

* Breadth First Search

Algoritmo mais simples para executar busca em um grafo.

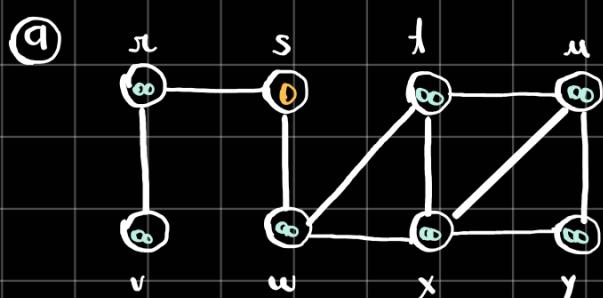
Entrada : $G(V, E)$ e um vértice fonte s

Descobre todos os vértices a uma distância K de s , antes de descobrir quaisquer vértice à distância $K+1$.

Para controlar o progresso pinta os vértices de:

- branco : todos os vértices são iniciados com essa cor
- cinza : quando o vértice foi visitado / descoberto
- preto : vértice explorado

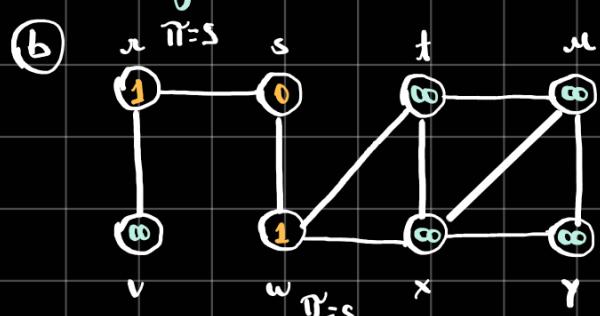
Estrutura Fila (FIFO) usando lista Adjacência



$$Q = [s |]$$

$$d = 0$$

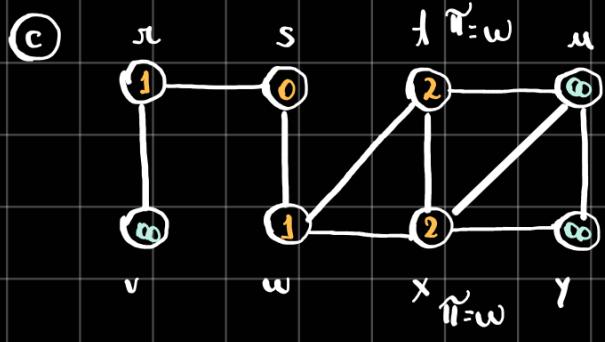
Começa nesse grafo o s está com 0 e vai pra fila como visitado, os outros estão como infinito.



$$Q = [w | r]$$

$$d = 1 \quad 1$$

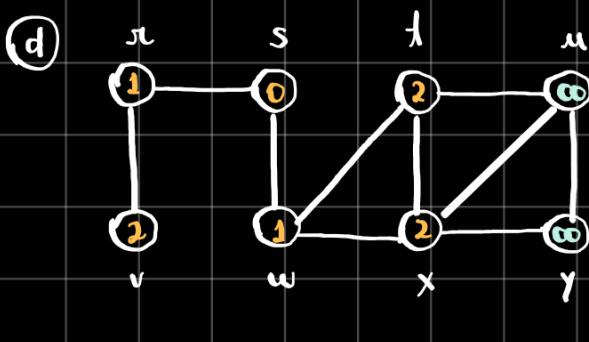
Olho para um dos vizinhos do s , retiro ele da fila e pinto ele de preto adiciono os seus vizinhos na fila e pinto eles de cinza como descoberto. Aumenta a distância e π fica sendo s .



$$Q = [r | t | x]$$

$$d = \begin{matrix} 1 & 2 & 2 \end{matrix}$$

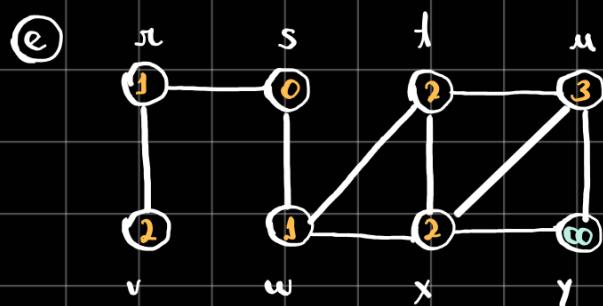
Retiro w da fila pinta de preto
adiciona os vizinhos dele no final
da fila e pinta eles de cinza. Aumenta a
distância e $\tilde{n}=w$.



$$Q = [t | x | v]$$

$$d = \begin{matrix} 2 & 2 & 2 \end{matrix}$$

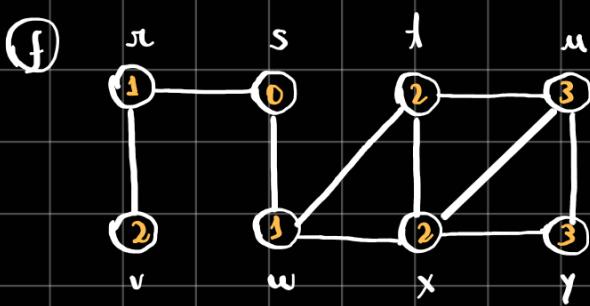
Retiro r da fila pinta de preto,
adiciono o vizinho de v na fila
e pinta de cinza. Aumenta a
distância e $\tilde{n}=w$.



$$Q = [x | v | u]$$

$$d = \begin{matrix} 2 & 2 & 3 \end{matrix}$$

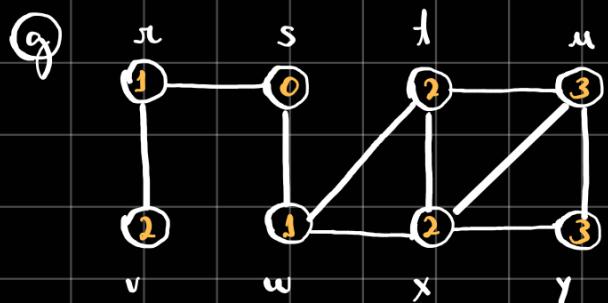
Retiro t da fila pinta preto.
Add u cinza na fila, $d=3$ e $\tilde{n}=t$



$$Q = [v | u | y]$$

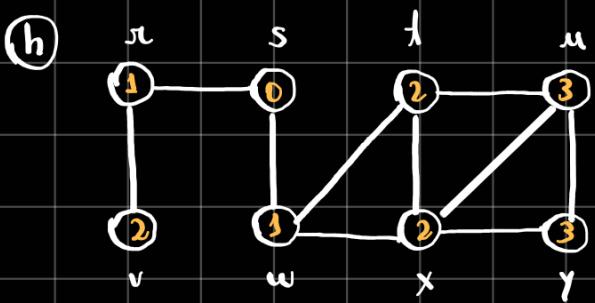
$$d = \begin{matrix} 2 & 3 & 3 \end{matrix}$$

Retiro x da fila pinta preto. Add
 y como cinza, $d=3$ e $\tilde{n}=x$.



$$Q = [u | y]$$

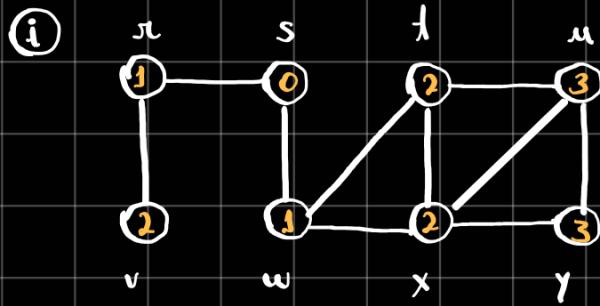
$$\begin{matrix} 3 & 3 \end{matrix}$$



$$Q = [y |]$$

$$\begin{matrix} 3 \end{matrix}$$

Retira u da fila pinta de preto, não tem vizinho.



$$Q = \emptyset$$

Pseudo Código

BFS(G, v)

```

1   for cada vértice  $u \in V[G] - \{s\}$  } Pinta todos os vértices
2      $u.\text{cor} = \text{BRANCO}$  } de branco
3      $u.d = \infty$  } } s já foi descoberto
4      $u.P = \text{NIL}$  } } s com distância zero
5    $s.\text{cor} = \text{CINZA}$  } } predecessor ou pai NULO
6    $s.d = 0$  } } Inicia Q como uma fila
7    $s.P = \text{NIL}$  } } que contém apenas s.
8    $Q = \emptyset$  } Enquanto houver vértices cinza
9   ENQUEUE(Q, s) } que são vértices descobertos cu-
10  while  $Q \neq \emptyset$  } ja lista de adjacência ainda não
11     $u = \text{DEQUEUE}(Q)$  } foram totalmente examinados.
12    for cada  $v = \text{Adj}[u]$  } Na linha 10, Q consiste nos ver-
13      if  $v.\text{cor} == \text{BRANCO}$  } tices cinza. Linha 11 determina
14         $v.\text{cor} == \text{CINZA}$  } o vértice cinza  $u$  no início
15         $v.d = u.d + 1$  } da fila e remove de Q.
16         $v.P = u$  }
17        Enqueue(Q, v) }
18     $u.\text{cor} = \text{PRETO}$  }
```

Retira y da fila pinta de preto, não tem vizinho e não tem mais ninguém. Fila vazia.

1º Coloco cor, distância
e II, depois coloca fila

linhas 12-17 considera cada vértice v na lista de adjacência de u . Se v é branco ainda não foi descoberto, o procedimento descobre executando as linhas 14-17 pintando o v de cinza, define sua distância $v.d = u.d + 1$ registra u como seu pai $v.p$ e coloca no final da fila Q . Examinados todos os vértices na lista de adjacências de u , o procedimento pinta u de preto linha 18. O laço é mantido enquanto a fila não for vazia.

Se $(u, v) \in E$ e o vértice u é preto, então o vértice v é cinza ou preto; isto é todos os vértices adjacentes a vértice pretos foram descobertos. Vértices cinzas podem ter alguns vértices adjacentes brancos, eles representam a fronteira entre vértices descobertos e não descobertos.

Se u está em um caminho simples na árvore que vai da raiz s até o vértice v , então u é ancestral de v , e v é descendente de u . $G(u, v)$

Análise

Tempo de pintar todos os vértices de branco e asseguramos que cada vértice seja colocado na fila no máximo uma vez e é retirado uma vez operação enfileirar / desenfileirar $O(1)$ assim o tempo total dedicado a operações na fila é $O(v)$. Como varremos cada lista de adjacência de cada vértice sómente quando o vértice é desenfilerado, varre cada linha no máximo uma vez. O tempo total gasto na varredura das listas de adjacências é $O(E)$. A sobre carga de inicialização é $O(v)$, portanto o tempo de execução total é $O(V+E)$.