


`d(s,t)=4`
na DFS

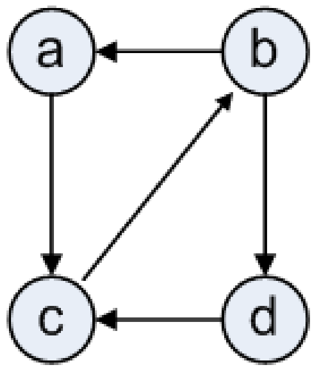


Busca em largura

- Assim como a *Busca em Profundidade (DFS)*, o algoritmo da *Busca em Largura (BFS)* funciona sobre grafos orientados e também não orientados;
  - O que importa, é a relação de adjacência;



G



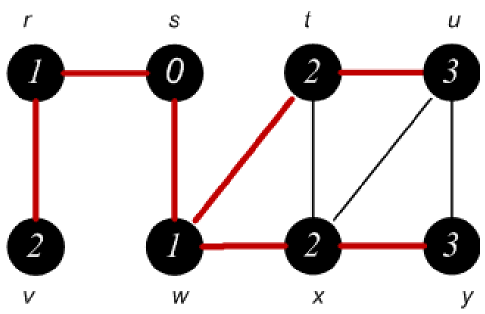
G'

Busca em largura

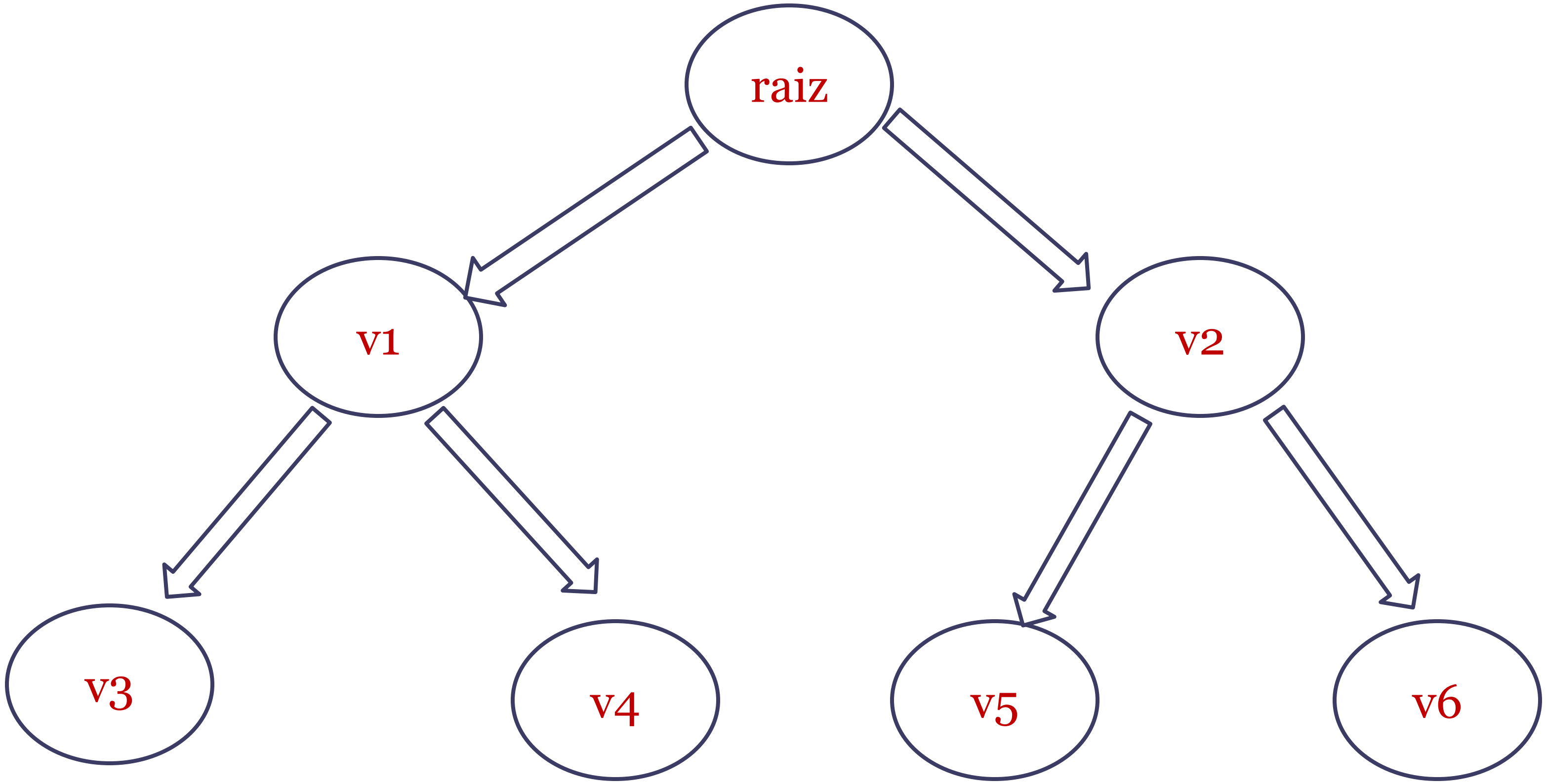
- A busca em largura recebe esse nome porque expande a fronteira entre vértices descobertos e não descobertos uniformemente ao longo da extensão da fronteira;

- Isto é, o algoritmo descobre todos os vértices à distância *k* a partir de *s*, antes de descobrir quaisquer vértices à distância *k+1*; *(ponto chave)*

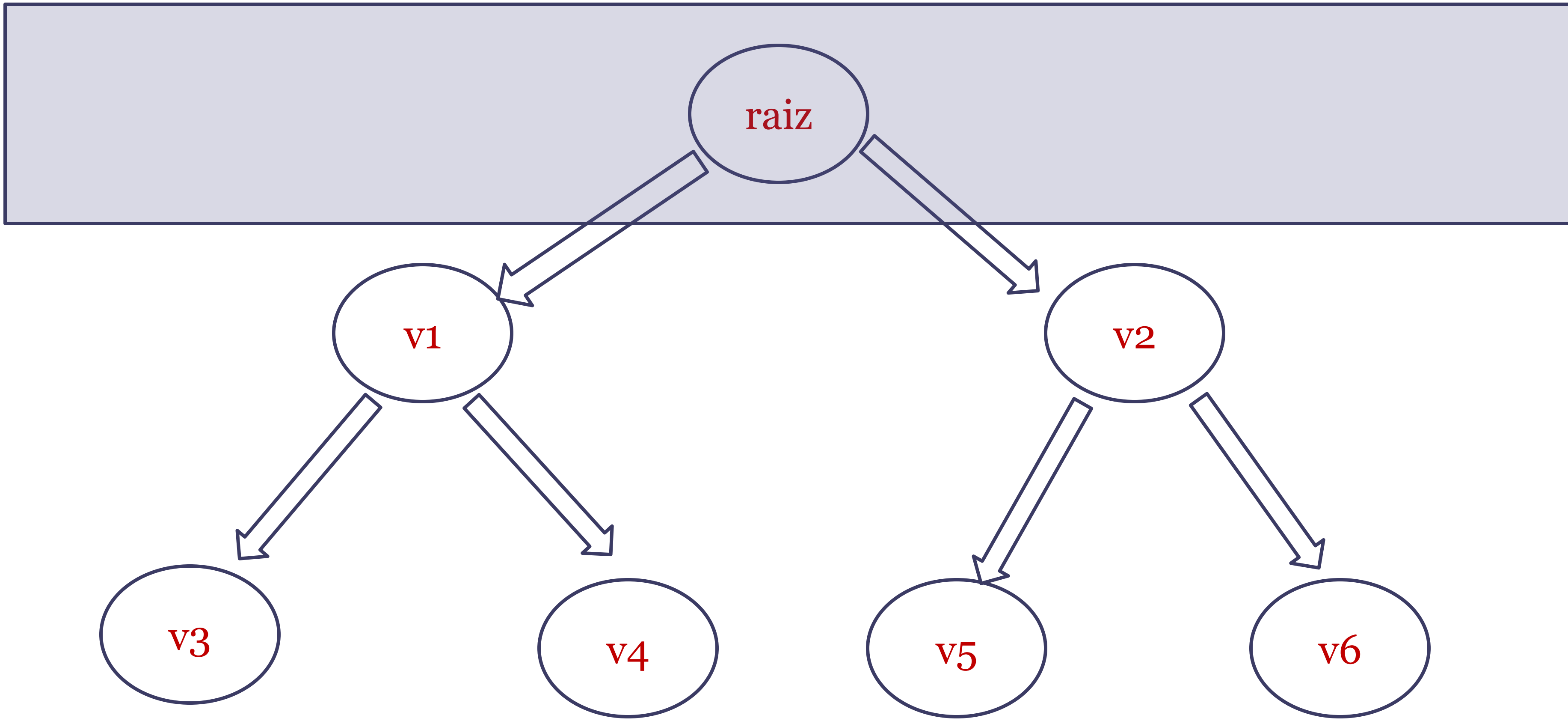
- Comparação com o movimento da água;



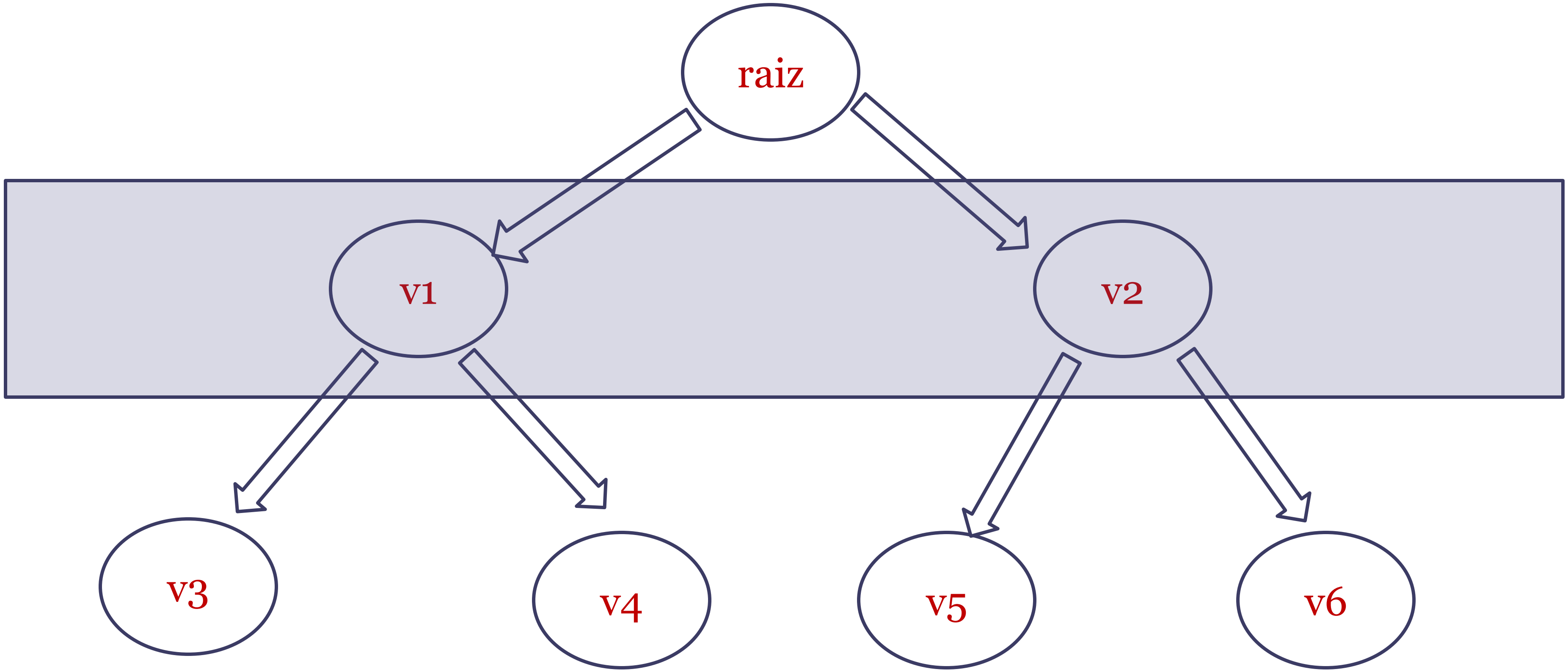
Aplicando Busca em Largura em uma Árvore



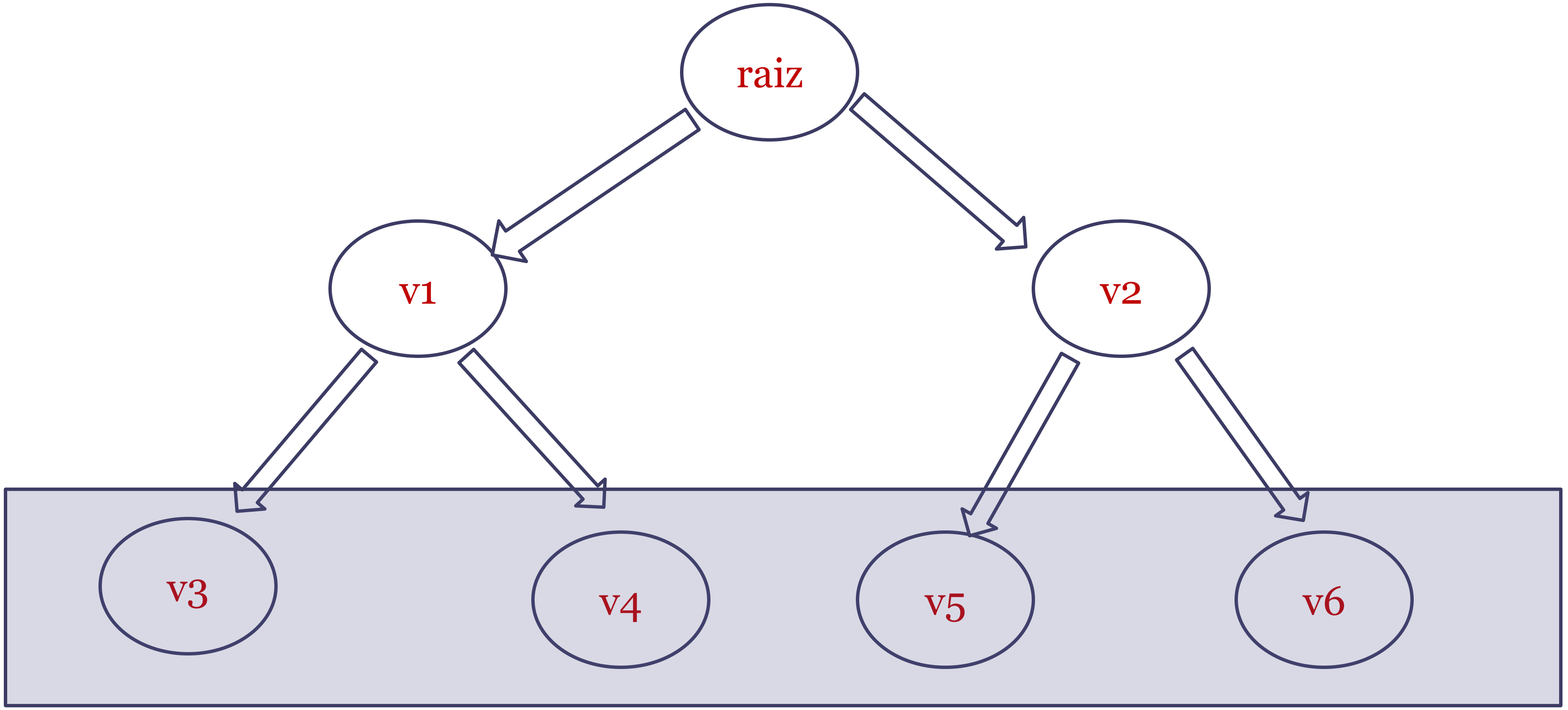
Aplicando Busca em Largura em uma Árvore



Aplicando Busca em Largura em uma Árvore



Aplicando Busca em Largura em uma Árvore



Busca em largura

- O controle do descobrimento dos nós na busca em largura é feito de forma semelhante ao controle utilizado na busca em profundidade anteriormente apresentada:

  - Nó branco = Não visitado/não conhecido;

  - Nó cinza = Nó conhecido/não visitado; Seus adjacentes não foram inseridos em uma fila;

  - Nó preto = Nó conhecido/Nó visitado; Todos os seus adjacentes foram inseridos na fila (não necessariamente visitados, como na DFS);

Busca em largura

- Um vértice é descoberto na primeira vez em que é encontrado;
- Neste momento ele se torna não branco;
- Assim como na DFS, os vértices de cor cinza e preta distinguem os vértices já localizados em duas categorias;
- Vértices de cor cinza podem ter alguns vértices adjacentes brancos; Eles representam a fronteira entre vértices descobertos e não descobertos;

Busca em largura

- A Busca em largura constrói uma árvore primeiro na extensão, contendo inicialmente apenas sua raiz;
- Sempre que um vértice *v* é descoberto no curso da varredura da lista de adjacências de um vértice *u* já descoberto, o vértice *v* e a aresta *(u,v)* são adicionados à árvore primeiro na extensão;
- Neste caso, dizemos que *u* é predecessor ou pai de *v* na árvore primeiro na extensão;

Busca em largura

- Como um vértice é descoberto no máximo uma vez, este possui apenas um pai;
  - A relação de "pai" depende da organização em função da representação do grafo (especificamente da relação de adjacência);

- Conceito de Ancestral:
  - Se u está no caminho na árvore a partir da raiz s até o vértice v, então u é ancestral de v, e v é um descendente de u.

- Tudo depende do nó escolhido para raiz; *As vezes é prefixado, como em algumas aplicações da área de redes*;
  - Roteamento, por exemplo (montando tabelas de encaminhamento);

Busca em largura

- Assim como na DFS, a BFS faz uso de algumas estruturas auxiliares durante a pesquisa:

  - cor[u]; *//indicativo de atingibilidade;*
  - π[u]; *//indica o vértice predecessor de u(pai);*
  - d[u]; *//indica a distância desde a origem d(s,u) - em arestas;*
  - Q; *//indica a fila (FIFO) – ponto chave do algoritmo.*

Busca em Largura

```
BFS(G, s)
para cada vértice u ← V[G] − {s}
    cor[u] ← BRANCO
    d[u] ← ∞
    π[u] ← NULL
cor[s] ← CINZA
d[s] ← 0
π[s] ← NULL
Q ← novaFila()
ENFILEIRA(Q, s)
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u] + 1
            π[v] ← u
            ENFILEIRA(Q, v)
    cor[u] ← PRETO
```

Busca em Largura

$BFS(G, s)$

1 $para\ cada\ vértice\ u \leftarrow V[G] - \{s\}$

2 $\quad cor[u] \leftarrow BRANCO$

3 $\quad d[u] \leftarrow \infty$

4 $\quad \pi[u] \leftarrow NULL$

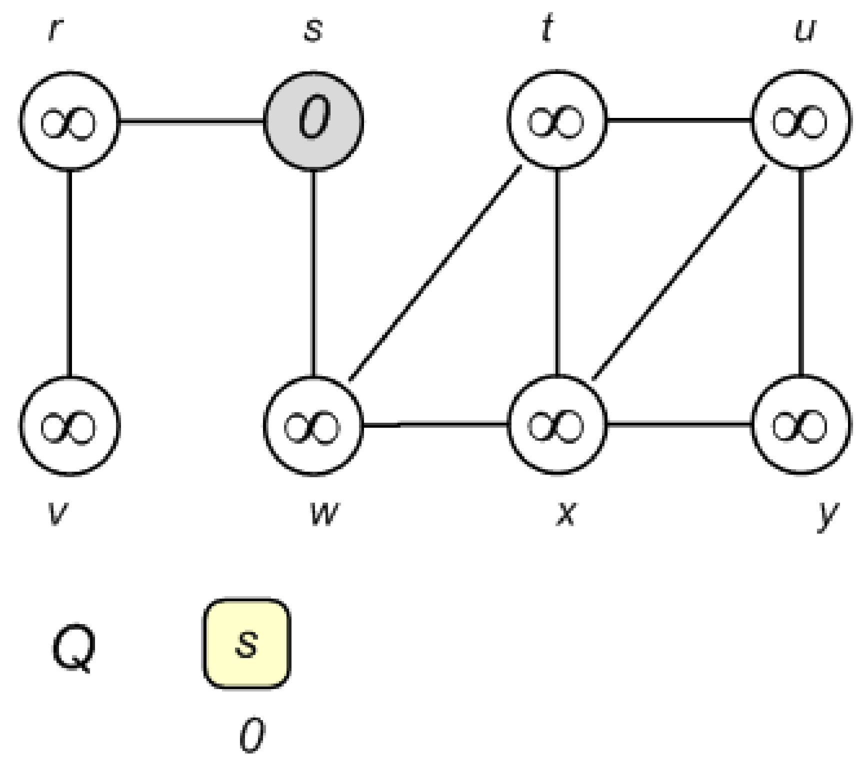
5 $cor[s] \leftarrow CINZA$

6 $d[s] \leftarrow 0$

7 $\pi[s] \leftarrow NULL$

8 $Q \leftarrow nova\,Fila()$

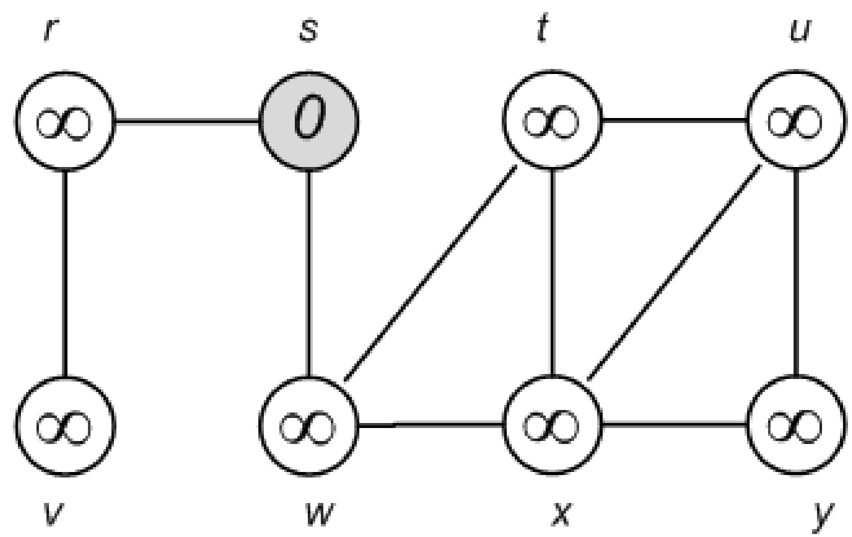
9 $ENFILEIRA(Q, s)$

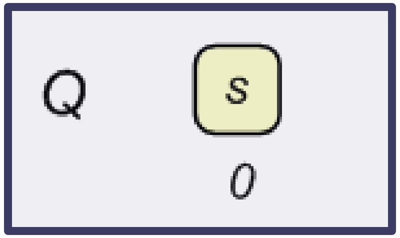


Inicializa as variáveis da BFS

Busca em Largura

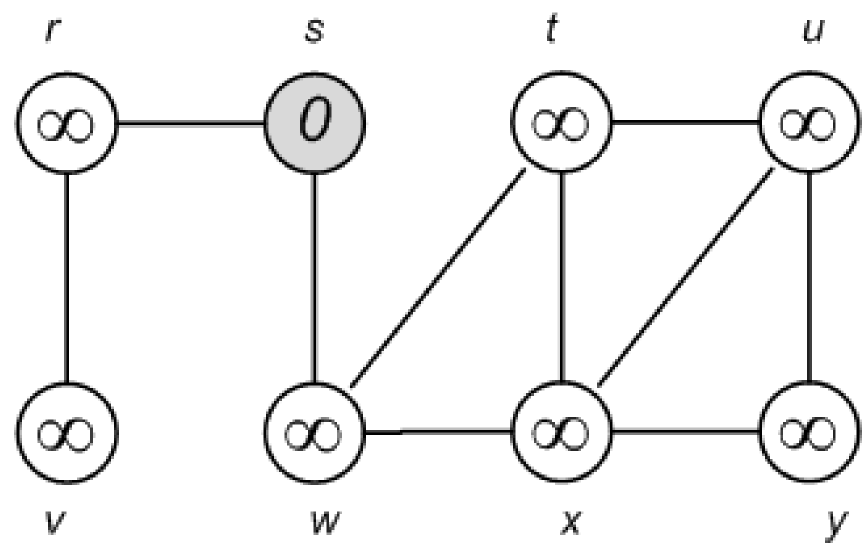
```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u] + 1
            π[v] ← u
            ENFILEIRA(Q, v)
    cor[u] ← PRETO
```





A fila não está vazia!

Busca em Largura




```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u]+1
            π[v] ← u
            ENFILEIRA(Q,v)
    cor[u] ← PRETO
```

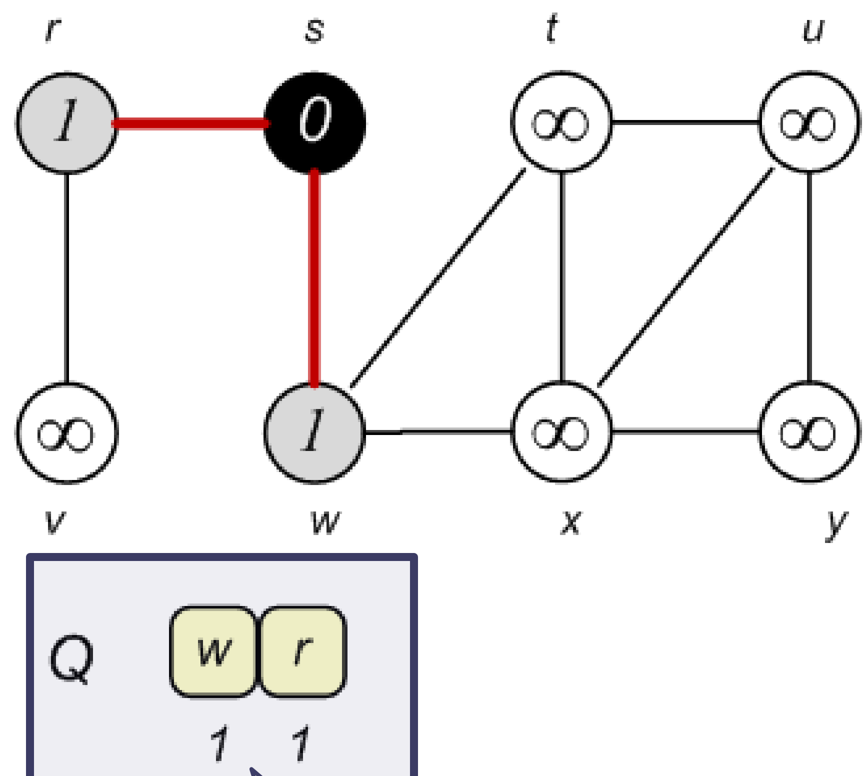
Q

Retira *s* da fila, e parte para seus adjacentes...

u=s

Adj[u]={r,w}

Busca em Largura



$10\ enquanto\ !vazia(Q)$

$\quad u \leftarrow DESENFILEIRA(Q)$

$\quad para\ cada\ v \leftarrow Adj[u]$

$\quad\quad se\ cor[v] = BRANCO$

$\quad\quad\quad cor[v] \leftarrow CINZA$

$\quad\quad\quad d[v] = d[u] + 1$

$\quad\quad\quad \pi[v] \leftarrow u$

$\quad\quad\quad ENFILEIRA(Q, v)$

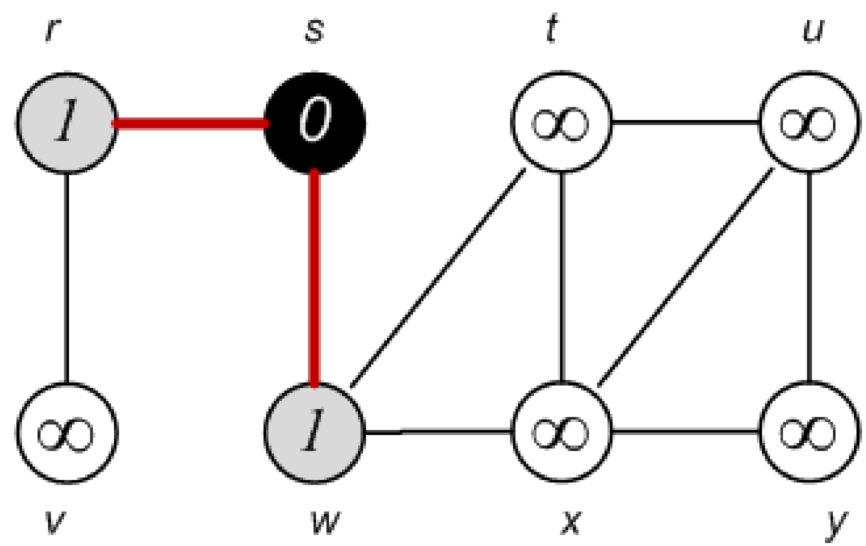
$\quad cor[u] \leftarrow PRETO$

Enfileirou os vértices desconhecidos pela busca...

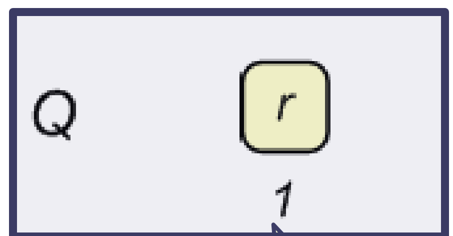
u=s

Adj[u]={r,w}

Busca em Largura




```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u]+1
            π[v] ← u
            ENFILEIRA(Q, v)
    cor[u] ← PRETO
```

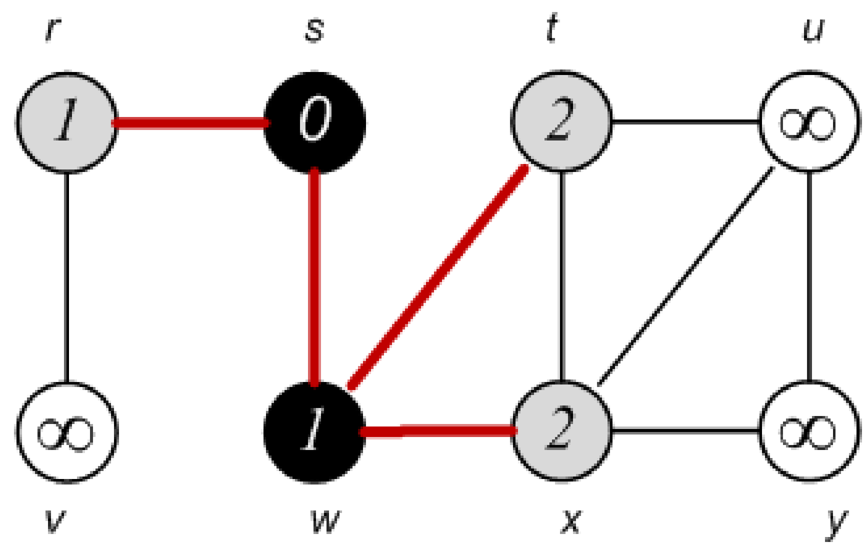


Retira *w* da fila, testa seus adjacentes...

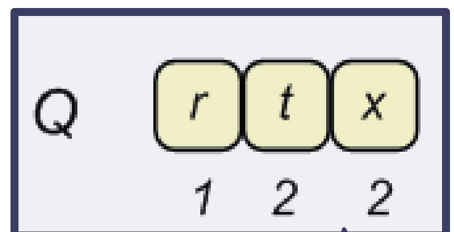
u=w

Adj[u]={s,t,x}

Busca em Largura



$$
\begin{array}{l}
\textit{enquanto } !\mathit{vazia}(Q) \\
\quad u \leftarrow \mathit{DESENFILEIRA}(Q) \\
\quad \textit{para cada } v \leftarrow \mathit{Adj}[u] \\
\quad\quad \textit{se } \mathit{cor}[v] = \mathit{BRANCO} \\
\quad\quad\quad \mathit{cor}[v] \leftarrow \mathit{CINZA} \\
\quad\quad\quad d[v] = d[u] + 1 \\
\quad\quad\quad \pi[v] \leftarrow u \\
\quad\quad\quad \mathit{ENFILEIRA}(Q, v) \\
\quad \mathit{cor}[u] \leftarrow \mathit{PRETO}
\end{array}
$$

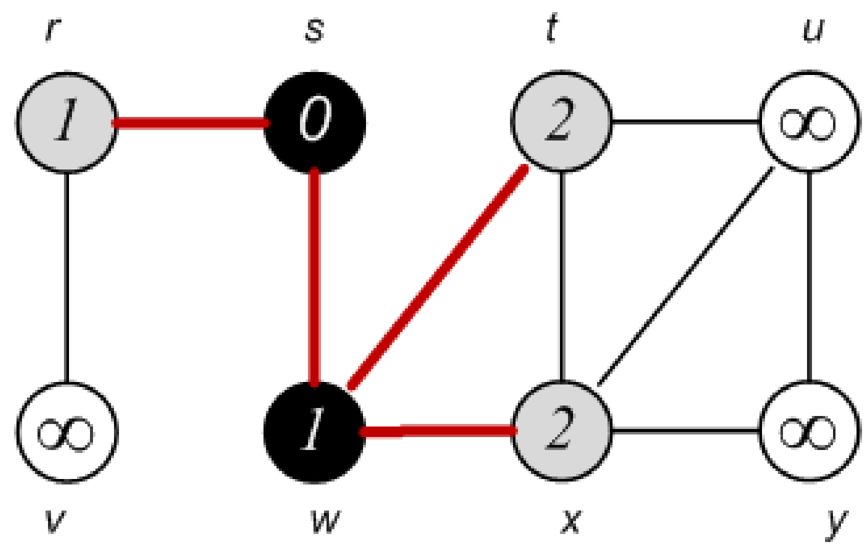


Enfileirou os vértices desconhecidos pela busca...

u=s

Adj[u]={t,x}

Busca em Largura



$enquanto\ !vazia(Q)$

$\boxed{u \leftarrow DESENFILEIRA(Q)}$

$\boxed{para\ cada\ v \leftarrow Adj[u]}$

$\quad se\ cor[v] = BRANCO$

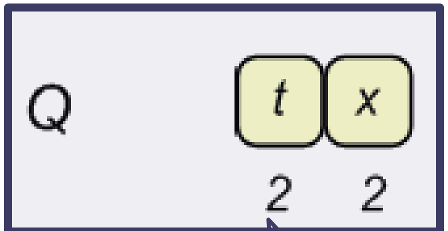
$\quad\quad cor[v] \leftarrow CINZA$

$\quad\quad d[v] = d[u]+1$

$\quad\quad \pi[v] \leftarrow u$

$\quad\quad ENFILEIRA(Q, v)$

$cor[u] \leftarrow PRETO$

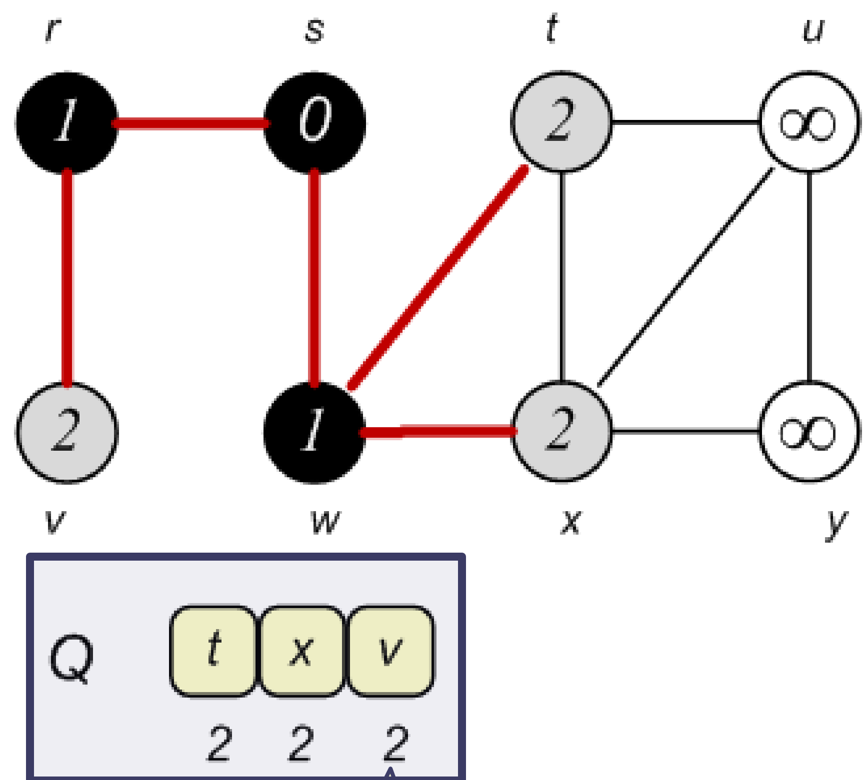


u=r

Adj[u]={s,v}

Retira *r* da fila, testa seus adjacentes...

Busca em Largura



$\textit{enquanto } !vazia(Q)$

$\quad u \leftarrow DESENFILEIRA(Q)$

$\quad \textit{para cada } v \leftarrow Adj[u]$

$\quad\quad \textit{se } cor[v] = BRANCO$

$\quad\quad\quad cor[v] \leftarrow CINZA$

$\quad\quad\quad d[v] = d[u] + 1$

$\quad\quad\quad \pi[v] \leftarrow u$

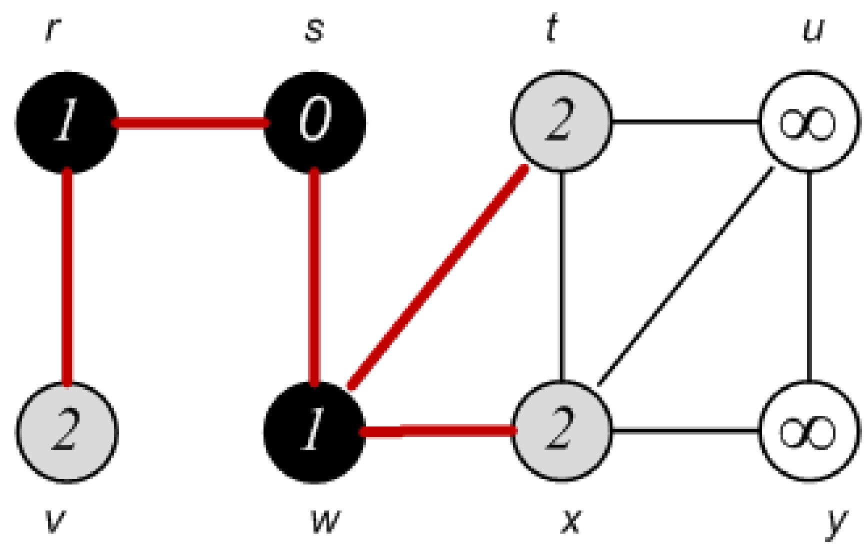
$\quad\quad\quad ENFILEIRA(Q, v)$

$\quad cor[u] \leftarrow PRETO$

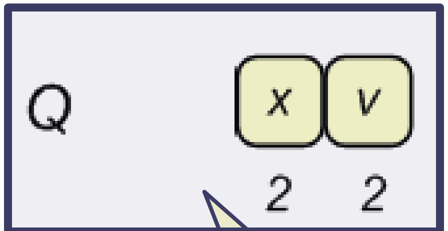
u=r
Adj[u]={s,v}

Enfileirou os vértices desconhecidos pela busca...

Busca em Largura



$$
\begin{array}{l}
\textit{enquanto}\ !vazia(Q) \\
\quad u \leftarrow DESENFILEIRA(Q) \\
\quad \textit{para cada}\ v \leftarrow Adj[u] \\
\qquad \textit{se}\ cor[v] = BRANCO \\
\qquad\quad cor[v] \leftarrow CINZA \\
\qquad\quad d[v] = d[u]+1 \\
\qquad\quad \pi[v] \leftarrow u \\
\qquad\quad ENFILEIRA(Q,v) \\
\quad cor[u] \leftarrow PRETO
\end{array}
$$

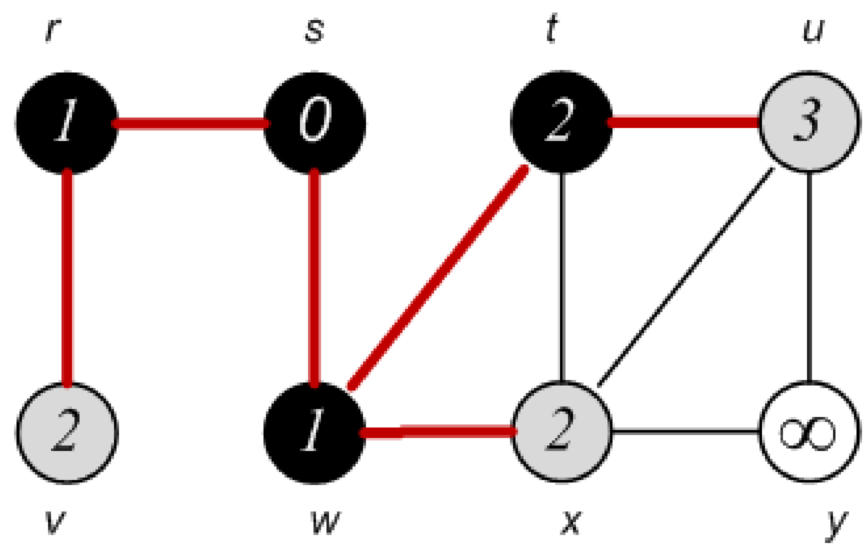


u=t

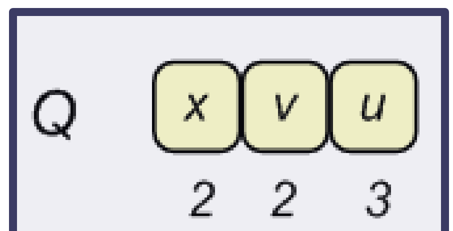
Adj[u]={w,x,u}

Retira *t* da fila, testa seus adjacentes...

Busca em Largura




```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u]+1
            π[v] ← u
            ENFILEIRA(Q,v)
    cor[u] ← PRETO
```

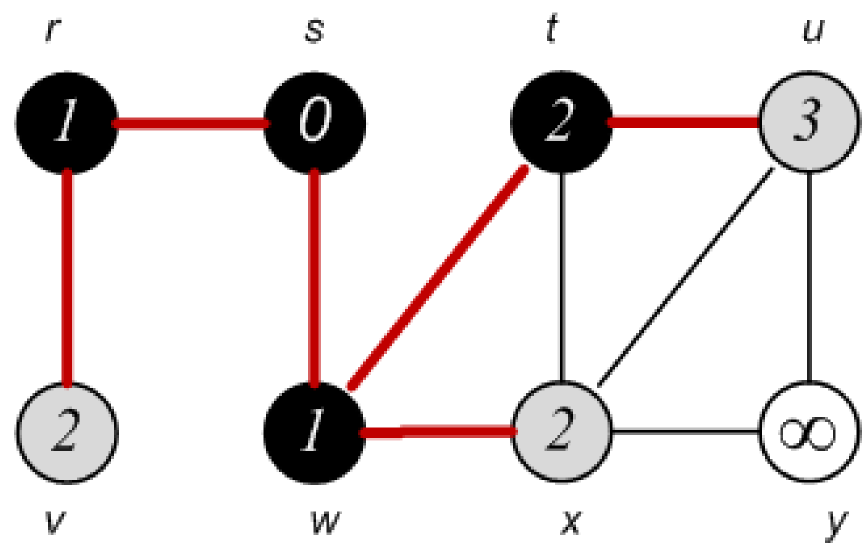


u=t
Adj[u]={w,x,u}

Enfileirou os vértices desconhecidos pela busca...

Busca em Largura

```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u]+1
            π[v] ← u
            ENFILEIRA(Q,v)
    cor[u] ← PRETO
```

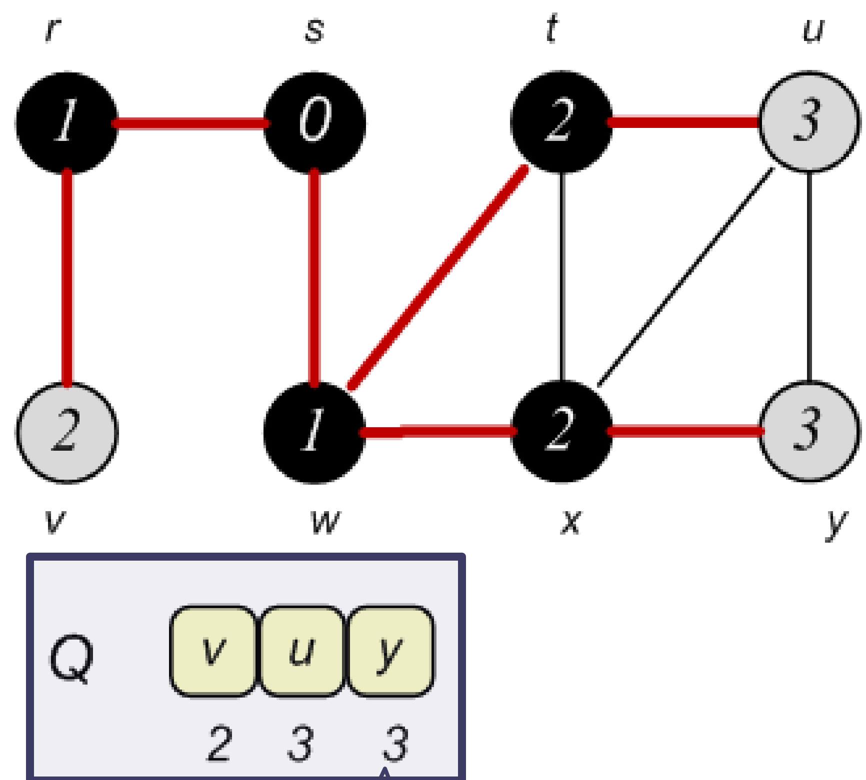


u=x

Adj[u]={w,t,u,y}

Retira *x* da fila, testa seus adjacentes...

Busca em Largura

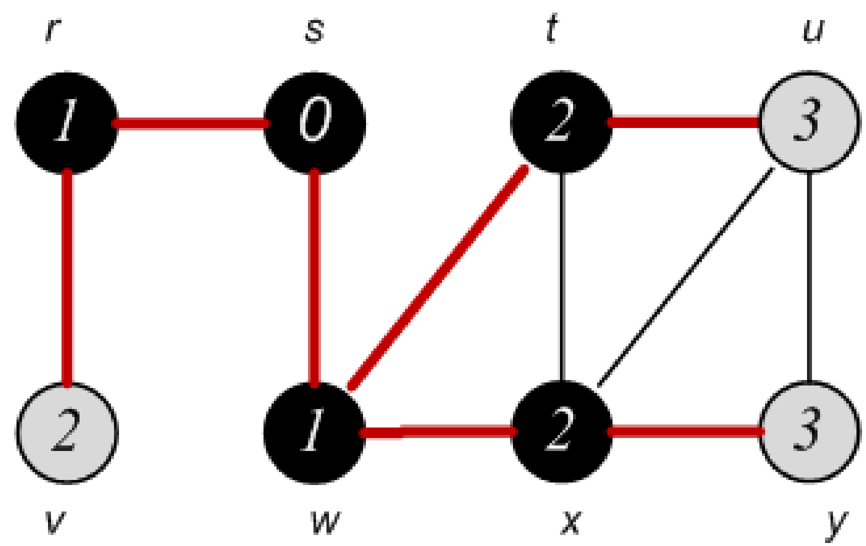



```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u]+1
            π[v] ← u
            ENFILEIRA(Q,v)
    cor[u] ← PRETO
```

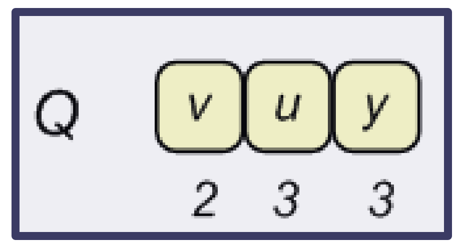
u=x
Adj[u]={w,t,u,y}

Enfileirou os vértices desconhecidos pela busca...

Busca em Largura

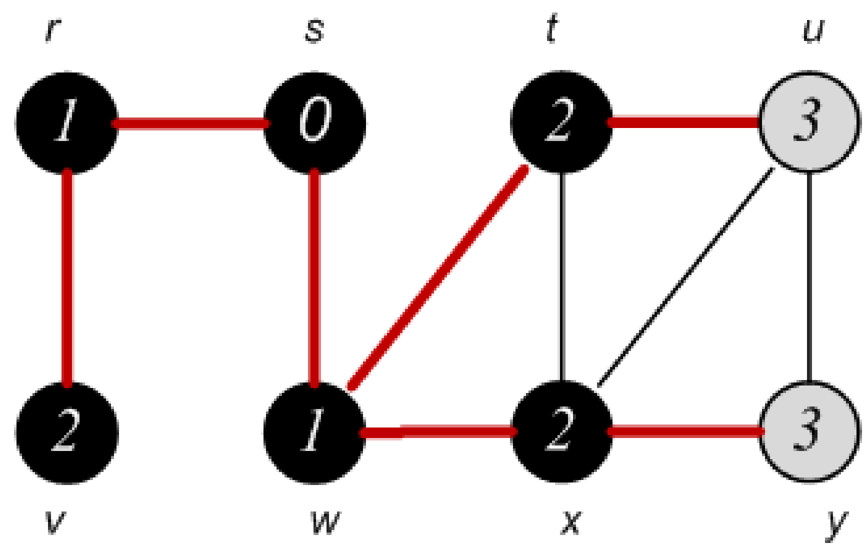



```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u]+1
            π[v] ← u
            ENFILEIRA(Q,v)
    cor[u] ← PRETO
```

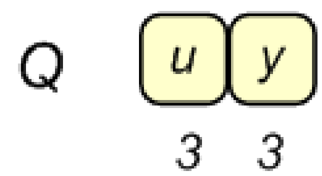


Marcando os vértices de preto...

Busca em Largura



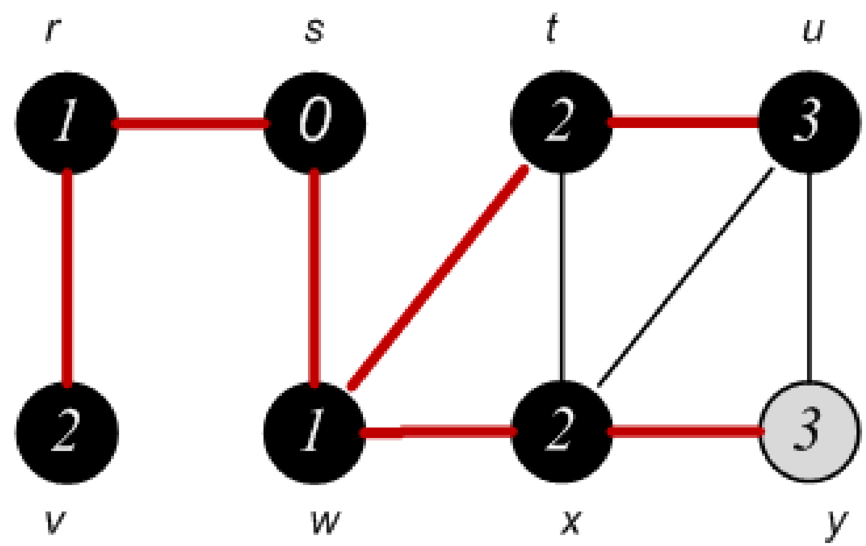

```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u]+1
            π[v] ← u
            ENFILEIRA(Q,v)
    cor[u] ← PRETO
```

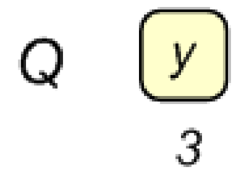


Marcando os vértices de preto...

Busca em Largura

$$
\begin{array}{l}
\textit{enquanto } !\textit{vazia}(Q) \\
\quad u \leftarrow \textit{DESENFILEIRA}(Q) \\
\quad \textit{para cada } v \leftarrow \textit{Adj}[u] \\
\quad\quad \textit{se } \textit{cor}[v] = \textit{BRANCO} \\
\quad\quad\quad \textit{cor}[v] \leftarrow \textit{CINZA} \\
\quad\quad\quad d[v] = d[u] + 1 \\
\quad\quad\quad \pi[v] \leftarrow u \\
\quad\quad\quad \textit{ENFILEIRA}(Q, v) \\
\quad \textit{cor}[u] \leftarrow \textit{PRETO}
\end{array}
$$

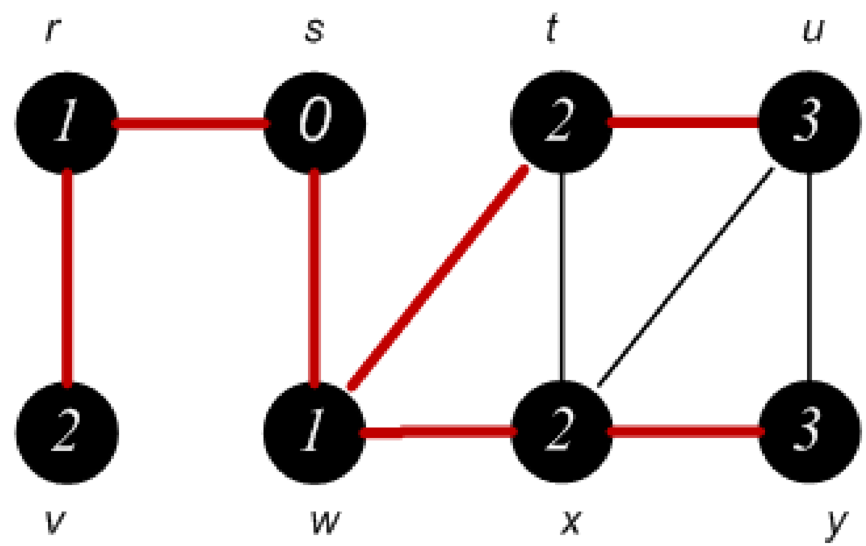




Marcando os vértices de preto...

Busca em Largura

```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u]+1
            π[v] ← u
            ENFILEIRA(Q,v)
    cor[u] ← PRETO
```

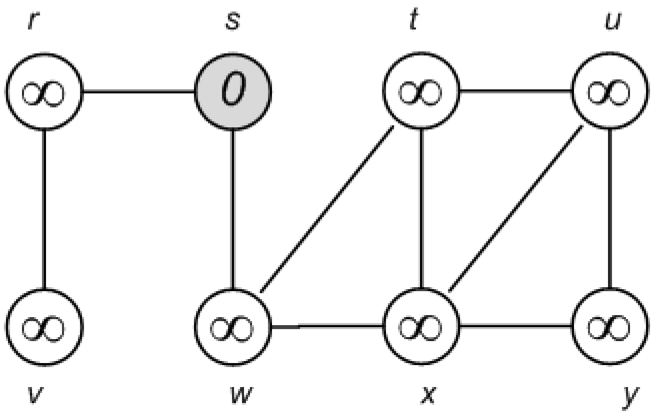


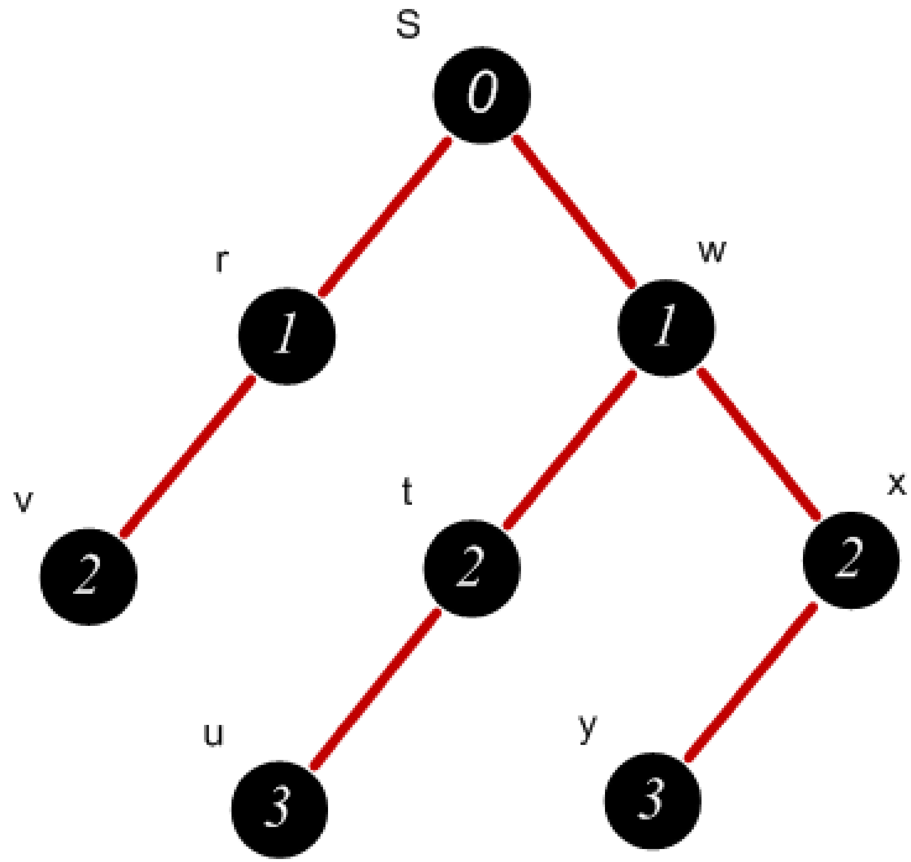
Q

Marcando os vértices de preto...

Busca em Largura

Árvore gerada na busca





Vetor Π

Índice:	s	y	x	t	w	u	v	r
Valor:	NULL	x	w	w	s	t	r	s

Busca em largura

Análise de complexidade

```
BFS(G, s)
para cada vértice u ← V[G] − {s}
    cor[u] ← BRANCO
    d[u] ← ∞
    π[u] ← NULL
cor[s] ← CINZA
d[s] ← 0
π[s] ← NULL
Q ← novaFila()
ENFILEIRA(Q, s)
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u] + 1
            π[v] ← u
            ENFILEIRA(Q, v)
    cor[u] ← PRETO
```

Busca em largura

Análise de complexidade

- Obviamente que a complexidade da busca em largura depende diretamente da representação do grafo utilizada;

```
enquanto !vazia(Q)
    u ← DESENFILEIRA(Q)
    para cada v ← Adj[u]
        se cor[v] = BRANCO
            cor[v] ← CINZA
            d[v] = d[u] + 1
            π[v] ← u
            ENFILEIRA(Q, v)
    cor[u] ← PRETO
```

- Utilizando lista de adjacência: O(|V|+|A|)