SCC 503 - ALGORITMOS E ESTRUTURAS DE DADOS II

Grafos - Mais Conceitos e Busca em Largura

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

*Material baseado em aulas dos professores: Elaine Parros Machado de Souza, Gustavo Batista, Robson Cordeiro, Moacir Ponti Jr., Maria Cristina Oliveira e Cristina Ciferri.

Comparações entre Matriz e Listas

Comparações entre Matriz e Listas

Operação	Matriz	Listas
Inicializa	$O(\|V\|^2)$	**$O(\|V\|)$**
InsereAresta	$O(1)$	$O(1)$
ExisteAresta	**$O(1)$**	$O(grau(v))$
RetiraAresta	**$O(1)$**	$O(grau(v))$
LiberaGrafo	**$O(1)$**	$O(\|V\|+\|A\|)$
ExisteAdj	$O(\|V\|)$	**$O(1)$**
PrimeiroAdj	$O(\|V\|)$	**$O(1)$**
ProxAdj	$O(\|V\|)$	**$O(1)$**

Comparações entre Matriz e Listas

Operação	Matriz	Listas
Percorrer um grafo	$O(\|V\|^2)$	**$O(\|V\|+\|A\|)$**
Determinar o grau de um vértice em um grafo não direcionado	$O(\|V\|)$	**$O(grau(v))$**
Determinar o grau de um vértice em um grafo direcionado	$O(\|V\|)$	$O(grau(v))$ (*out-degree*) $O(\|V\|+\|A\|)$ (*in-degree*)

Comparações entre Matriz e Listas

- Cada representação é mais eficiente em diferentes operações do TAD.
- Em geral... a representação de listas de adjacências é a mais utilizada.
  - Muitos algoritmos clássicos sobre grafos requerem percorrer todos os vértices.
  - Listas ligadas permitem representar naturalmente arestas múltiplas (multi-grafos).

Caminho de um Grafo

Caminho de um Grafo

- Eu estou ligado a uma celebridade por uma cadeia de amigos?
  - Existe um caminho entre mim e uma celebridade?
  - Caminho
    - Sequência de arestas que conectam dois vértices.

Caminho de um Grafo

- Um caminho de comprimento k de um vértice x a um vértice y em um grafo G = (V,A) é uma sequência de vértices (v0 ,v1 ,v2 ,...,vk) tal que:
  - $x = v_0$
  - $y = v_k$
  - (vi-1, vi) ∈ A para i = 1,2,...,k.

Caminho de um Grafo

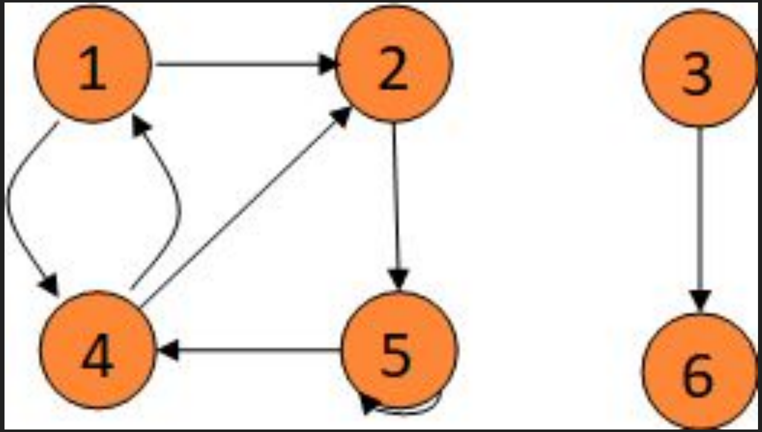
- O comprimento de um caminho é o número de arestas nele, isto é, o caminho contém:
  - Os vértices v0 , v1 , v2 ,..., vk
  - As arestas (v0 , v1),(v1 , v2) ,..., (vk-1 , vk).

Caminho de um Grafo

- Se existir um caminho c de x a y então y é alcançável a partir de x via c.
- Caminho simples
  - Todos os vértices do caminho são distintos.
- Caminho Hamiltoniano
  - Caminho que passa por todos os vértices de um grafo exatamente uma vez

Caminho de um Grafo - Exemplo

- Caminho (1,2,5,4)
  - Simples, com comprimento 3
- Caminho (1,4,1,2)
  - Não é simples, com comprimento 3



Busca em Grafos

Busca em Grafos

- Exemplos:
  - Dado um grafo $G = (V, A)$ e um vértice $v \in V$
    - Encontrar todos os vértices em G que estão conectados a v.
  - Dado um grafo $G = (V, A)$
    - Visitar todos os vértices de G.

Busca em Grafos

- Duas abordagens principais de realizar essas tarefas:
  - Busca em profundidade
  - Busca em largura

Busca em Largura

Busca em Largura

- A busca em largura (breadth-first search) expande a fronteira entre vértices descobertos e não descobertos uniformemente através da “largura da fronteira”.
- 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

- A busca em largura permite descobrir todos os vértices alcançáveis a partir de um vértice de origem u, com o menor número de arestas entre u e todos os outros vértices

Busca em Largura

- Busca em largura é base de algoritmos para:
  - Encontrar a árvore geradora mínima (MST)
    - Algoritmo de Prim
  - Encontrar o caminho mais curto de um vértice v a todos os outros
    - Algoritmo de Dijkstra

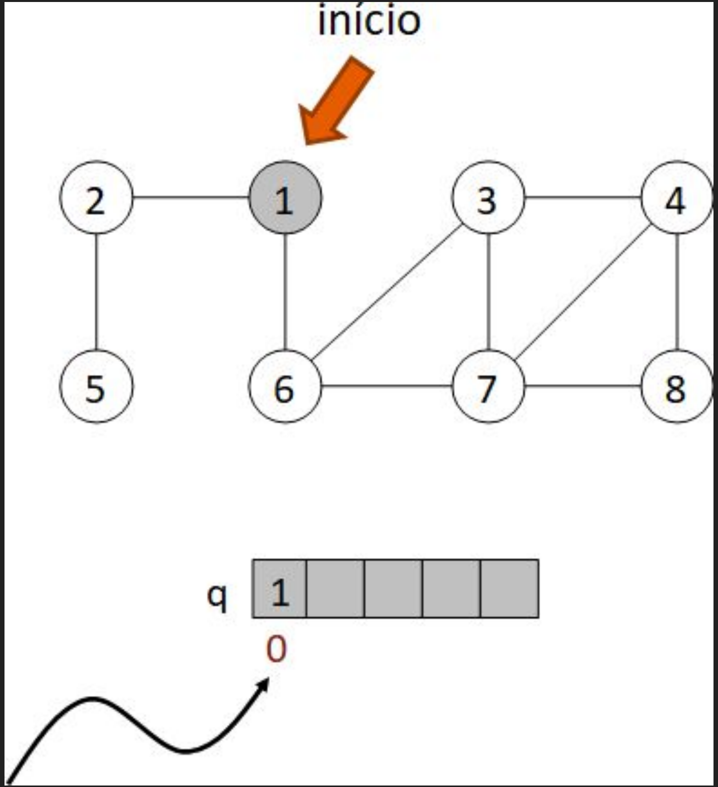
Busca em Largura

- Estratégia:
  - Cada vértice é colorido de branco, cinza ou preto.
  - Todos os vértices são inicialmente brancos.
  - Quando um vértice é visitado pela primeira vez ele torna-se cinza.
  - Vértices cinza e preto já foram visitados, mas são diferenciados para assegurar que a busca ocorra em largura.

Busca em Largura

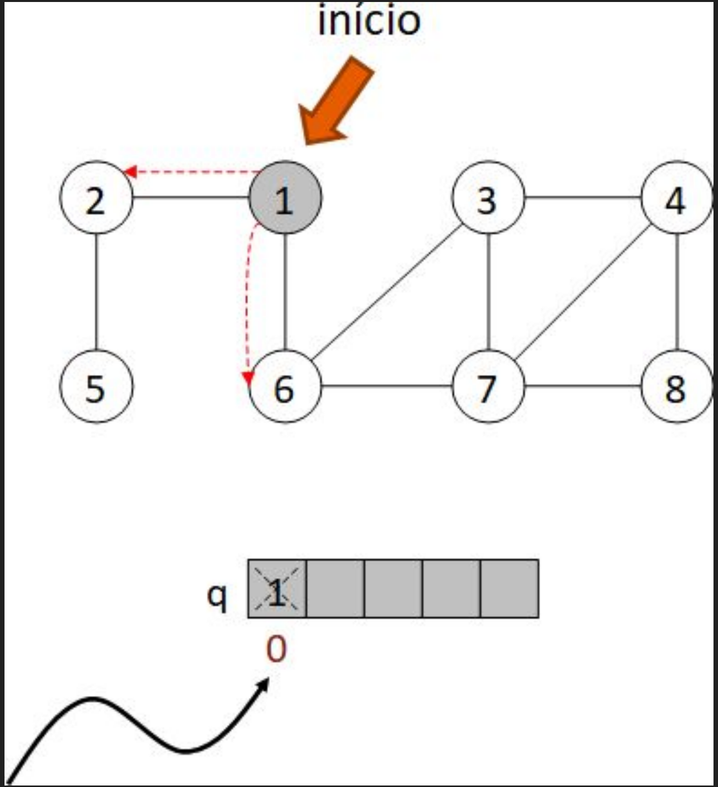
- Estratégia (cont.)
  - Se (u,v) ∈ A e o vértice u é preto, então o vértice v tem que ser cinza ou preto.
    - Todos os vértices adjacentes a um vértice preto já foram visitados
  - Vértices cinza podem ter alguns vértices adjacentes brancos, e eles representam a fronteira entre vértices “descobertos” e não “descobertos”.
  - Fila de vértices em “processamento”

Busca em Largura



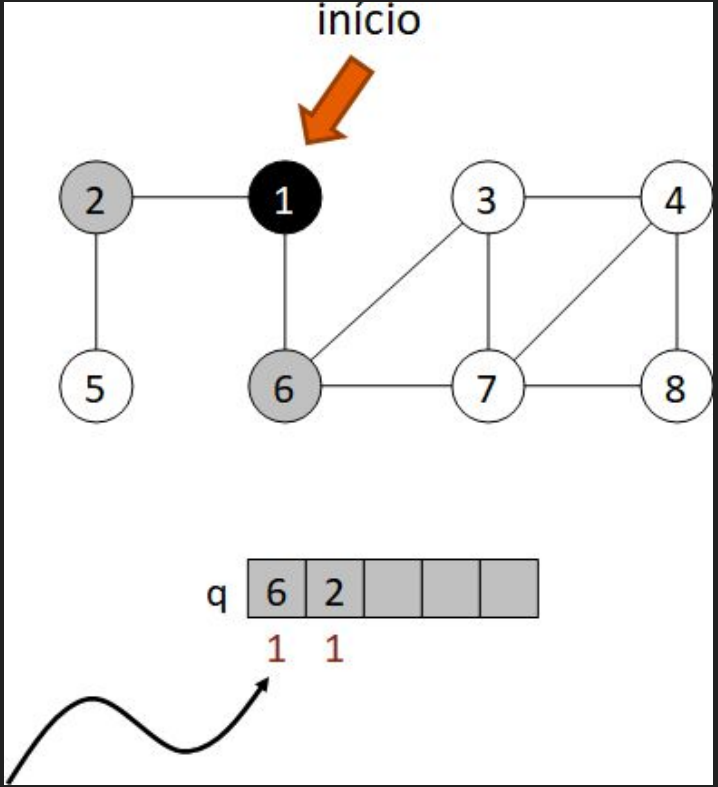
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



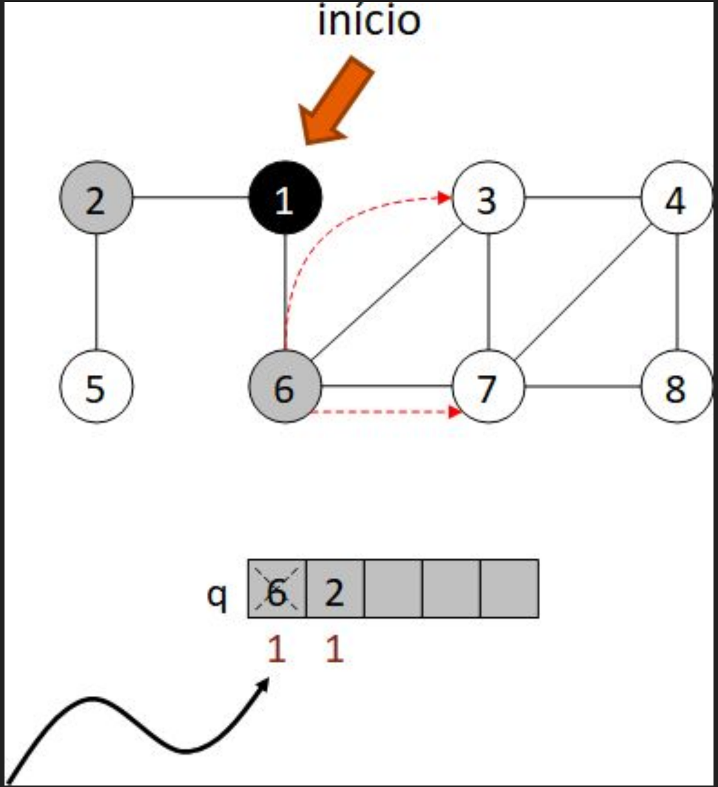
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



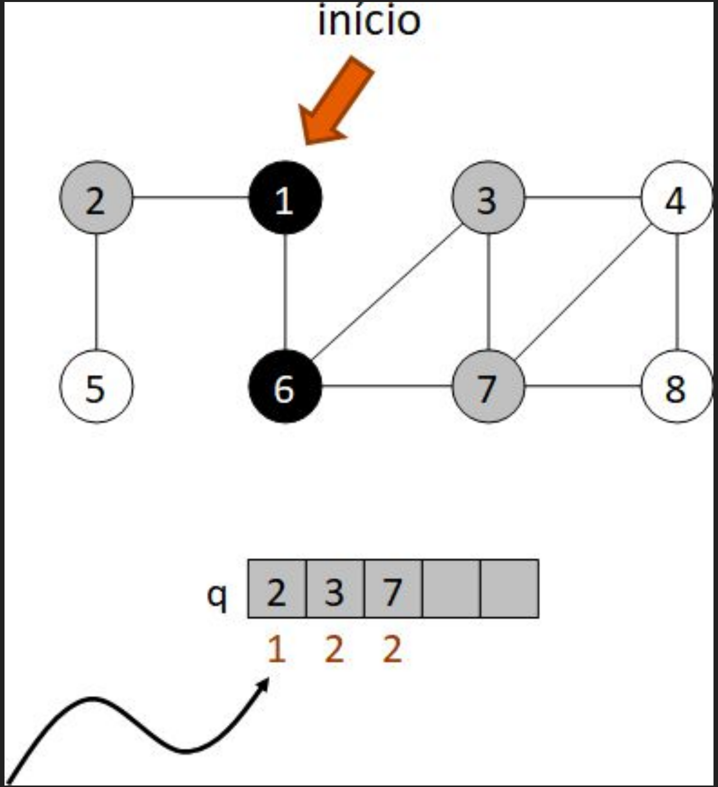
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



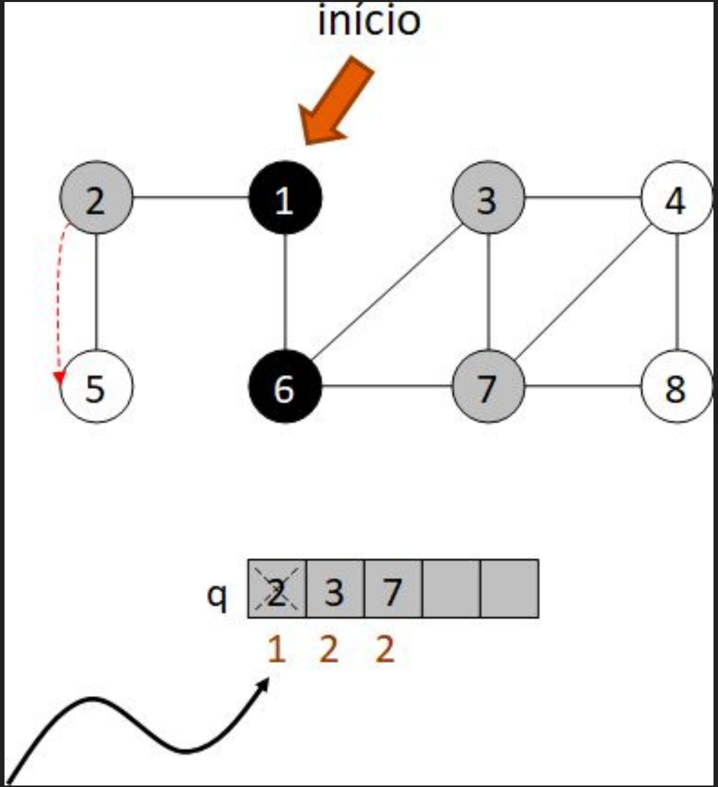
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



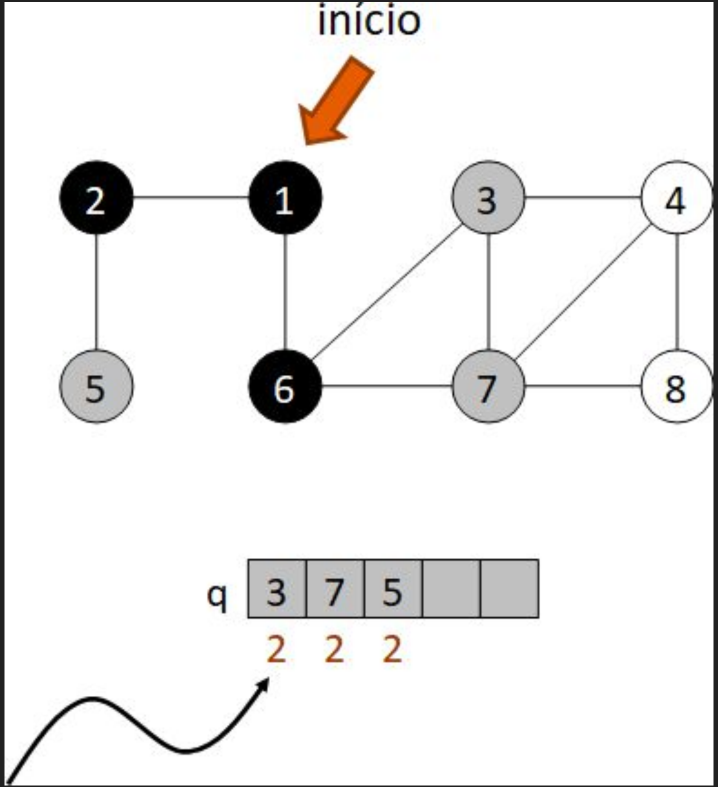
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



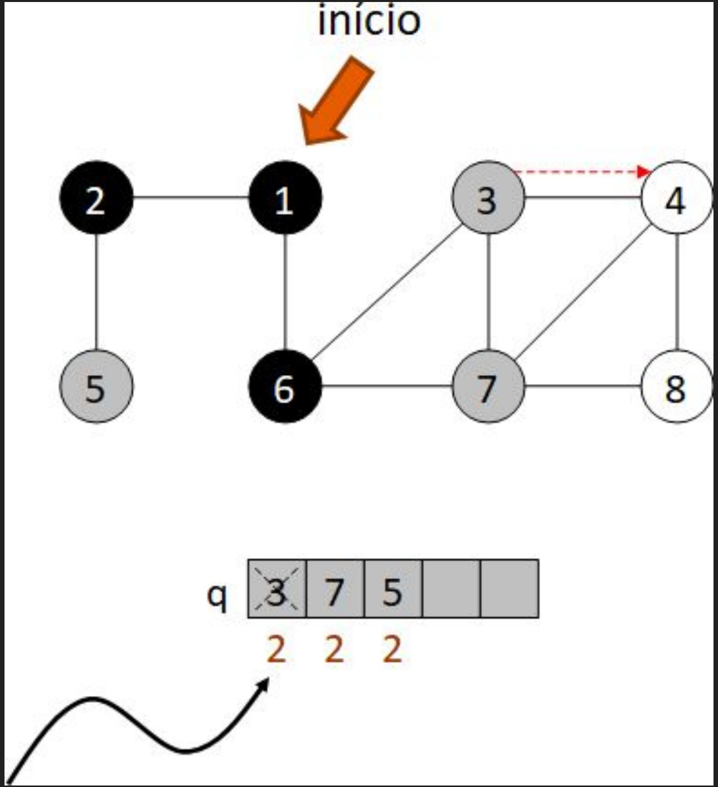
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



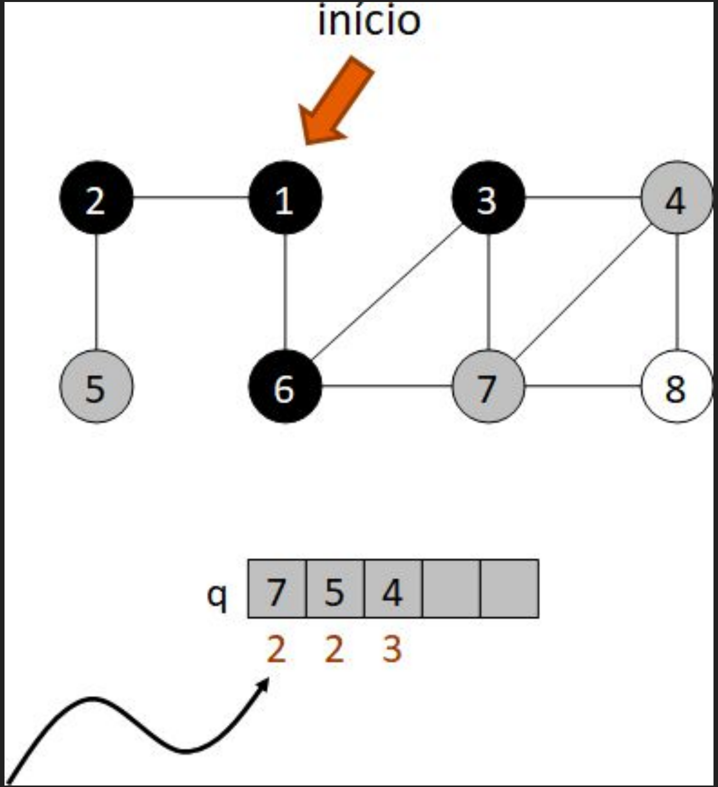
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



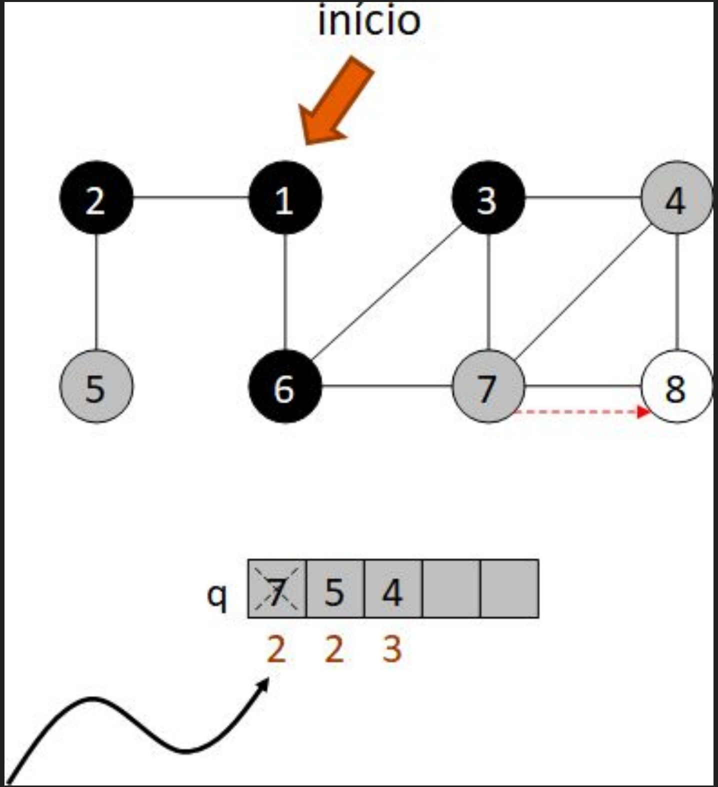
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



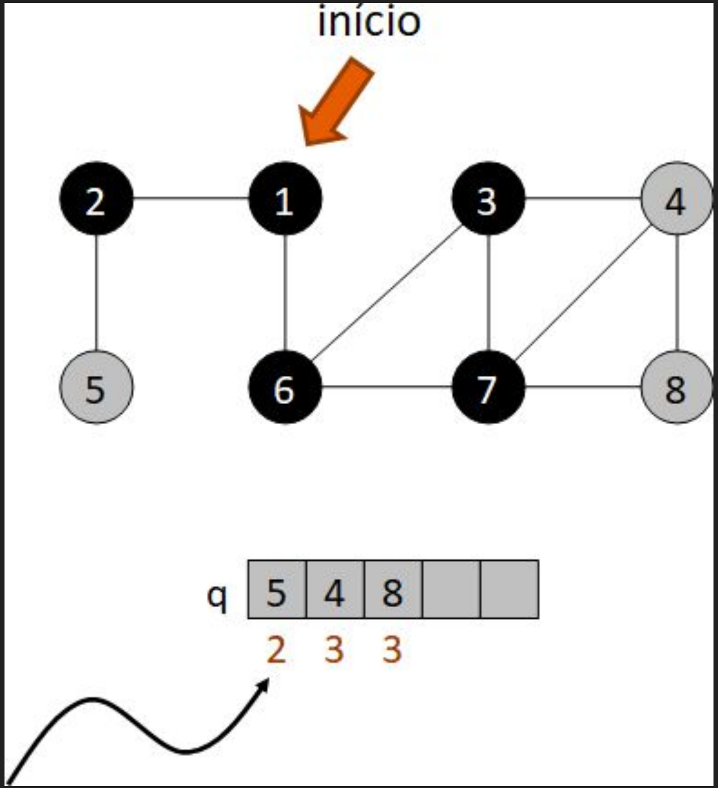
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



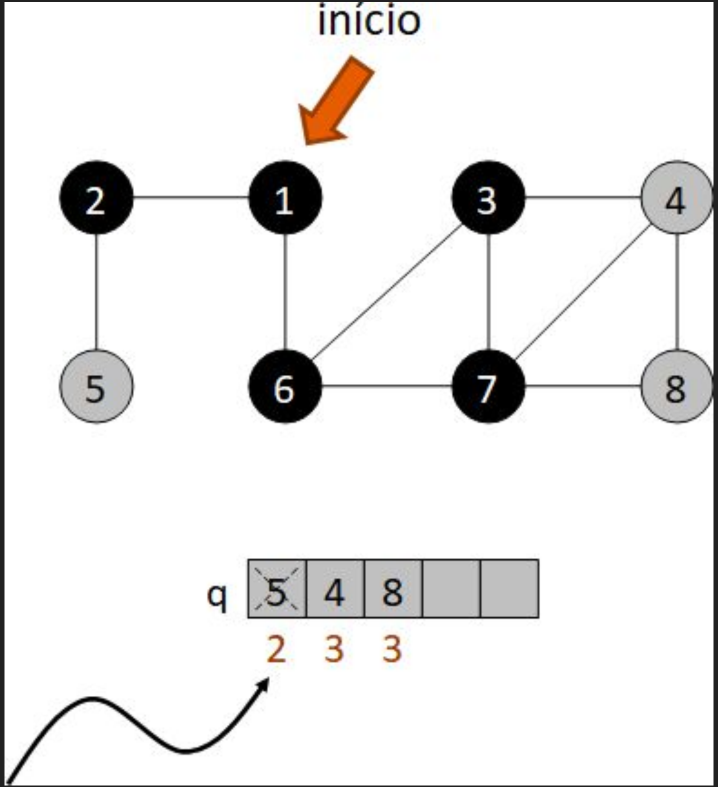
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



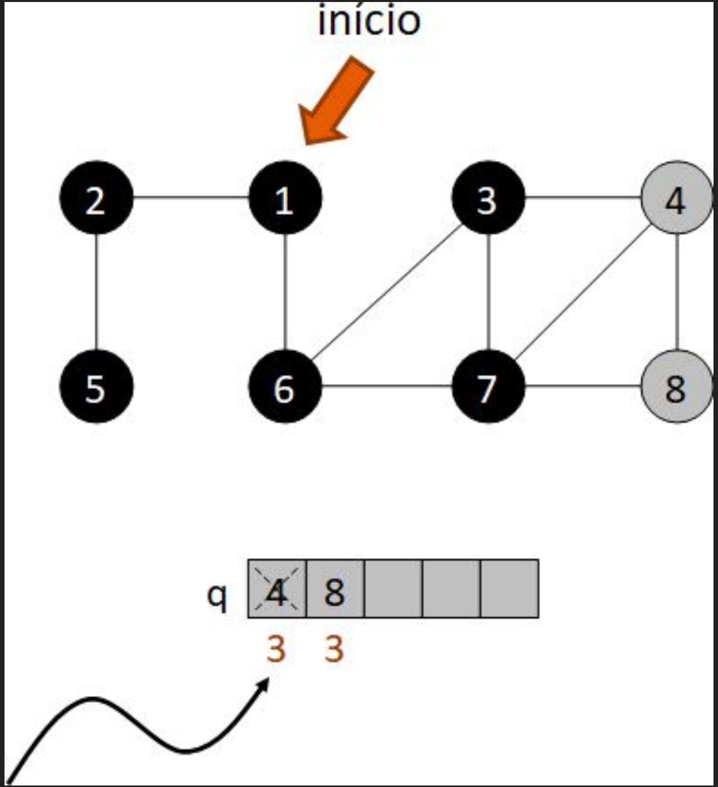
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



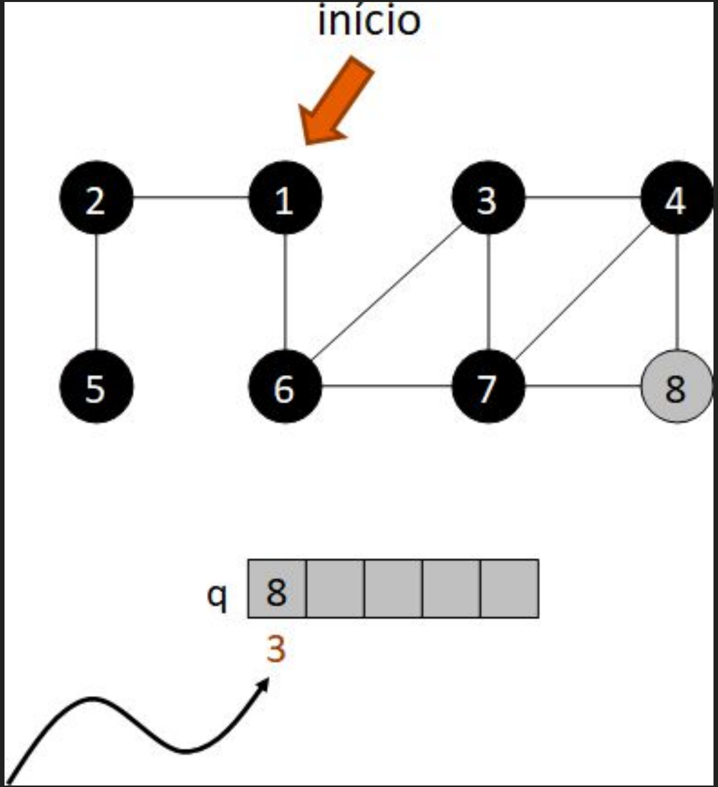
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



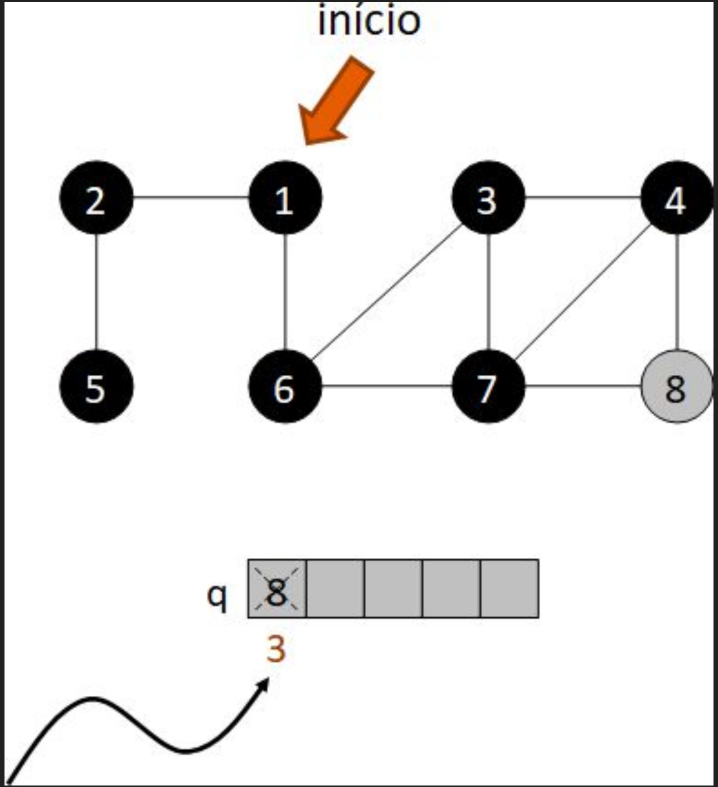
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



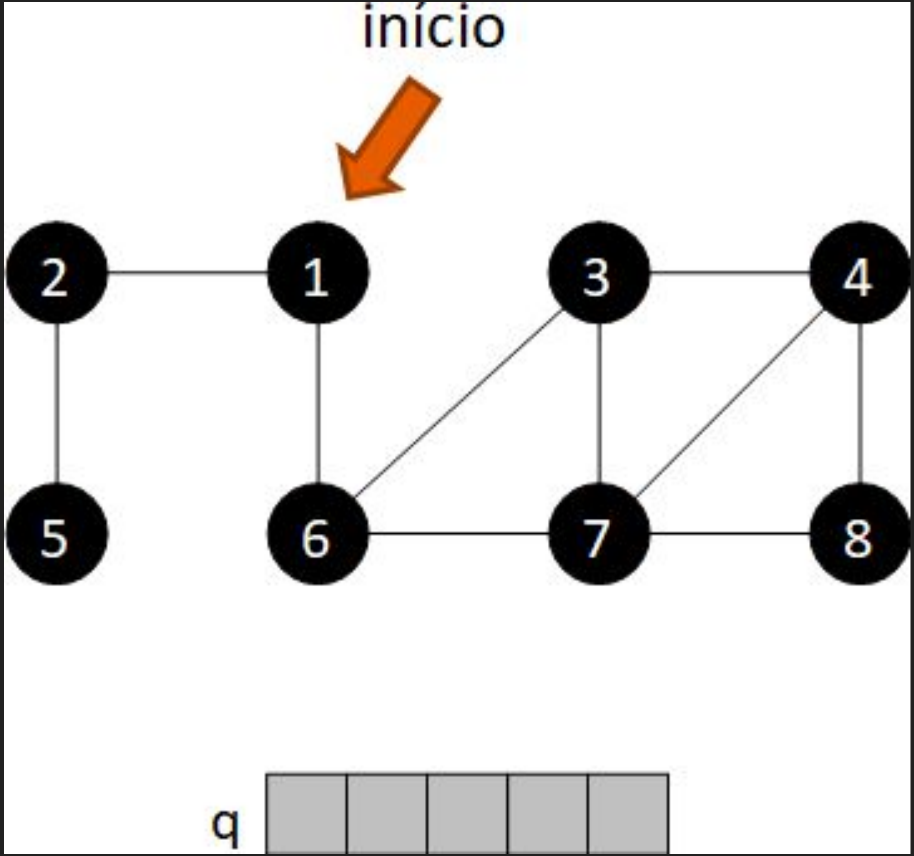
Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



Comprimento do caminho do vértice de início da busca até o vértice na fila

Busca em Largura



Busca em Largura
Implementação simplificada em C

```
void busca_largura(tgrafo *grafo) {
    tvertice v;
    int cor[MAXNUMVERTICES];

    for (v = 0; v < grafo->num_vertices; v++)
        cor[v] = BRANCO;
    for (v = 0; v < grafo->num_vertices; v++)
        if (cor[v] == BRANCO)
            visita_bfs(v, cor, grafo);
}
```

Busca em Largura
Implementação simplificada em C

```
void visita_bfs(tvertice v, int cor[], tgrafo *grafo) {
    tvertice w;
    tapontador p;
    tpeso peso;
    std::queue<tvertice> q;

    cor[v] = CINZA;
    q.push(v);
    while (!q.empty()) {
        v = q.front(); q.pop();
        p = primeiro_adj(v, grafo);
        while (p != NULO) {
            recupera_adj(v, p, &w, &peso, grafo);
            if (cor[w] == BRANCO) {
                cor[w] = CINZA;
                q.push(w);
            }
            p = proximo_adj(v, p, grafo);
        }
        cor[v] = PRETO;
    }
}
```

Vamos Programar

Busca em Largura

- Colorir todos os vértices de branco
  - O(|V|).
- Cada vértice entra na fila q exatamente uma vez
  - |V| iterações.

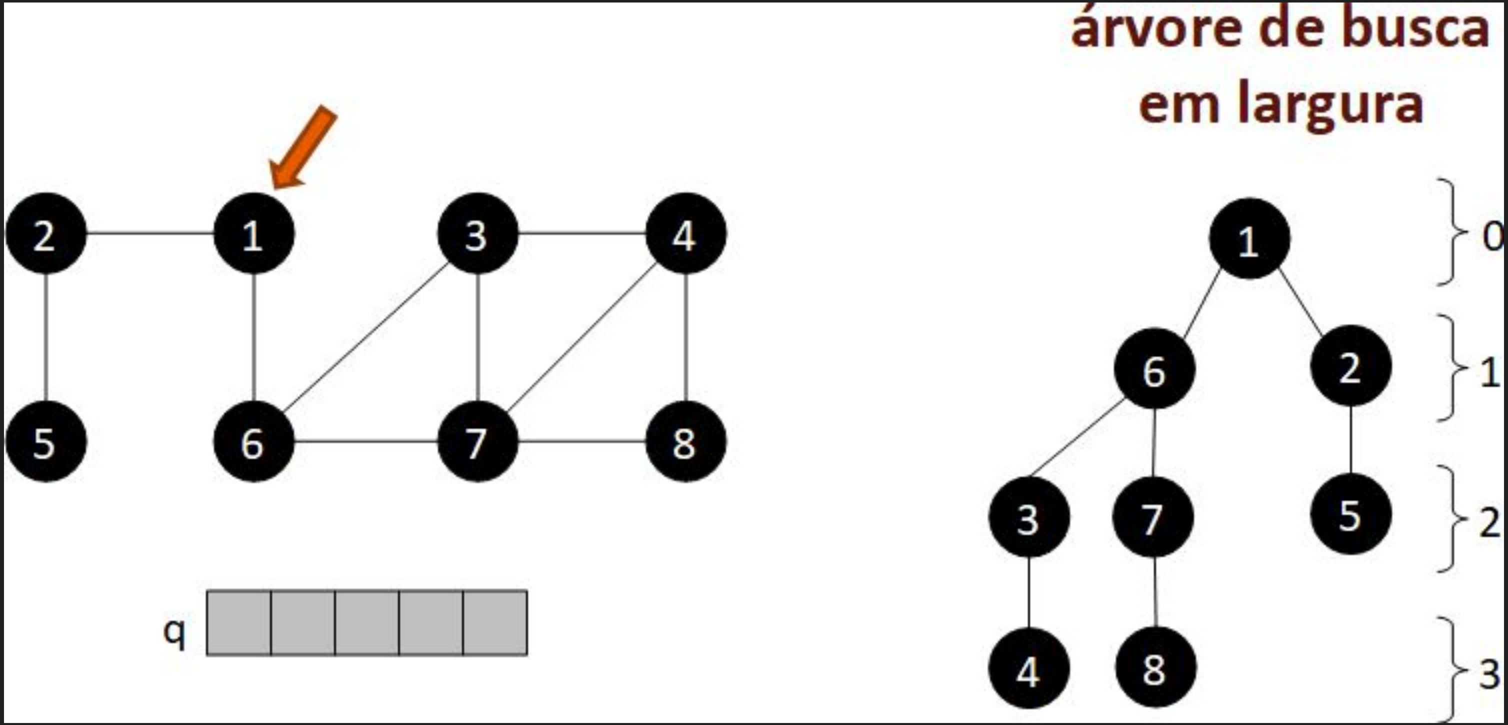
Busca em Largura

- Matriz de adjacências
    - O(|V|) para cada iteração
    - Custo total
        - O(|V|+|V|2).
- Lista de adjacências
    - O(grau(vértice)) para cada iteração
    - Custo total
        - O(|V|+|A|).

Árvore de Busca em Largura

- Representa os caminhos mais curtos entre o vértice de origem e os demais vértices quando todas as arestas possuem o mesmo peso.

Árvore de Busca em Largura



Referências

- WIRTH,N. Algorithms and Data Structures, Englewood Cliffs, Prentice-Hall, 1986.
- CORMEN, H.T.; LEISERSON, C.E.; RIVEST, R.L. Introduction to Algorithms, MIT Press, McGraw-Hill, 1999.
- ZIVIANI, N. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.
- SZWARCFITER,J.L. Grafos e Algoritmos Computacionais. Editora Campus, 1983.
- Van Steen, Maarten. "Graph theory and complex networks." An introduction 144 (2010).
- Gross, Jonathan L., and Jay Yellen. Graph theory and its applications. CRC press, 2005.
- Barabási, A.-L., Pósfai, M. (2016). Network science. Cambridge: Cambridge University Press. ISBN: 9781107076266 1107076269