Estruturas de Dados II

- Aula 9
  - Representação de grafos
    - Matriz de Adjacência
    - Listas de Adjacências
  - Busca em Largura

Prof. DSc Carlos de Salles

www.deinf.ufma.br/~csalles

csalles@deinf.ufma.br

Departamento de Informática
Universidade Federal do Maranhão
DEINF/UFMA

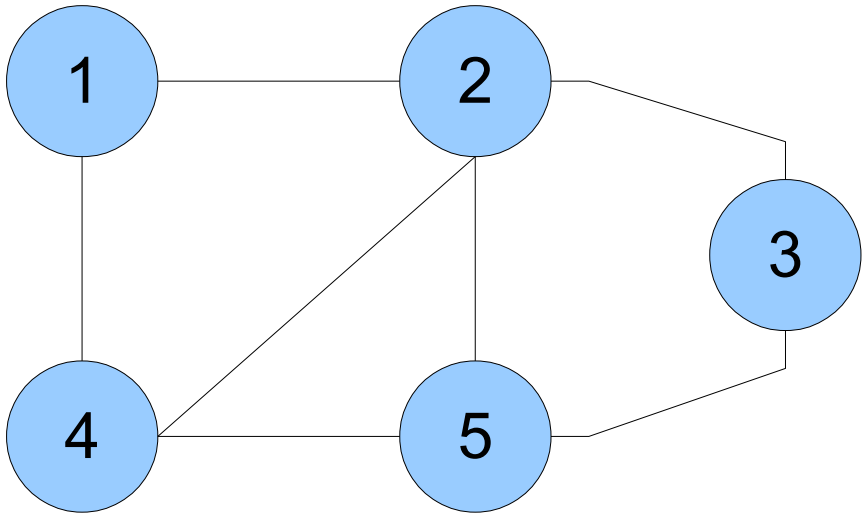
Grafos

- É uma estrutura de dados G, definida como G={V, E}, onde:
  - V = conjunto de nós / vértices
    - Servem para modelar elementos de um problema
  - E = conjunto de arestas
    - Cada aresta liga um par de nós
    - Podem ser ponderadas ou não
    - Podem ser direcionadas ou não

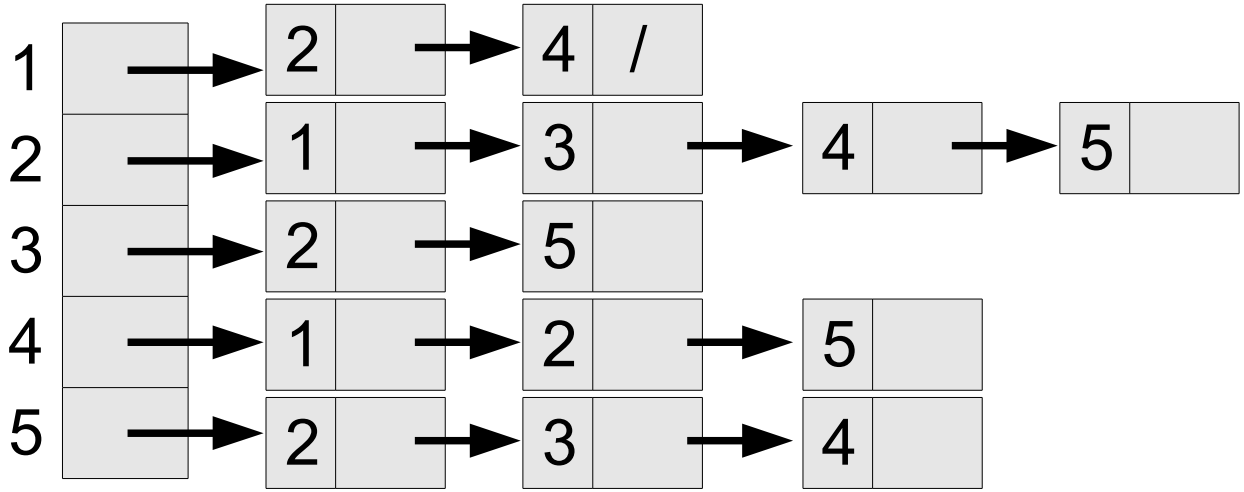
Representação de Grafos

- Matriz de Adjacência
  - Grafos não-ponderados
    - Uma matriz booleana com |V|x|V| elementos indica se há uma aresta entre qualquer par u, v (**u** e **v** são nós do grafo)
  - Grafos ponderados
    - A matriz armazena o peso de cada aresta
- Listas de Adjacência
    - Para cada nó do grafo, há um vetor indicando seus vizinhos

Exemplo de Grafo



	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	1
3	0	1	0	0	1
4	1	1	0	0	1
5	0	1	1	1	0



Matriz de Adjacência

```
function createGraph(N)
   local g = {}
   for i=1,N do
      g[i] = {}
      for j=1,N do
         g[i][j] = false
      end
   end
   return g
end

function createEdge(G, u, v)
   G[u][v] = true
end
function hasEdge(G, u, v)
   return G[u][v]

end
```

Listas de Adjacência

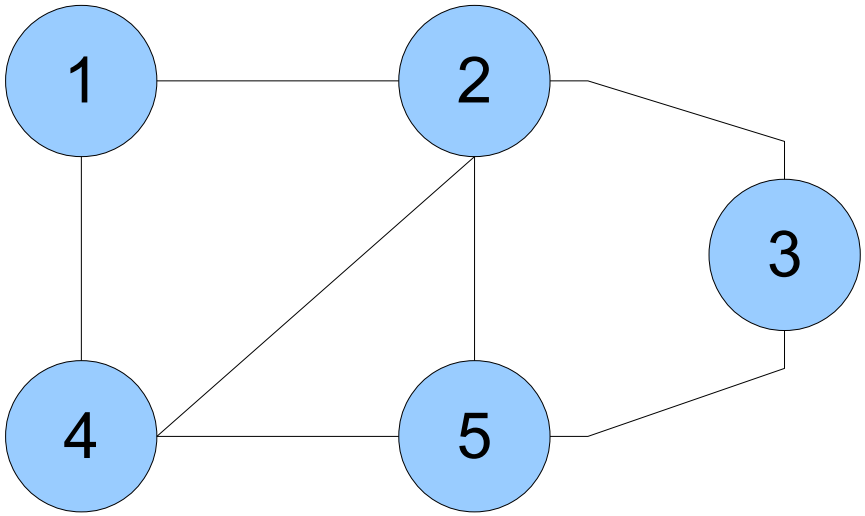
```
function createGraph(N)
   local g = {dg={}, adj={}}
   for i=1,N do g.dg[i]=0; g.adj[i]={} end
   return g
end

function createEdge(G, u, v)
   G.dg[u] = G.dg[u]+1
   G.adj[u][ G.dg[u] ] = v
   G.dg[v] = G.dg[v]+1
   G.adj[v][ G.dg[v] ] = u
end

function hasEdge(G, u, v)
   for i=1,G.dg[u] do
      if G.adj[u][i]==v then return true end
   end
   return false
end
```

Busca em Largura

- Percorre todos os vértices de um grafo G a partir de um vértice de origem ***s*** até descobrir cada vértice acessível a partir de ***s***
- Visita primeiro os vértices mais próximos de ***s***
- Em inglês: BFS (*breadth first search*)

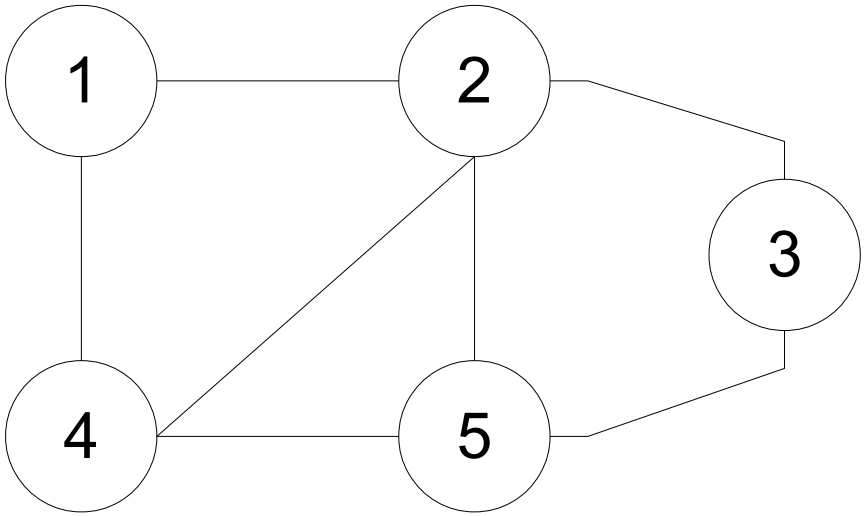
Busca em Largura

- Inicia pintando todos de branco (não-visitados)



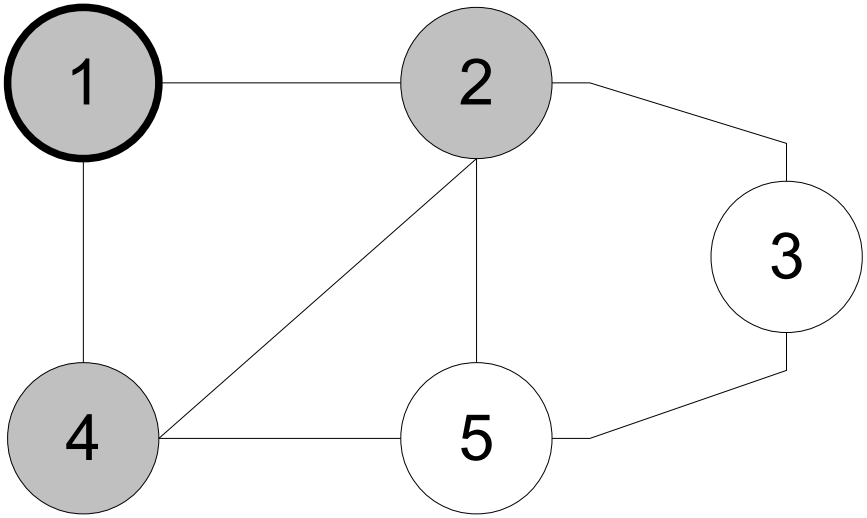
Busca em Largura

- Suponha o vértice inicial ***s = 1***



Busca em Largura

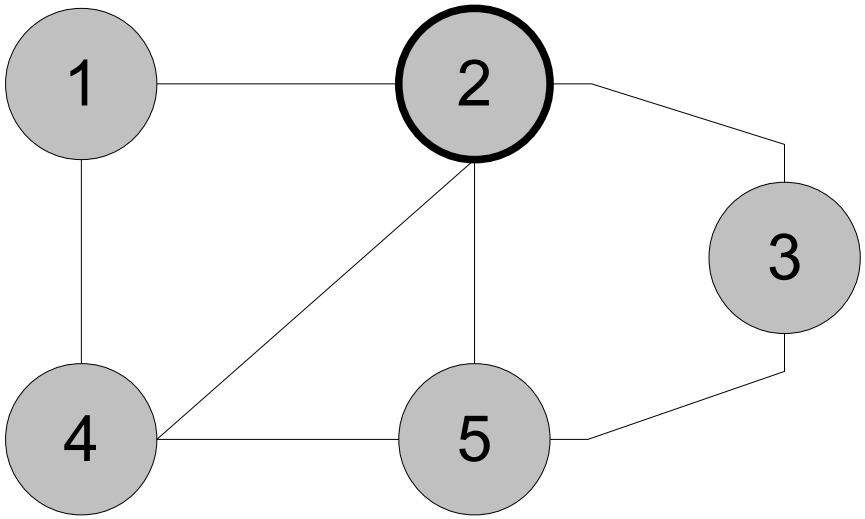
- BFS(1)



- Fila: 2 | 4

Busca em Largura

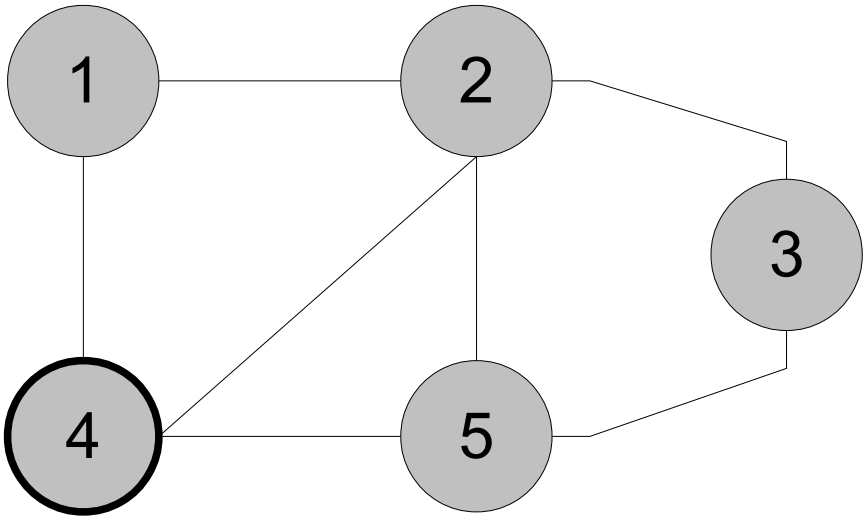
- BFS(2)



- Fila: 4 | 3 | 5

Busca em Largura

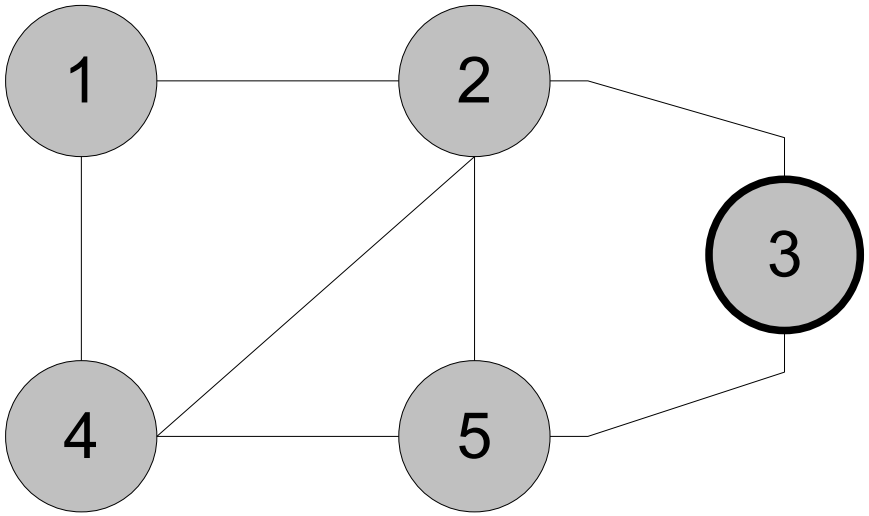
- BFS(4)



- Fila: 3 | 5

Busca em Largura

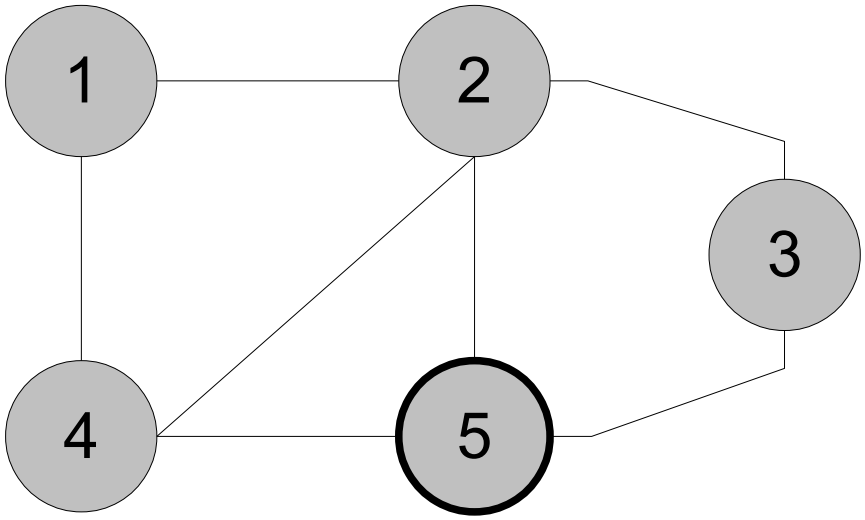
- BFS(3)



- Fila: 5

Busca em Largura

- BFS(5)



- Fila: /

Busca em Largura

```
function BFS(G, s)
   local WHITE, GRAY = 1,2
   local color = {}
   for i=1,#G.dg do color[i]=WHITE end
   Q = createQueue()
   insertQueue(Q, s)
   color[s] = GRAY
   while not emptyQueue(Q) do
      u = removeQueue(Q)
      for i=1,G.dg[u] do
         v = G.adj[u][i]
         if color[v]==WHITE then
            insertQueue(v)
            color[v] = GRAY
         end
      end
   end
end
```