Grafos – Busca em Largura - BFS

Estrutura de Dados Avançada — QXD0015

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará

1º semestre/2023

Busca em grafos

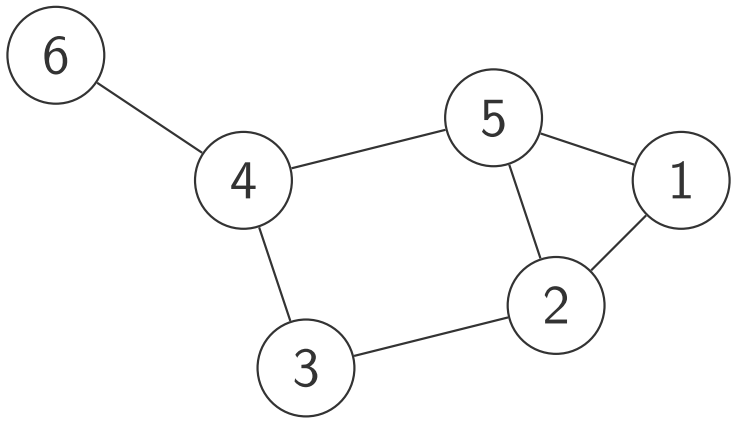
Problema fundamental em grafos:

> Como explorar um grafo de forma sistemática?

- Muitas aplicações são abstraídas como problemas de busca.
- Muitos algoritmos utilizam fundamentos similares.

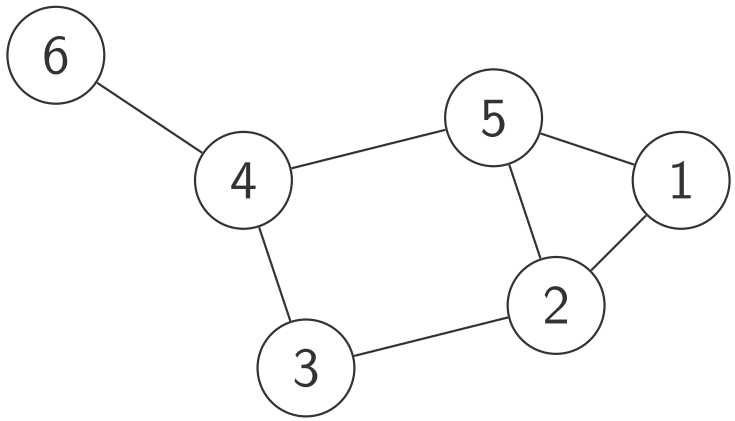
Busca em grafos

- Como saber se existe caminho entre dois vértices, de maneira eficiente?



Busca em grafos

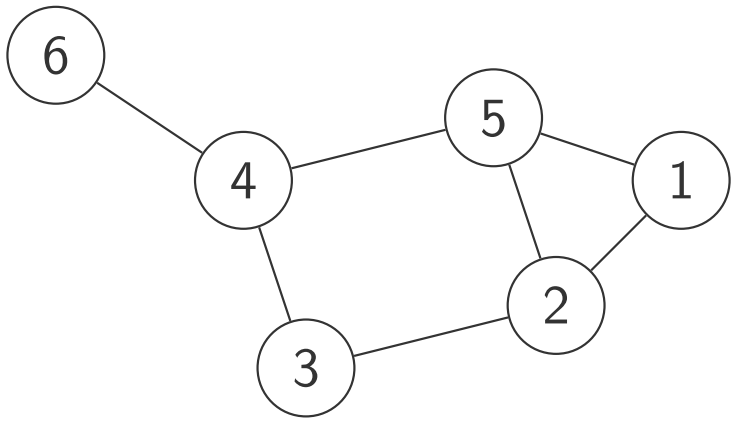
- Como saber se existe caminho entre dois vértices, de maneira eficiente?



- **Ideia:** durante a busca no grafo, evitar passar por um vértice mais de uma vez
  - quando chegar num vértice pela primeira vez, marcá-lo.

Busca em grafos

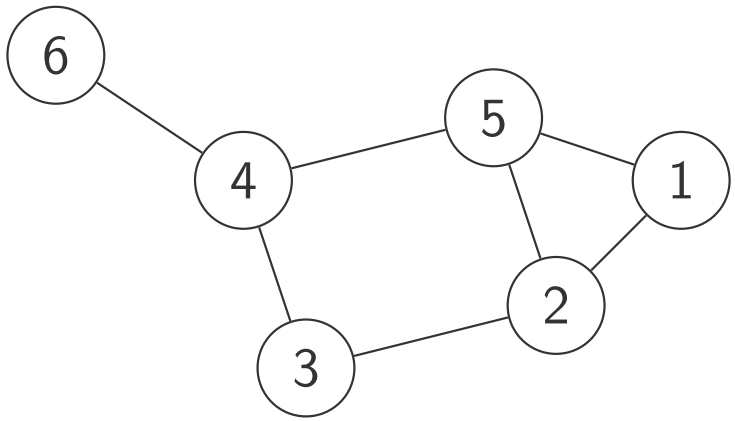
- Como saber se existe caminho entre dois vértices, de maneira eficiente?



- **Ideia:** durante a busca no grafo, evitar passar por um vértice mais de uma vez
  - quando chegar num vértice pela primeira vez, marcá-lo.
  - ao sair de um vértice u e chegar, por meio de uma aresta (u, v), num vértice v nunca visto antes, vamos de alguma forma marcar essa aresta como especial.

Busca em grafos

- Como saber se existe caminho entre dois vértices, de maneira eficiente?



- **Ideia:** durante a busca no grafo, evitar passar por um vértice mais de uma vez
  - quando chegar num vértice pela primeira vez, marcá-lo.
  - ao sair de um vértice u e chegar, por meio de uma aresta (u, v), num vértice v nunca visto antes, vamos de alguma forma marcar essa aresta como especial.
  - É preciso garantir não só que todos os vértices possíveis de serem alcançados foram alcançados, mas também garantir que todas as arestas incidentes nesses vértices foram analisadas.

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:
  - **Não-explorada:** ainda não considerada pela busca.

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:
  - **Não-explorada:** ainda não considerada pela busca.
  - **Explorada:** já foi considerada pela busca.

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:
  - **Não-explorada:** ainda não considerada pela busca.
  - **Explorada:** já foi considerada pela busca.
  - **Aresta de árvore:** assim que ela foi considerada pela busca, um de seus extremos já tinha sido descoberto, mas o outro ainda não.

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:
  - **Não-explorada:** ainda não considerada pela busca.
  - **Explorada:** já foi considerada pela busca.
  - **Aresta de árvore:** assim que ela foi considerada pela busca, um de seus extremos já tinha sido descoberto, mas o outro ainda não.

- Possíveis estágios da marcação de um vértice:

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:
  - **Não-explorada:** ainda não considerada pela busca.
  - **Explorada:** já foi considerada pela busca.
  - **Aresta de árvore:** assim que ela foi considerada pela busca, um de seus extremos já tinha sido descoberto, mas o outro ainda não.

- Possíveis estágios da marcação de um vértice:
  - **Não-descoberto:** vértice que ainda não foi encontrado pela busca.

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:
  - **Não-explorada:** ainda não considerada pela busca.
  - **Explorada:** já foi considerada pela busca.
  - **Aresta de árvore:** assim que ela foi considerada pela busca, um de seus extremos já tinha sido descoberto, mas o outro ainda não.

- Possíveis estágios da marcação de um vértice:
  - **Não-descoberto:** vértice que ainda não foi encontrado pela busca.
  - **Descoberto:** vértice foi descoberto (visitado pela primeira vez).

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:
  - **Não-explorada:** ainda não considerada pela busca.
  - **Explorada:** já foi considerada pela busca.
  - **Aresta de árvore:** assim que ela foi considerada pela busca, um de seus extremos já tinha sido descoberto, mas o outro ainda não.

- Possíveis estágios da marcação de um vértice:
  - **Não-descoberto:** vértice que ainda não foi encontrado pela busca.
  - **Descoberto:** vértice foi descoberto (visitado pela primeira vez).
  - **Descoberto e não-explorado:** vértice foi descoberto, mas possui arestas incidentes que ainda não foram exploradas.

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:
  - **Não-explorada:** ainda não considerada pela busca.
  - **Explorada:** já foi considerada pela busca.
  - **Aresta de árvore:** assim que ela foi considerada pela busca, um de seus extremos já tinha sido descoberto, mas o outro ainda não.

- Possíveis estágios da marcação de um vértice:
  - **Não-descoberto:** vértice que ainda não foi encontrado pela busca.
  - **Descoberto:** vértice foi descoberto (visitado pela primeira vez).
  - **Descoberto e não-explorado:** vértice foi descoberto, mas possui arestas incidentes que ainda não foram exploradas.
  - **Explorado:** vértice foi descoberto e todas as arestas incidentes no vértice foram exploradas e seus vizinhos foram descobertos.

Busca em grafos – Estratégia

- Definir vértice inicial (origem da busca)

- Possíveis tipos de arestas durante a busca:
  - **Não-explorada:** ainda não considerada pela busca.
  - **Explorada:** já foi considerada pela busca.
  - **Aresta de árvore:** assim que ela foi considerada pela busca, um de seus extremos já tinha sido descoberto, mas o outro ainda não.

- Possíveis estágios da marcação de um vértice:
  - **Não-descoberto:** vértice que ainda não foi encontrado pela busca.
  - **Descoberto:** vértice foi descoberto (visitado pela primeira vez).
  - **Descoberto e não-explorado:** vértice foi descoberto, mas possui arestas incidentes que ainda não foram exploradas.
  - **Explorado:** vértice foi descoberto e todas as arestas incidentes no vértice foram exploradas e seus vizinhos foram descobertos.

- Essa estratégia resulta em um algoritmo genérico para buscas em grafos.

Algoritmo de busca genérico

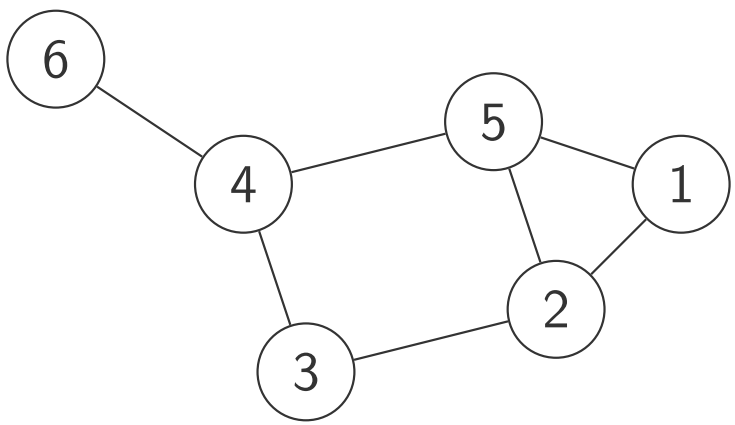
Dado grafo $G = (V, E)$ (direcionado ou não-direcionado):

(1) Marcar todos os vértices como `não-descobertos`

(2) Escolher um vértice inicial e marcá-lo `descoberto`

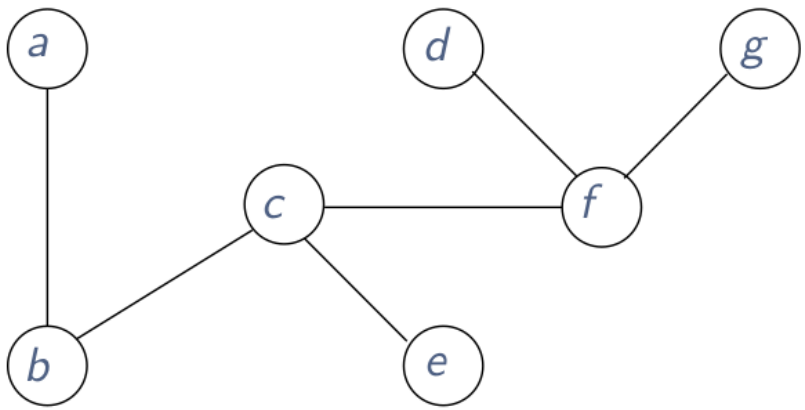
(3) **Enquanto** houver vértice `descoberto` e `não-explorado`:

  (a) Selecionar vértice descoberto e não-explorado u
  (b) Considerar aresta não-explorada (u, v)
  (c) Se v for `não-descoberto`, marcar v como `descoberto` e marcar aresta (u, v) como **aresta de árvore**.
  (d) Se não houver mais arestas incidentes a u a serem exploradas, marcar u como **explorado**.



Representação de árvores

Uma árvore enraizada é uma árvore com um vértice especial chamado raiz.



Representação de árvores

Uma árvore direcionada com raiz r é um grafo direcionado acíclico $T = (V, E)$ tal que:

Representação de árvores

Uma árvore direcionada com raiz r é um grafo direcionado acíclico $T = (V, E)$ tal que:

1. $d^-(r) = 0$,

Representação de árvores

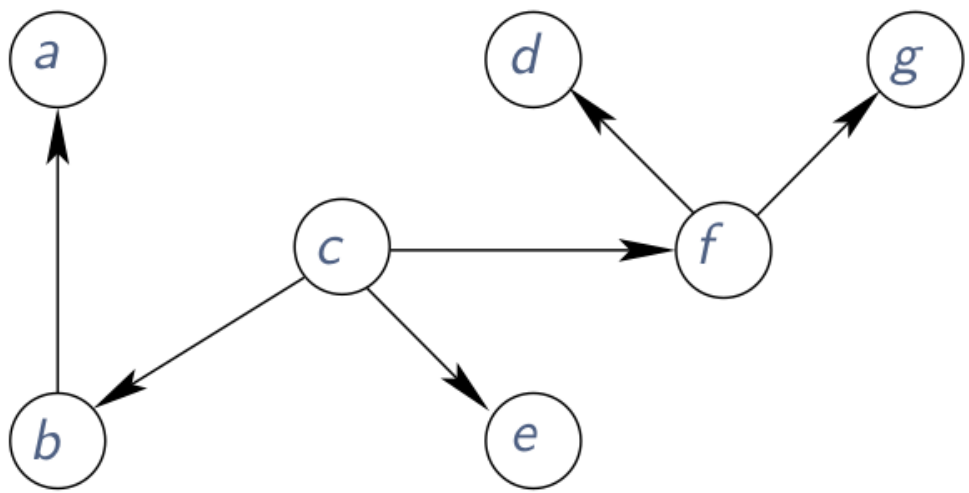
Uma árvore direcionada com raiz r é um grafo direcionado acíclico $T = (V, E)$ tal que:

1. $d^-(r) = 0$,
2. $d^-(v) = 1$ para todo $v \in V \setminus \{r\}$.

Representação de árvores

Uma árvore direcionada com raiz r é um grafo direcionado acíclico $T = (V, E)$ tal que:

1. $d^-(r) = 0$,
2. $d^-(v) = 1$ para todo $v \in V \setminus \{r\}$.



raiz c

Representação de árvores

Representar uma árvore enraizada com um vetor de predecessores π.

v	a	b	c	d	e	f	g
$\pi[v]$	b	c	NIL	f	c	c	f

Representação de árvores

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Representar uma árvore enraizada com um vetor de predecessores π.

v	a	b	c	d	e	f	g
$\pi[v]$	b	c	NIL	f	c	c	f

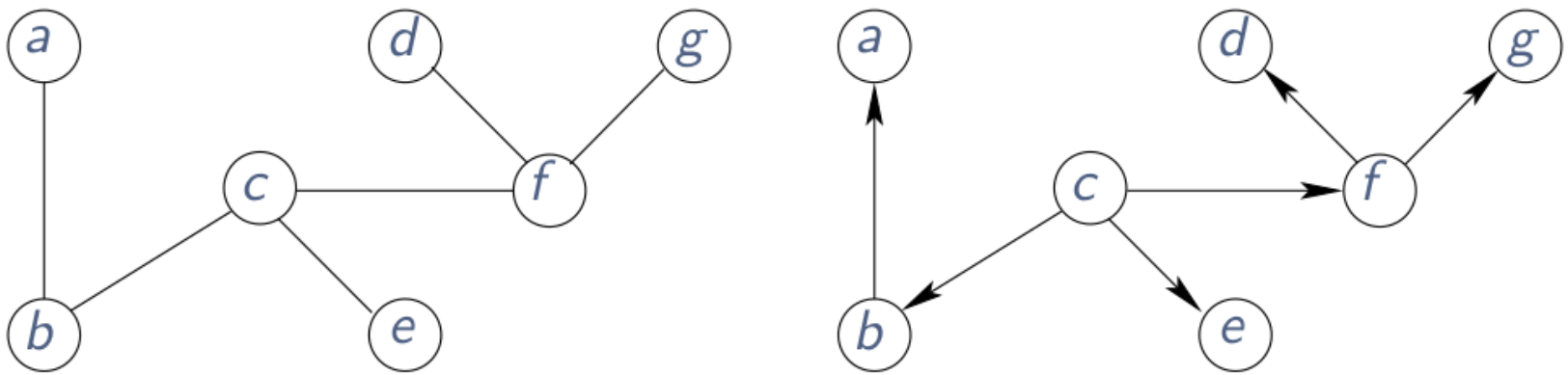
- usamos o símbolo NIL para indicar a ausência

Representação de árvores

Representar uma árvore enraizada com um vetor de predecessores π.

v	a	b	c	d	e	f	g
$\pi[v]$	b	c	NIL	f	c	c	f

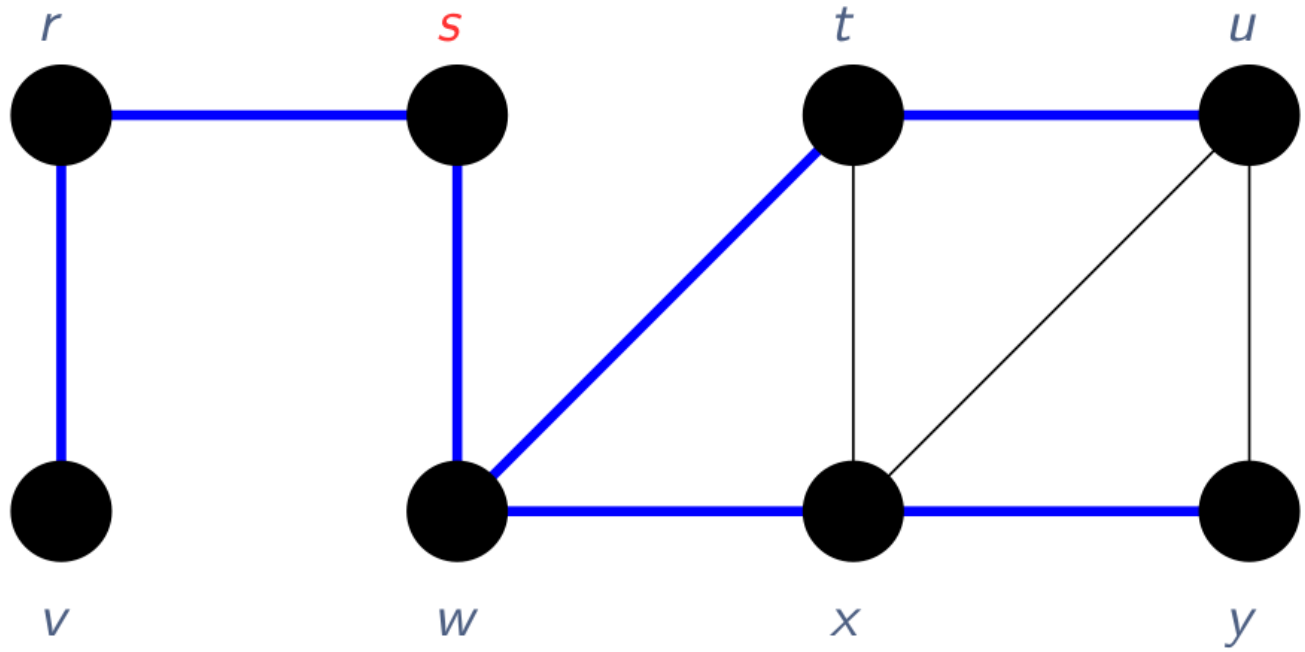
- usamos o símbolo NIL para indicar a ausência



raiz c

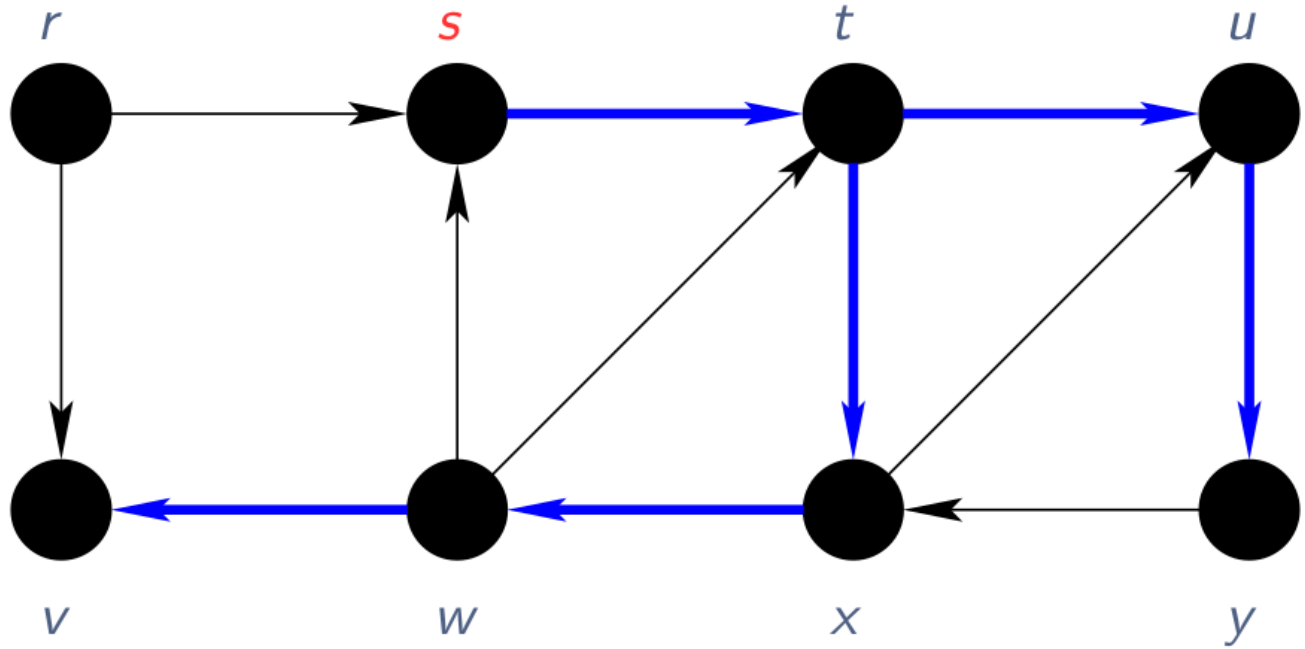
Exemplo com grafo não direcionado

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



v	r	s	t	u	v	w	x	y
$\pi[v]$	s	NIL	w	t	r	s	w	x

Exemplo com grafo direcionado



v	r	s	t	u	v	w	x	y
$\pi[v]$	NIL	NIL	s	t	w	x	t	u

Imprimindo caminho entre dois vértices

Imprimindo caminho entre dois vértices

Algoritmo recursivo para imprimir caminho de s a v no grafo:

`print-path`(π, s, v)

1. **se** $v == s$ **então**
2. `imprima` s
3. **senão se** $\pi[v] ==$ `NIL` **então**
4. `imprima` não existe caminho de s a v
5. **senão**
6. `print-path`$(\pi, s, \pi[v])$
7. `imprima` v

Ordenação da Exploração

O algoritmo genérico não estabelece ordem de visita dos vértices e arestas.

Ordenação da Exploração

O algoritmo genérico não estabelece ordem de visita dos vértices e arestas.

- A ordem de visita das arestas geralmente é aleatória e depende da representação do grafo no computador.

Ordenação da Exploração

O algoritmo genérico não estabelece ordem de visita dos vértices e arestas.

- A ordem de visita das arestas geralmente é aleatória e depende da representação do grafo no computador.
- Existem duas abordagens principais para a ordem de visita dos vértices:

Ordenação da Exploração

O algoritmo genérico não estabelece ordem de visita dos vértices e arestas.

- A ordem de visita das arestas geralmente é aleatória e depende da representação do grafo no computador.
- Existem duas abordagens principais para a ordem de visita dos vértices:
  - Explorar o vértice descoberto **"mais antigo"**
    - Busca em Largura (Breadth-First Search – BFS)

Ordenação da Exploração

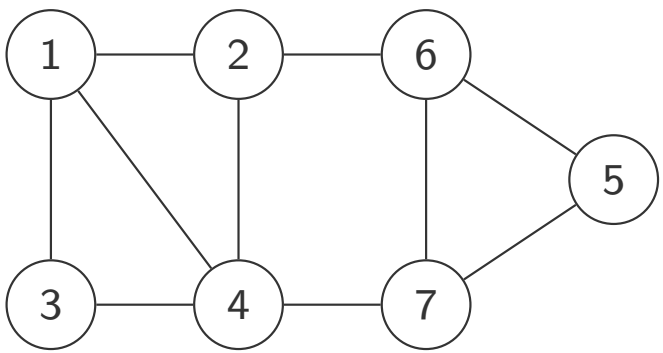
O algoritmo genérico não estabelece ordem de visita dos vértices e arestas.

- A ordem de visita das arestas geralmente é aleatória e depende da representação do grafo no computador.
- Existem duas abordagens principais para a ordem de visita dos vértices:
  - Explorar o vértice descoberto **"mais antigo"**
    - Busca em Largura (Breadth-First Search – BFS)
  - Explorar o vértice descoberto **"mais recente"**
    - Busca em Profundidade (Depth-First Search – DFS)
    - Já vimos busca em profundidade quando estudamos árvores: pré-ordem, pós-ordem e ordem simétrica.

Busca em largura

Busca em largura (BFS)

- Exercício: Explorar o grafo abaixo usando a regra: vértices descobertos mais antigos devem ser explorados primeiro.

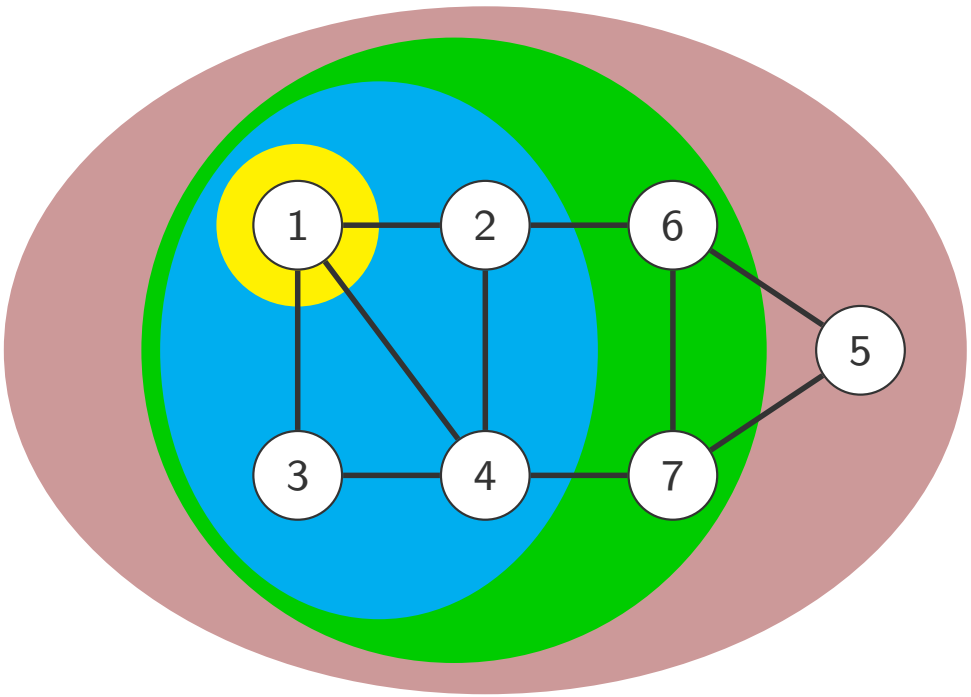


  - Origem da busca: vértice 1.
  - Considere que as arestas incidentes em um vértice v são exploradas em ordem crescente dos vértices adjacentes.

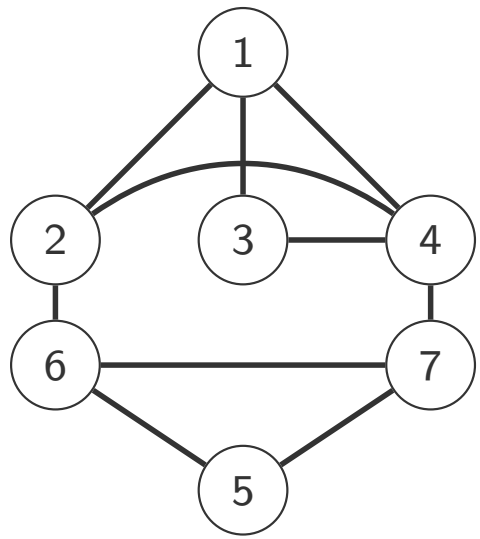
Interpretação da busca em largura

- A busca em largura é realizada em níveis.

- Primeiro a raiz é descoberta no nível L_1. Depois todos os vértices vizinhos da raiz são descobertos no nível L_2. Depois todos os vértices vizinhos dos vizinhos da raiz são descobertos no nível L_3, e assim por diante.



Grafo G

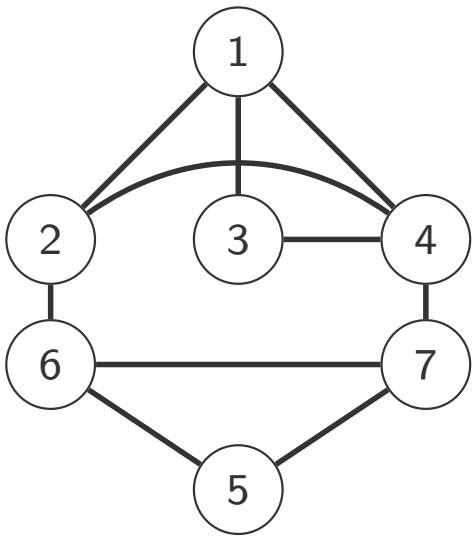


Grafo G redesenhado

Definição: Distância entre dois vértices

- A distância entre dois vértices u e v de um grafo G é o comprimento do menor caminho entre u e v em G, e é denotada por $d(u, v)$.

- Quando não há caminho entre u e v, definimos $d(u, v) = \infty$.

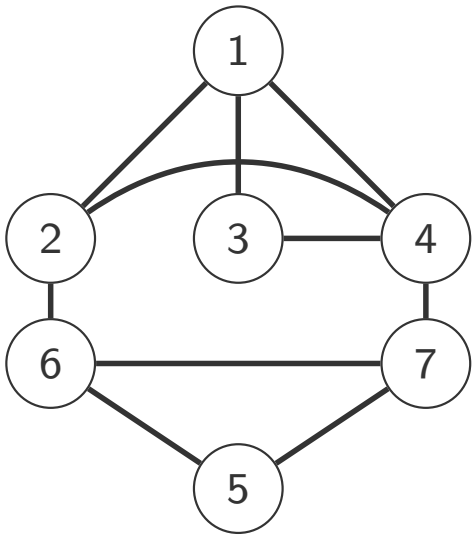


Grafo G

- $d(1, 2) = d(1, 3) = d(1, 4) = 1$
- $d(1, 6) = d(1, 7) = 2$
- $d(1, 5) = 3$

Níveis e Distâncias

- Qual a relação entre níveis e distâncias?

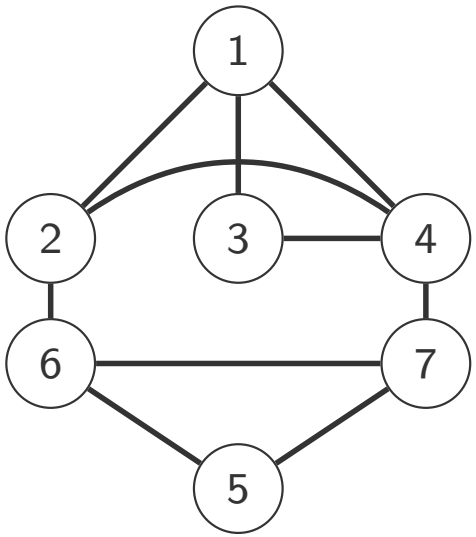


Grafo G

- $d(1,2) = d(1,2) = d(1,4) = 1$
- $d(1,6) = d(1,7) = 2$
- $d(1,5) = 3$

Níveis e Distâncias

- Qual a relação entre níveis e distâncias?



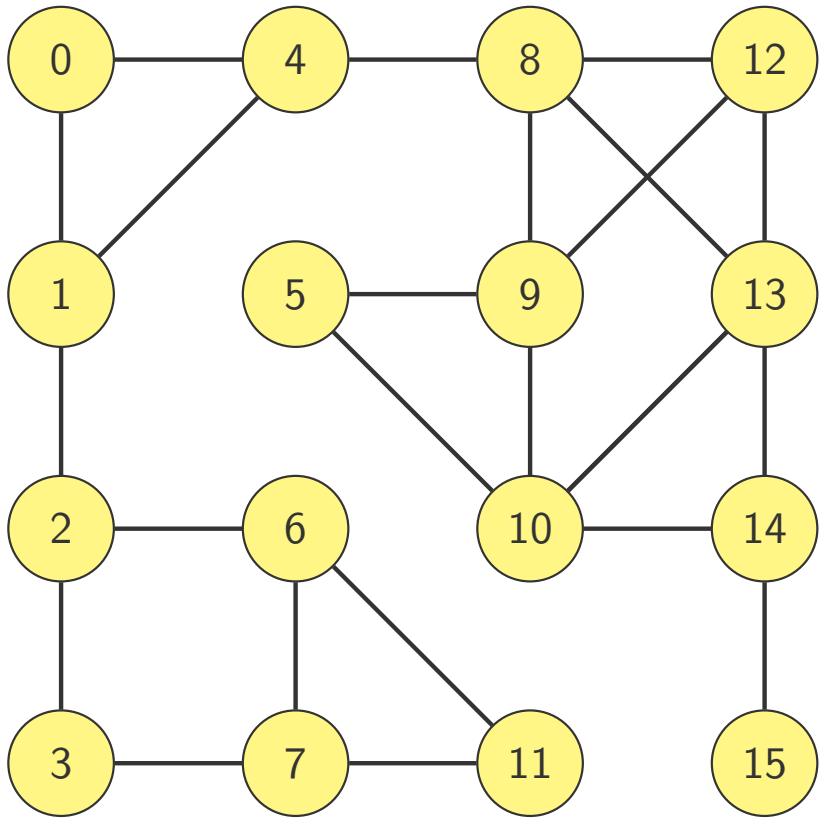
Grafo G

- $d(1,2) = d(1,2) = d(1,4) = 1$
- $d(1,6) = d(1,7) = 2$
- $d(1,5) = 3$

- Vértices pertencentes à camada L_i têm distância i da origem.

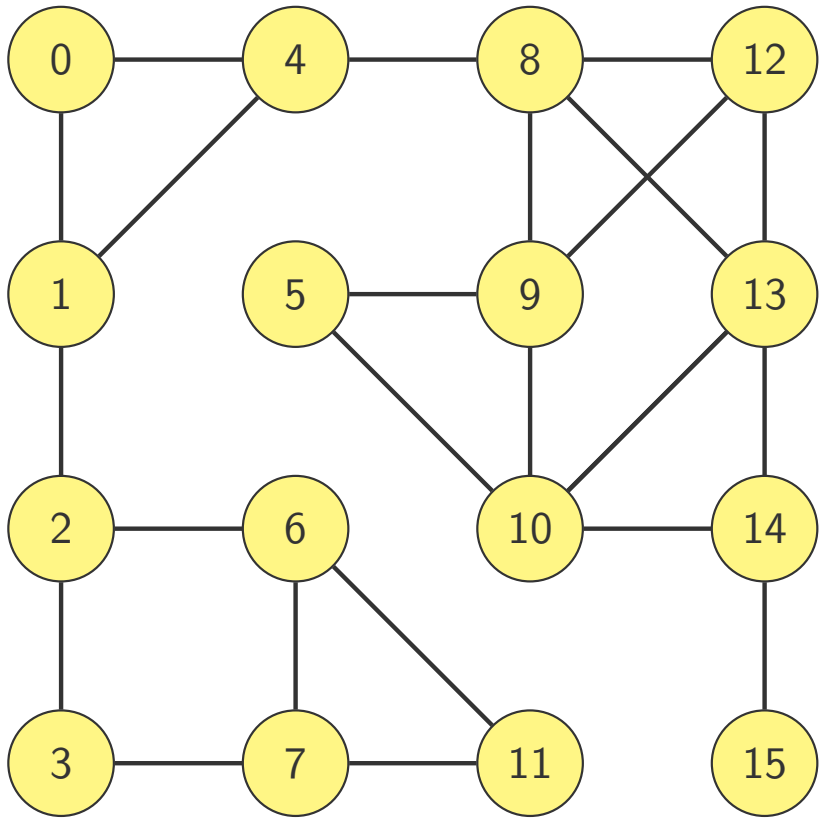
A Busca em Largura calcula distâncias da raiz aos demais vértices!

BFS usa fila



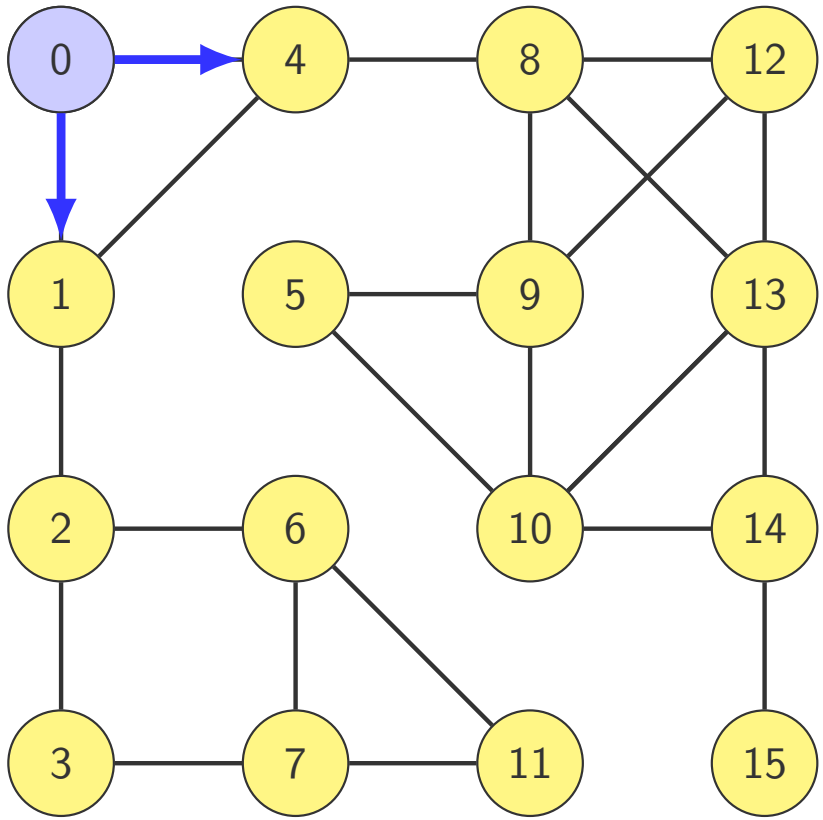
Fila

BFS usa fila



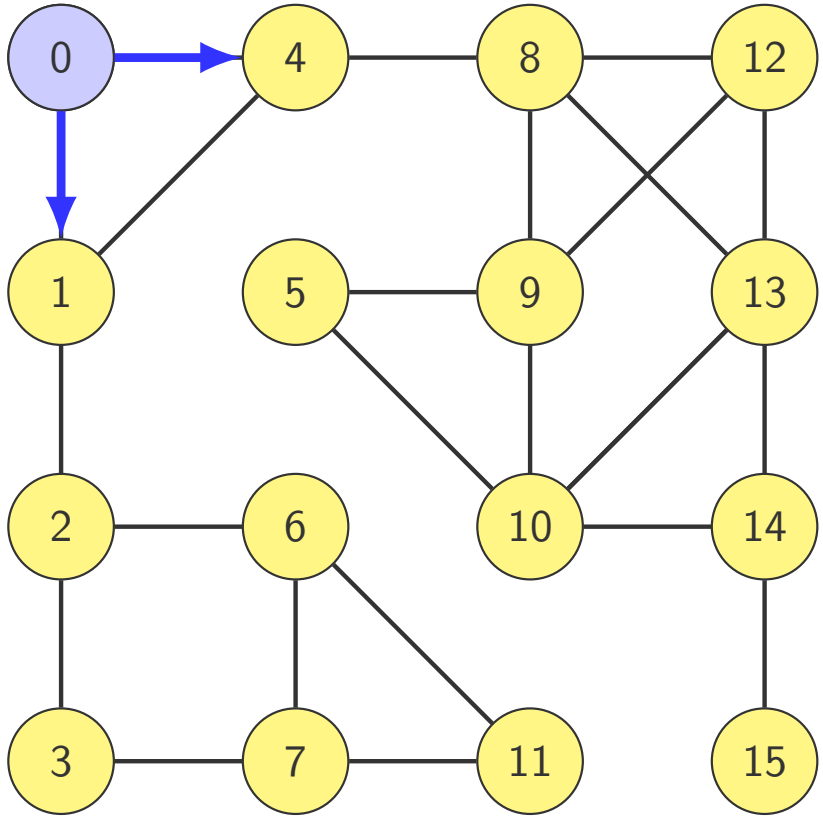
Fila | 0

BFS usa fila



Fila

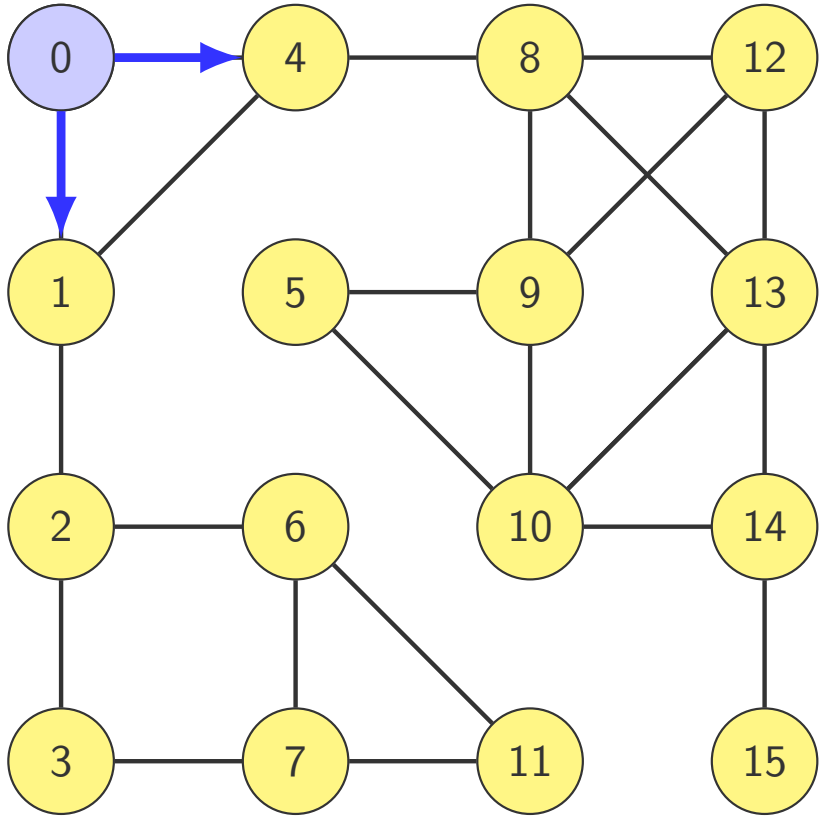
BFS usa fila



Fila | 1 |

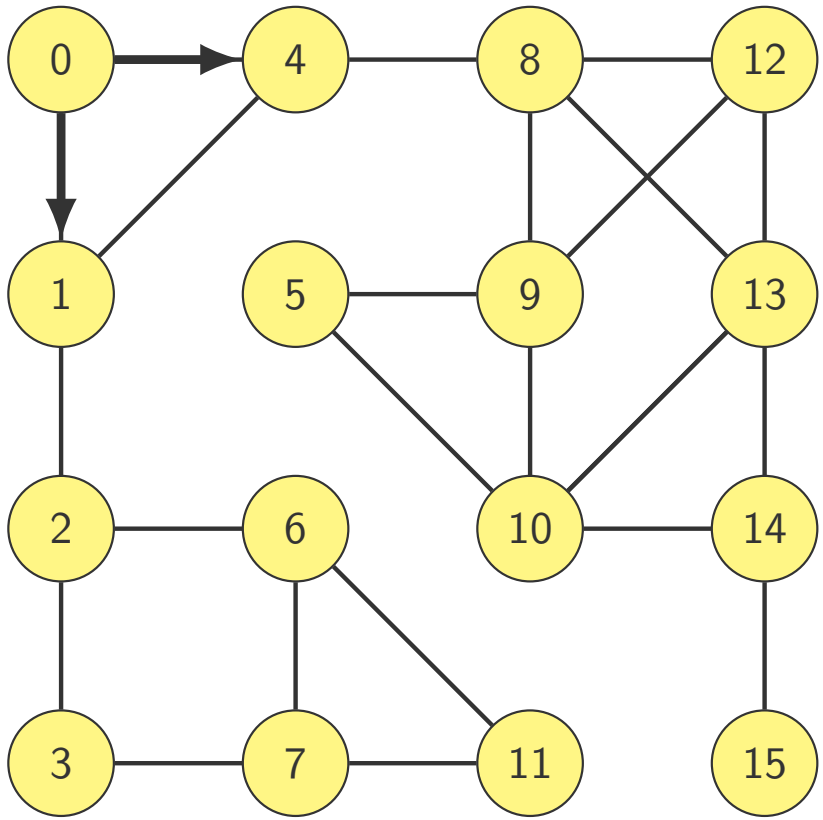
BFS usa fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



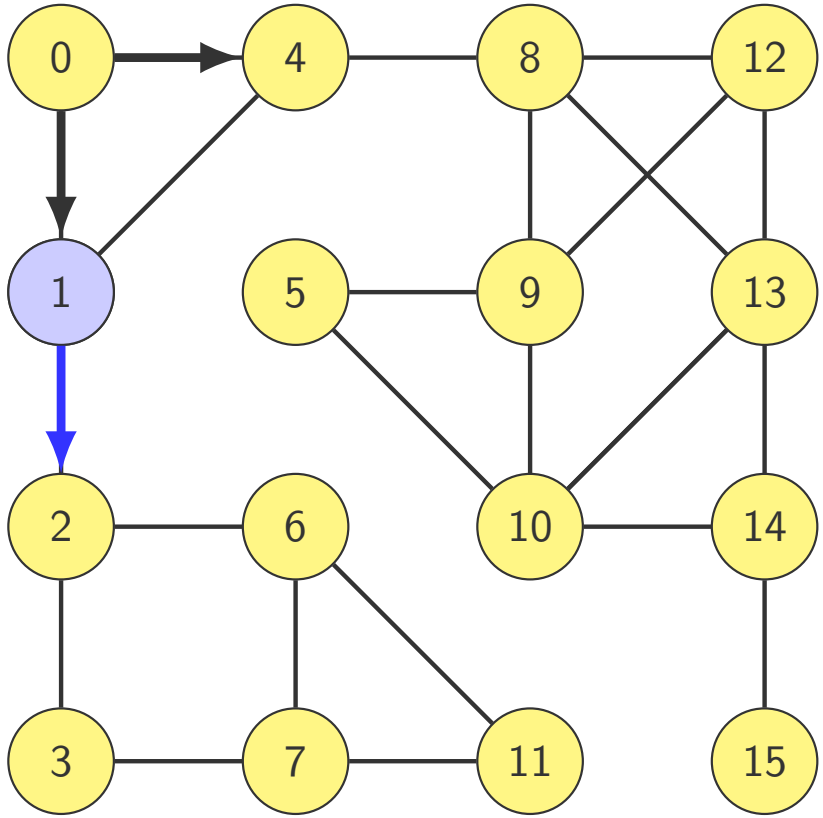
Fila | 1 | 4 |

BFS usa fila



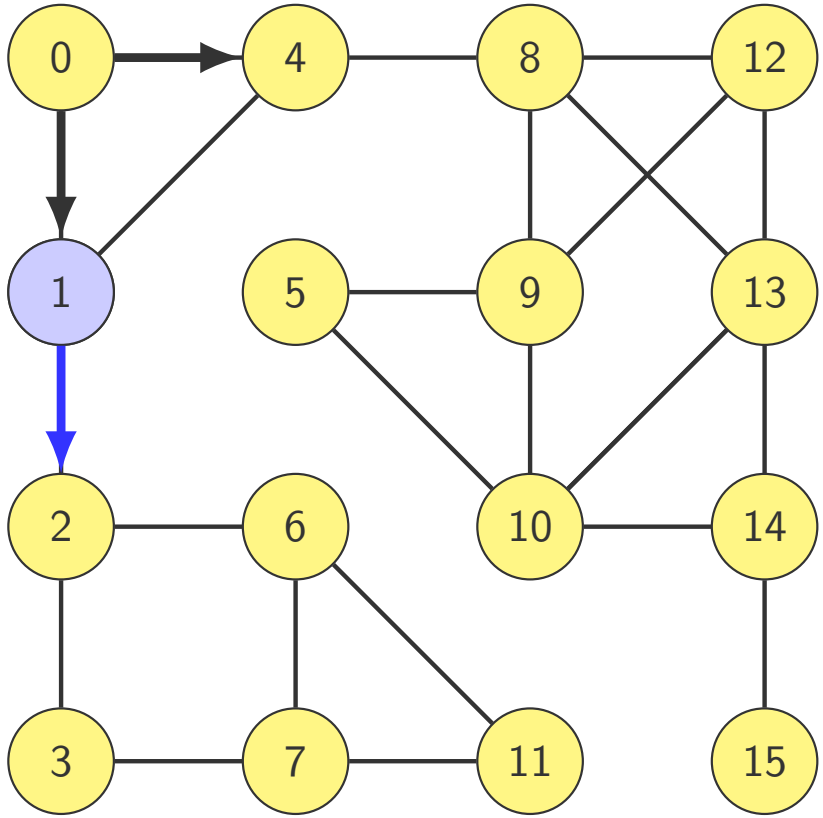
Fila | 1 | 4 |

BFS usa fila



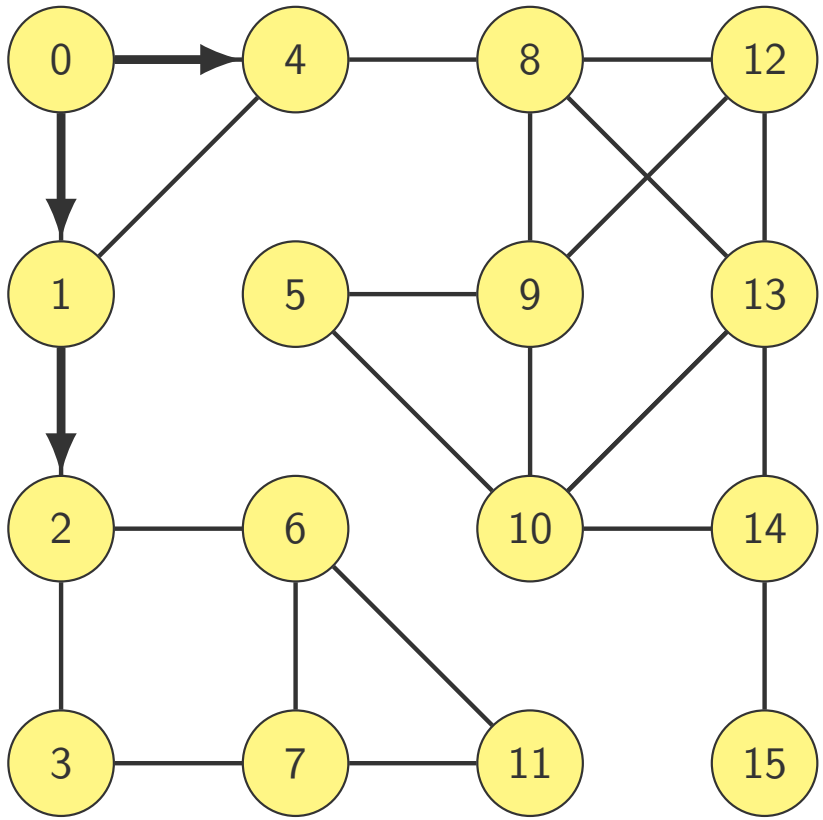
Fila | 4

BFS usa fila



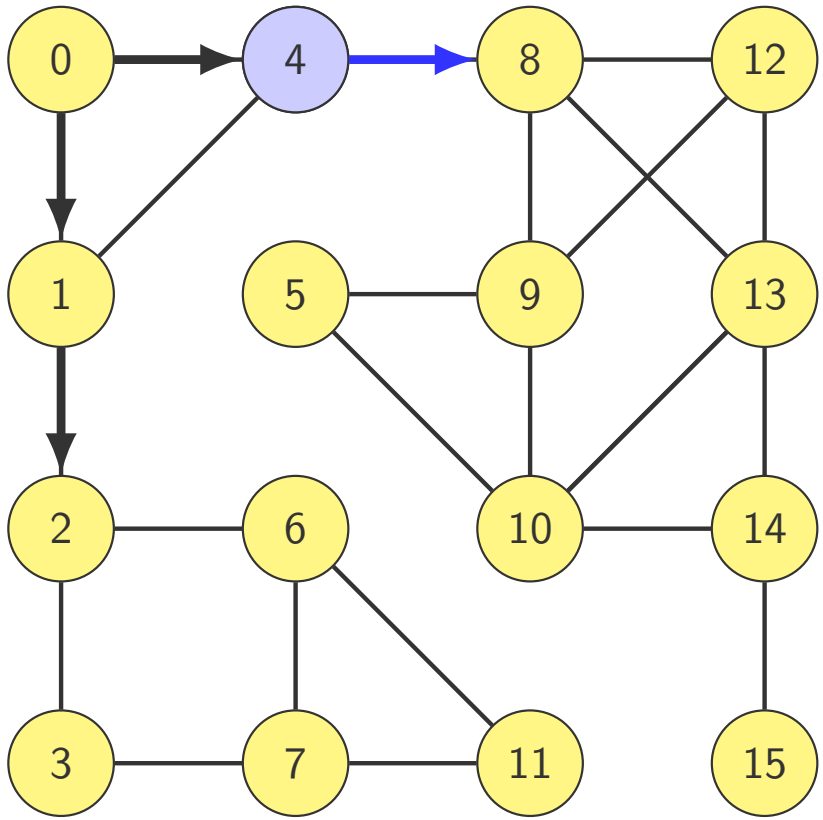
Fila | 4 | 2 |

BFS usa fila



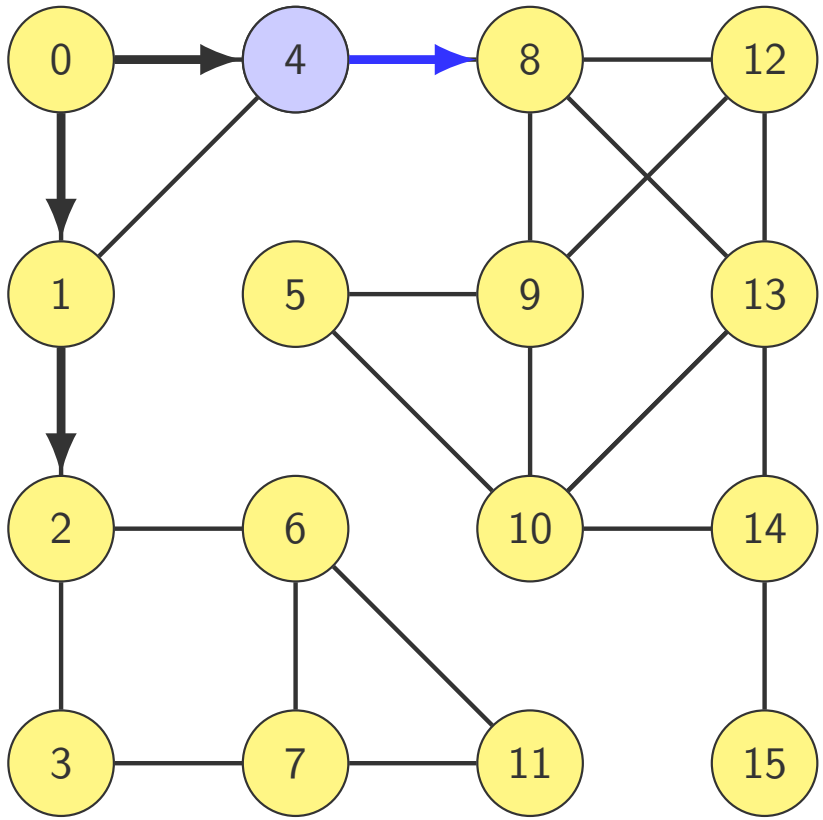
Fila | 4 | 2 |

BFS usa fila



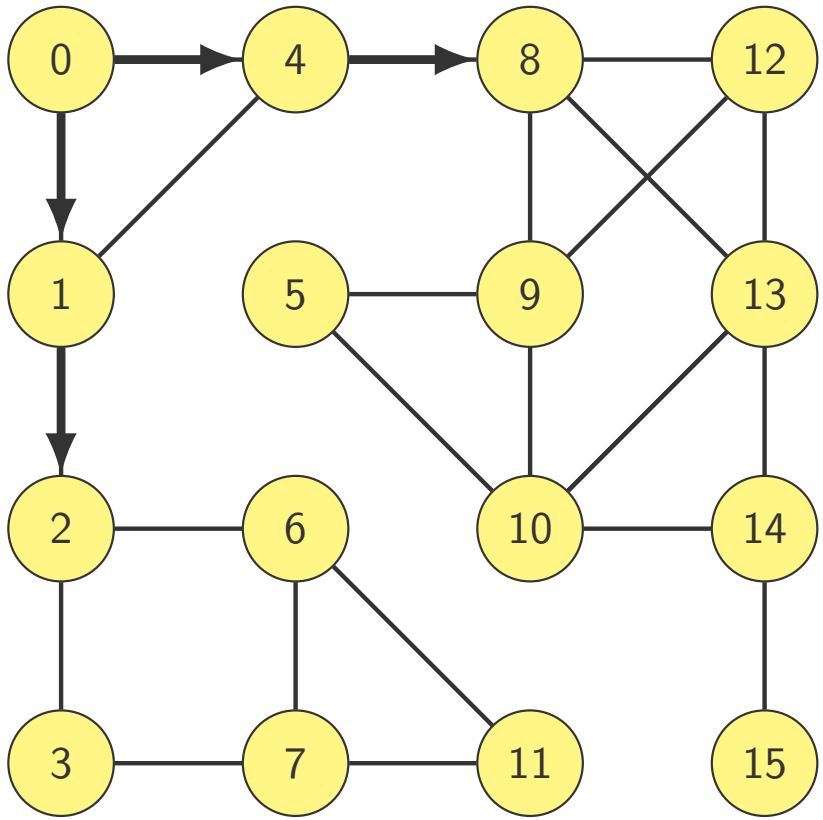
Fila | 2 |

BFS usa fila



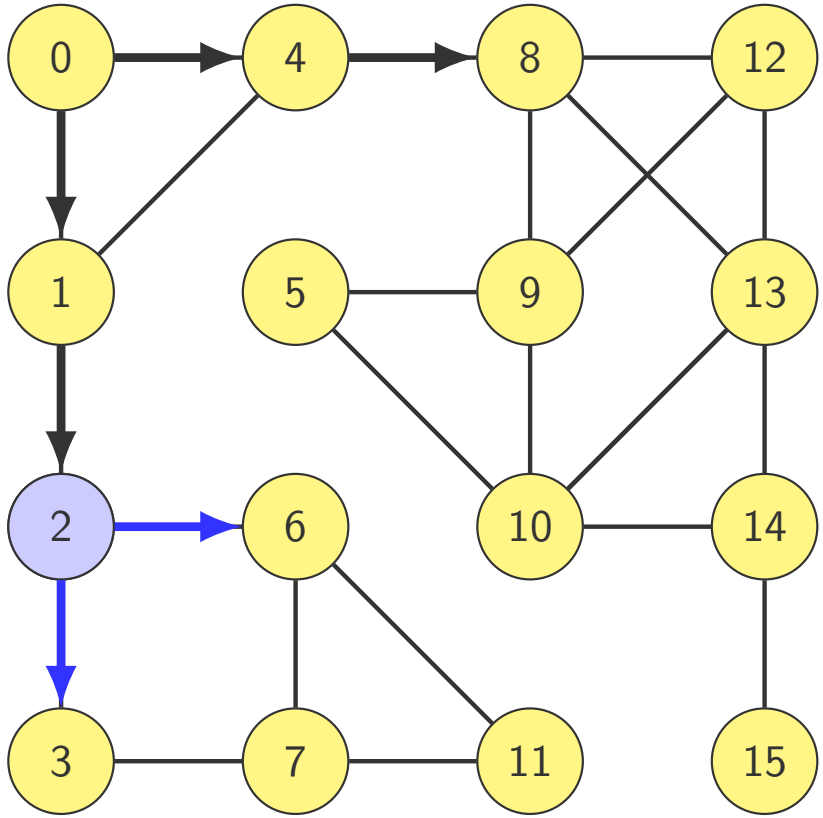
Fila | 2 | 8 |

BFS usa fila



Fila | 2 | 8 |

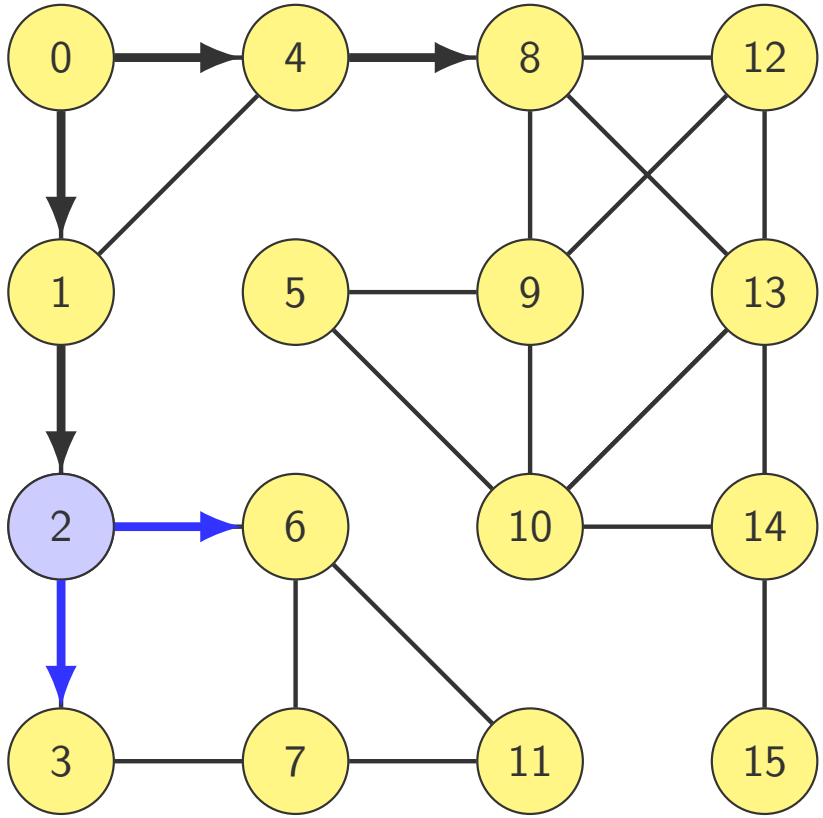
BFS usa fila



Fila | 8 |

BFS usa fila

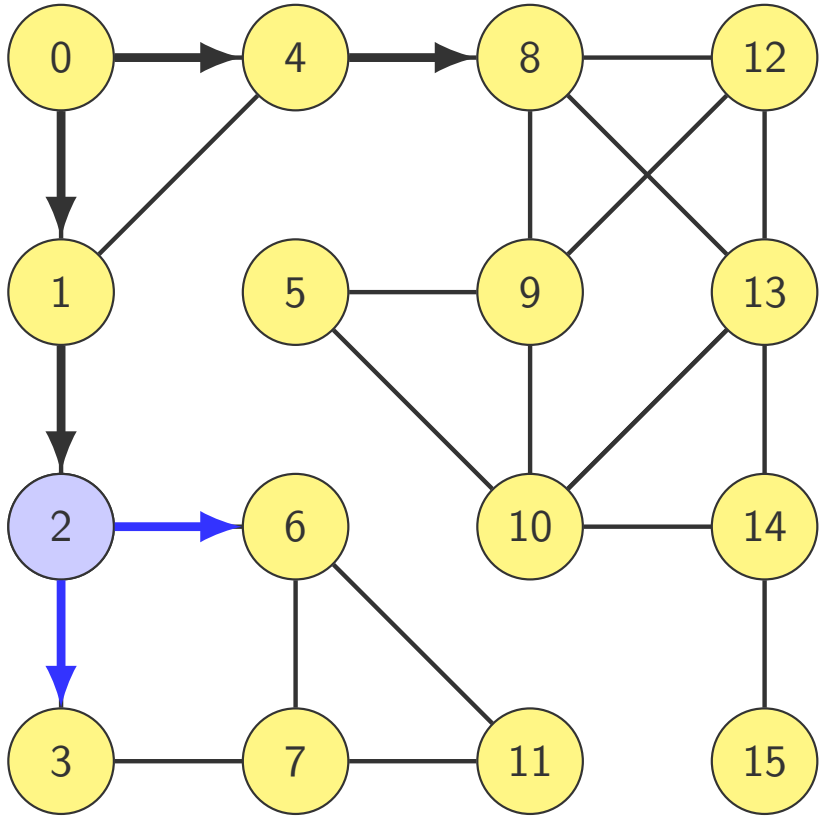
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



Fila | 8 | 3 |

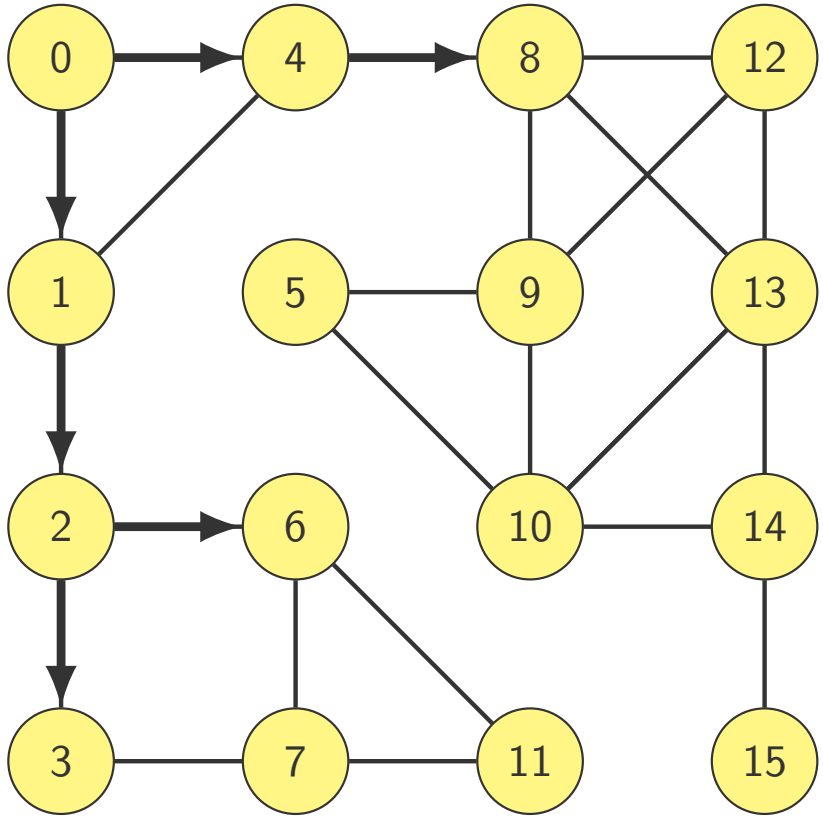
BFS usa fila

Universidade Federal do Ceará
Campus Quixadá



Fila | 8 | 3 | 6 |

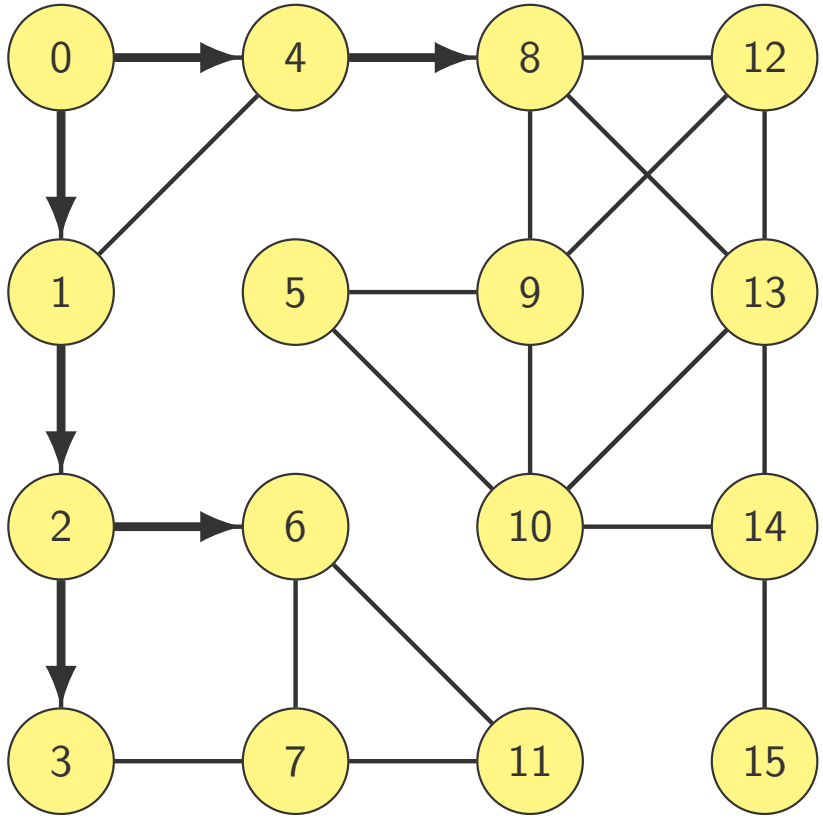
BFS usa fila



Fila

8	3	6

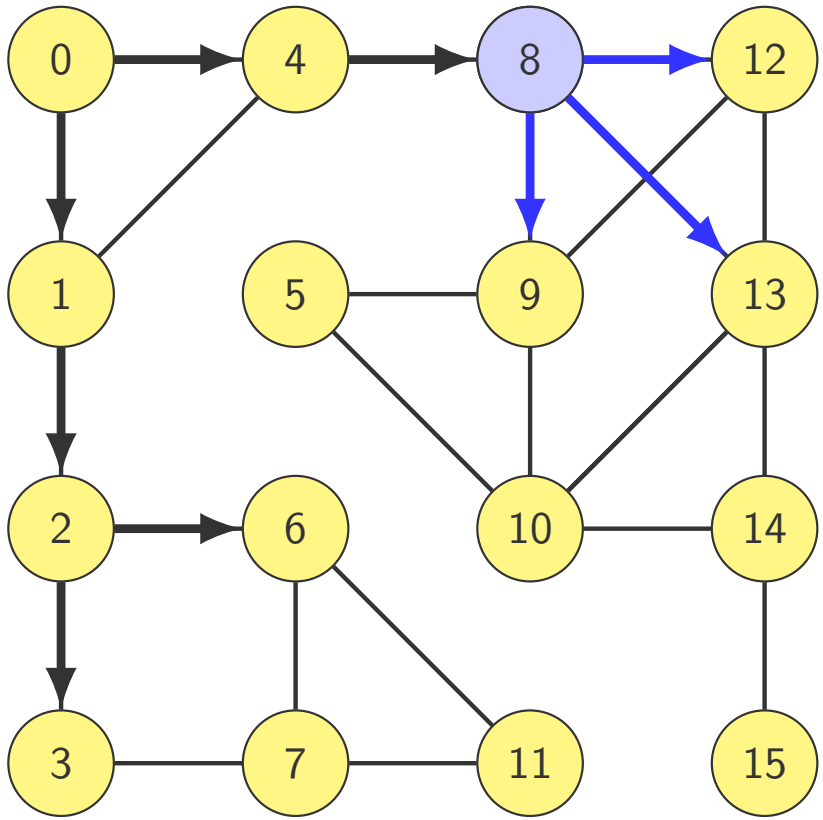
BFS usa fila



Fila

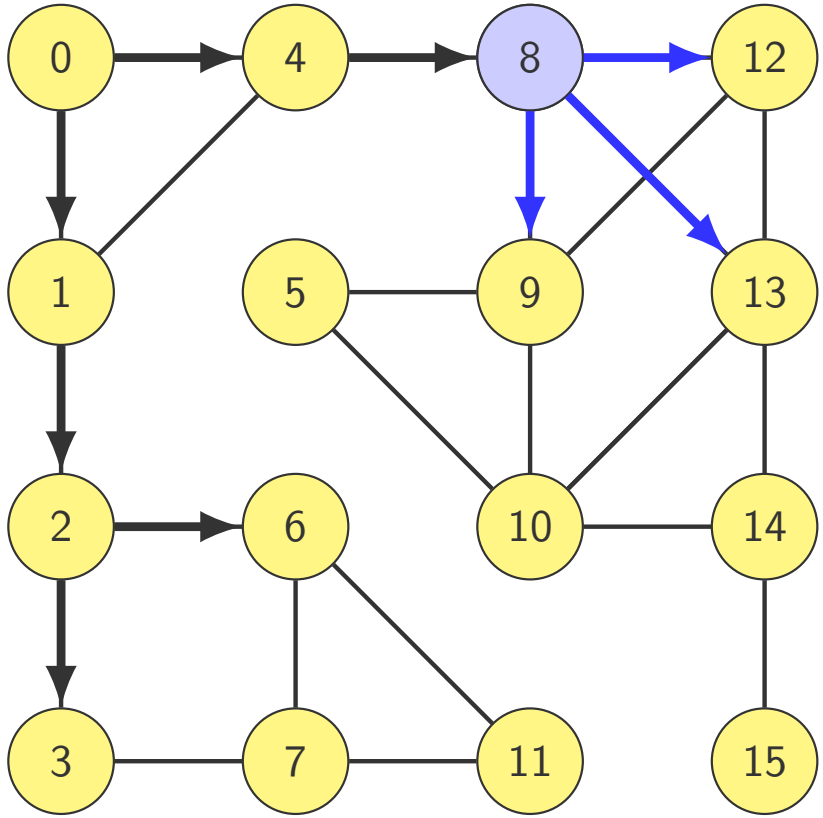
8	3	6

BFS usa fila



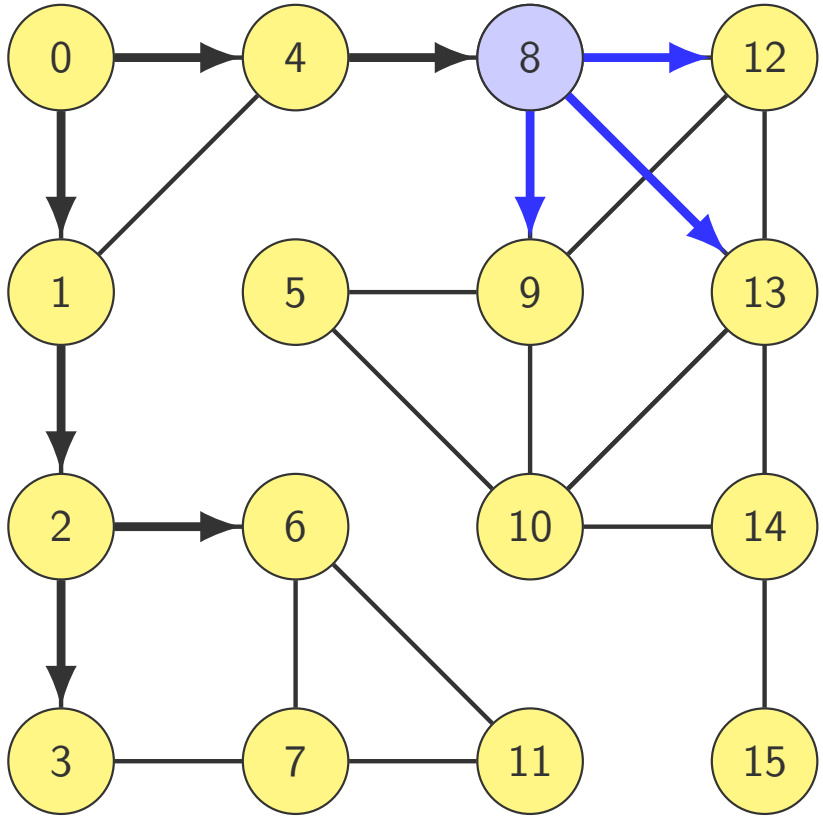
Fila | 3 | 6 |

BFS usa fila



Fila | 3 | 6 | 9 |

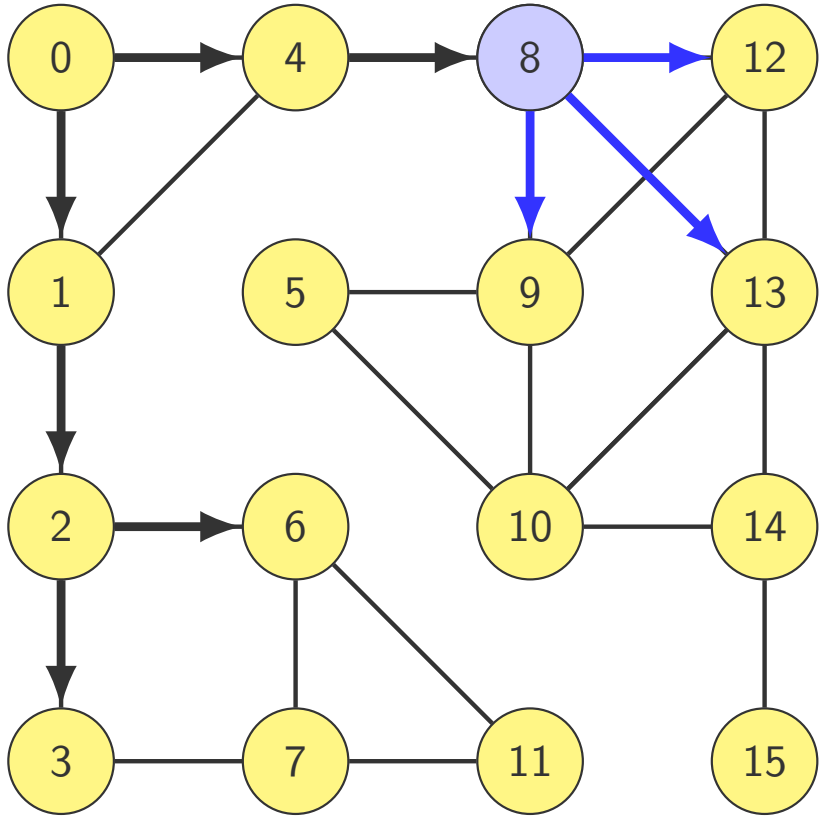
BFS usa fila



Fila

3	6	9	12

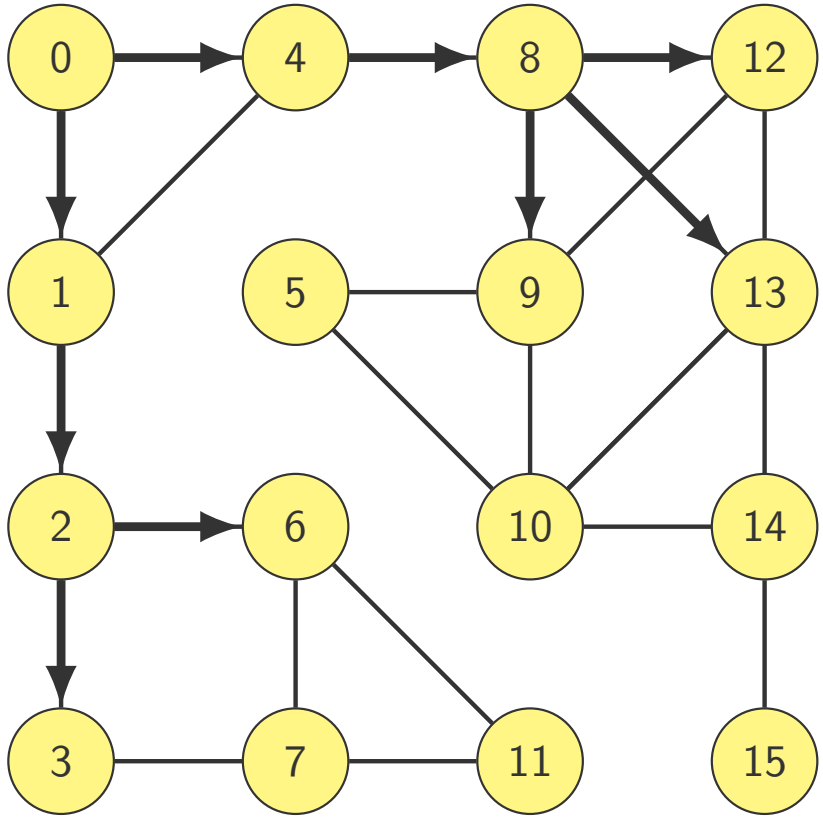
BFS usa fila



Fila

3	6	9	12	13

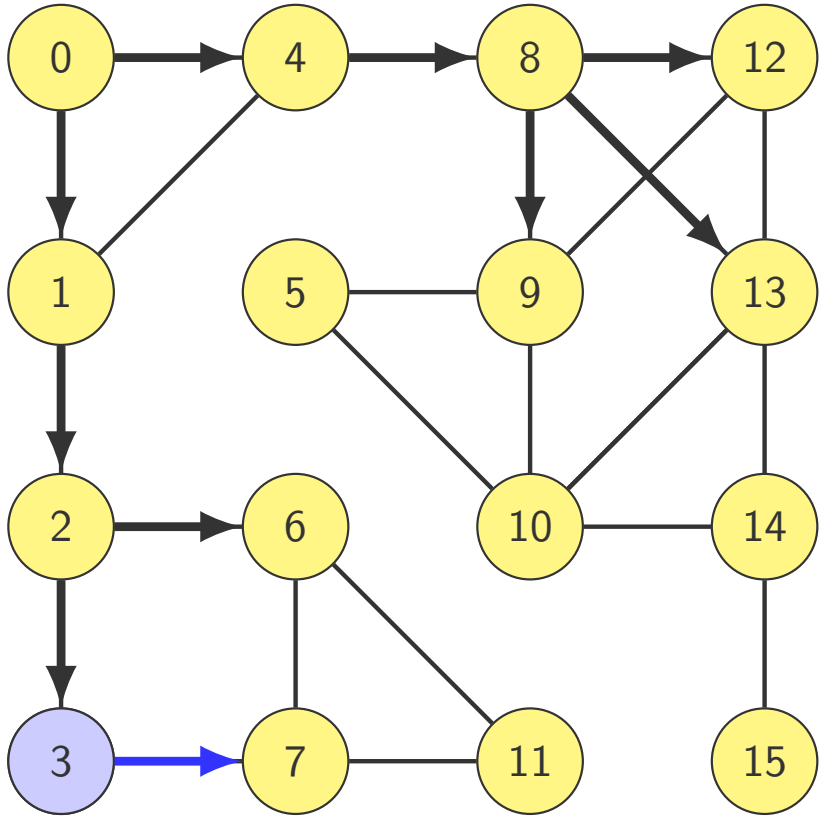
BFS usa fila



Fila

3	6	9	12	13

BFS usa fila

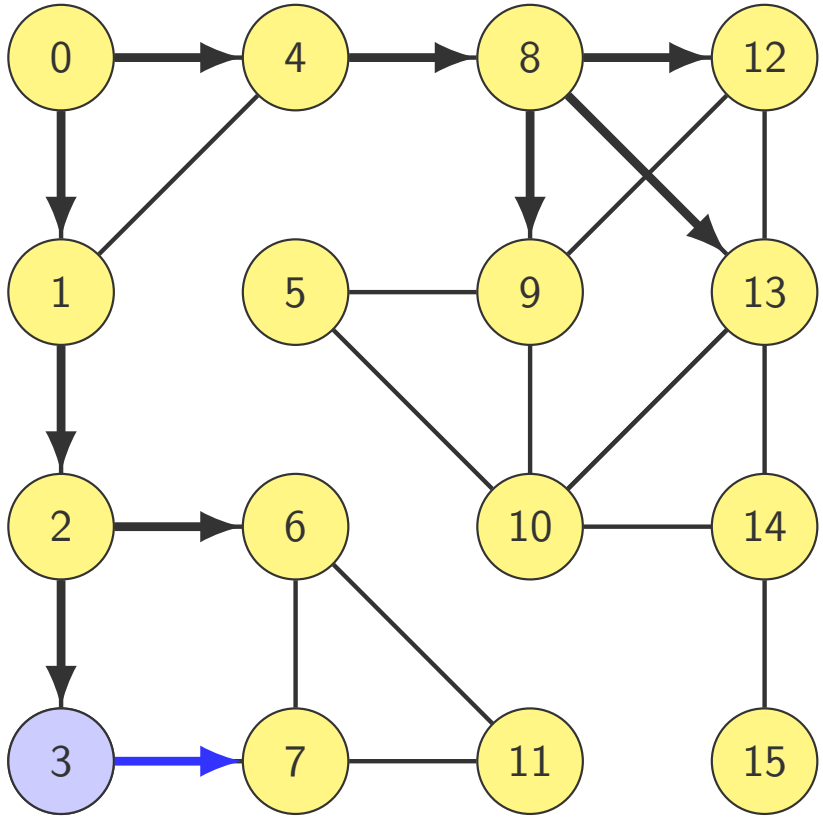


Fila

6	9	12	13

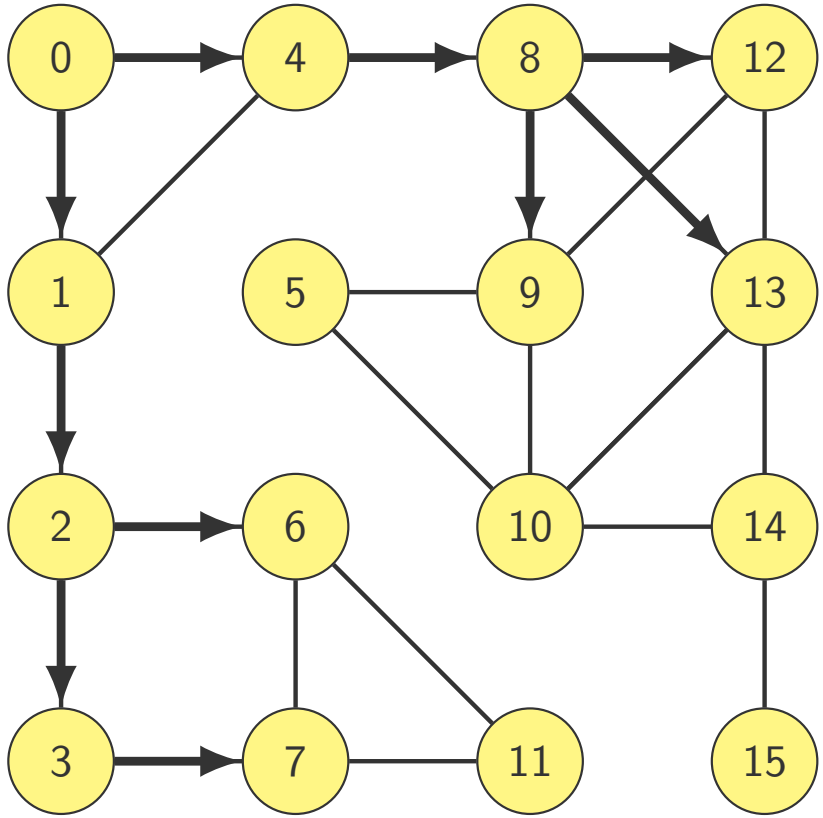
BFS usa fila

Universidade Federal do Ceará
Campus Quixadá



Fila	6	9	12	13	7

BFS usa fila

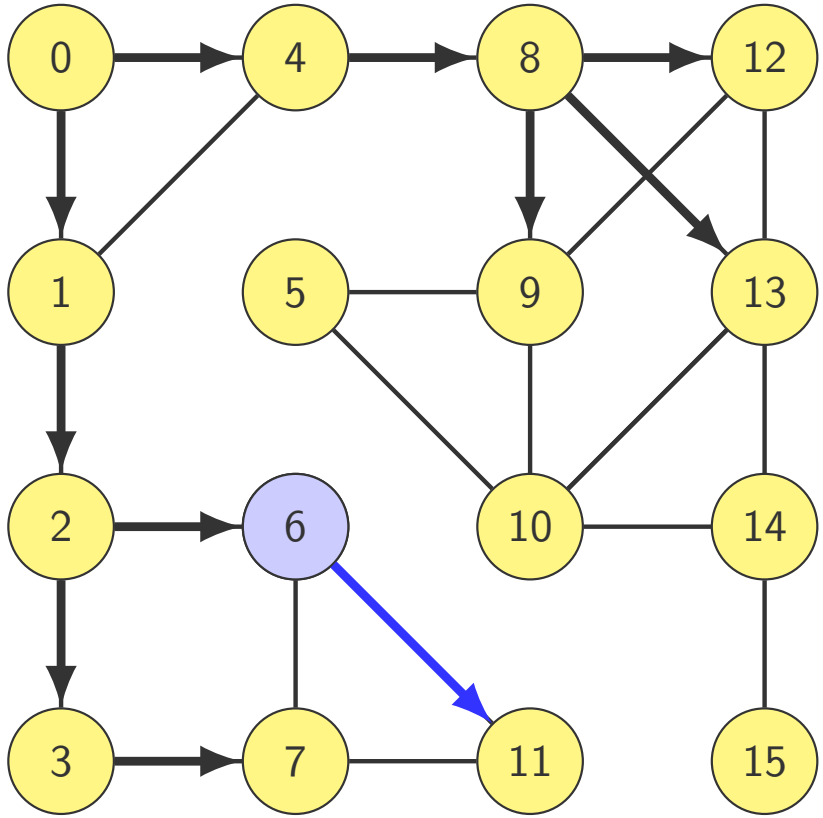


Fila

6	9	12	13	7

BFS usa fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

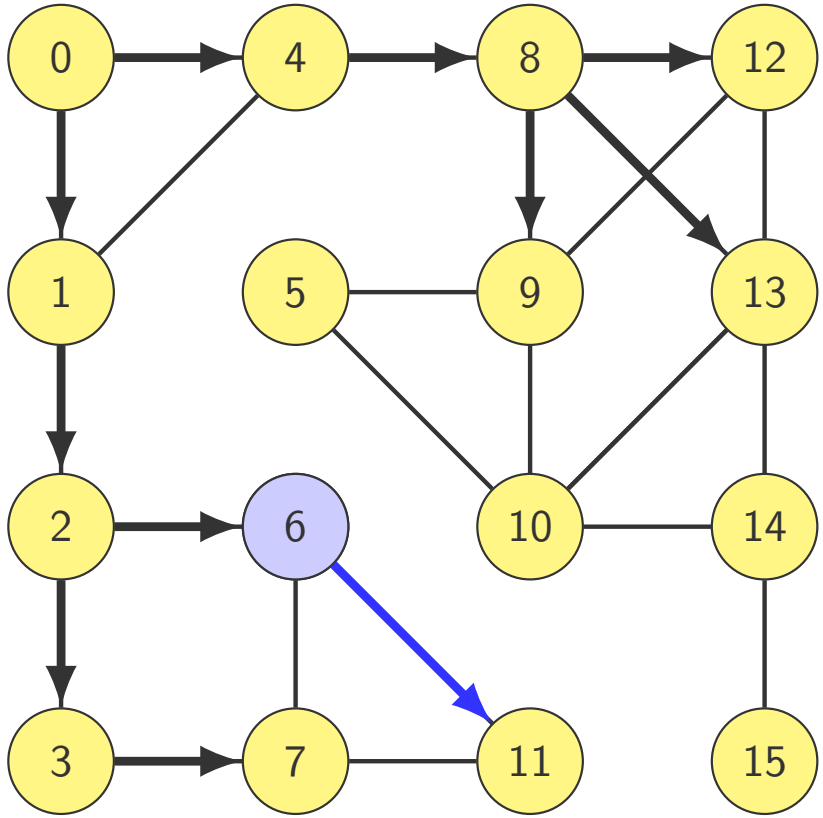


Fila

9	12	13	7

BFS usa fila

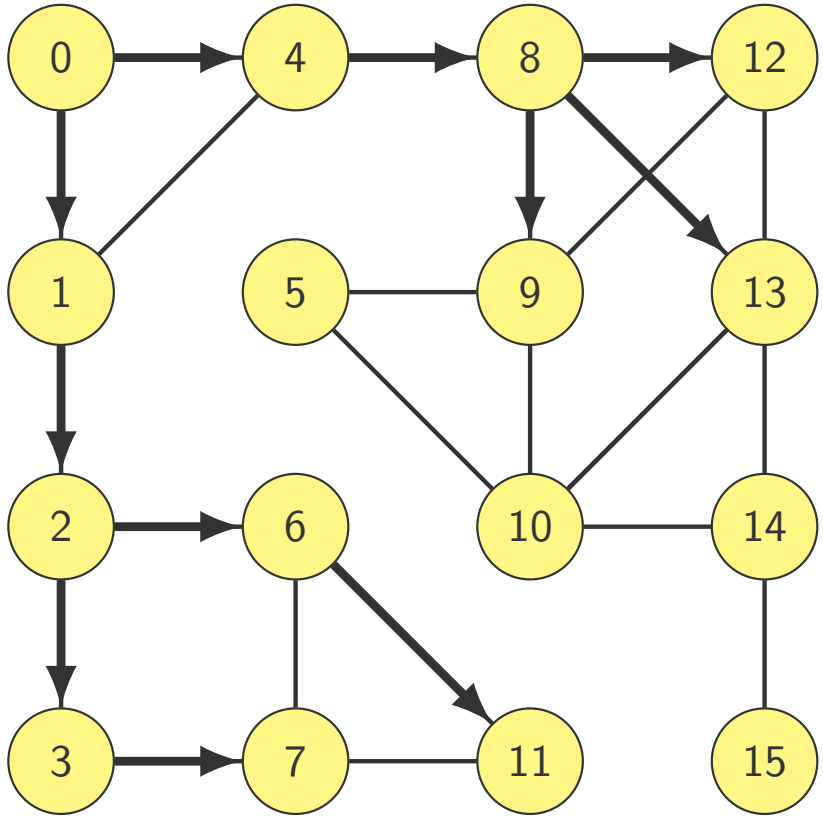
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



Fila

9	12	13	7	11

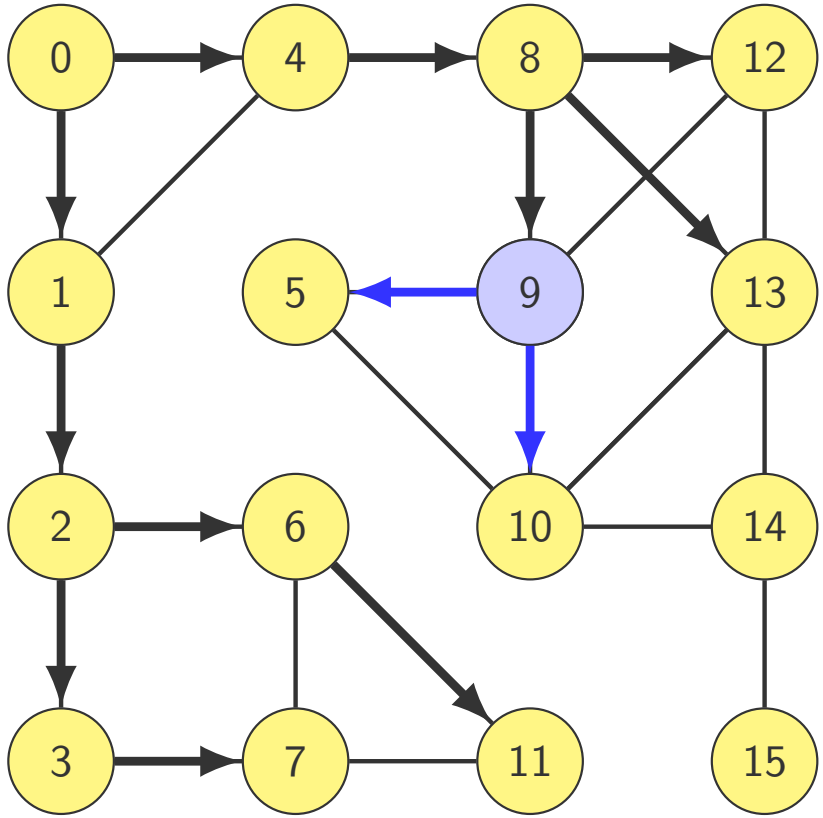
BFS usa fila



Fila	9	12	13	7	11

BFS usa fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

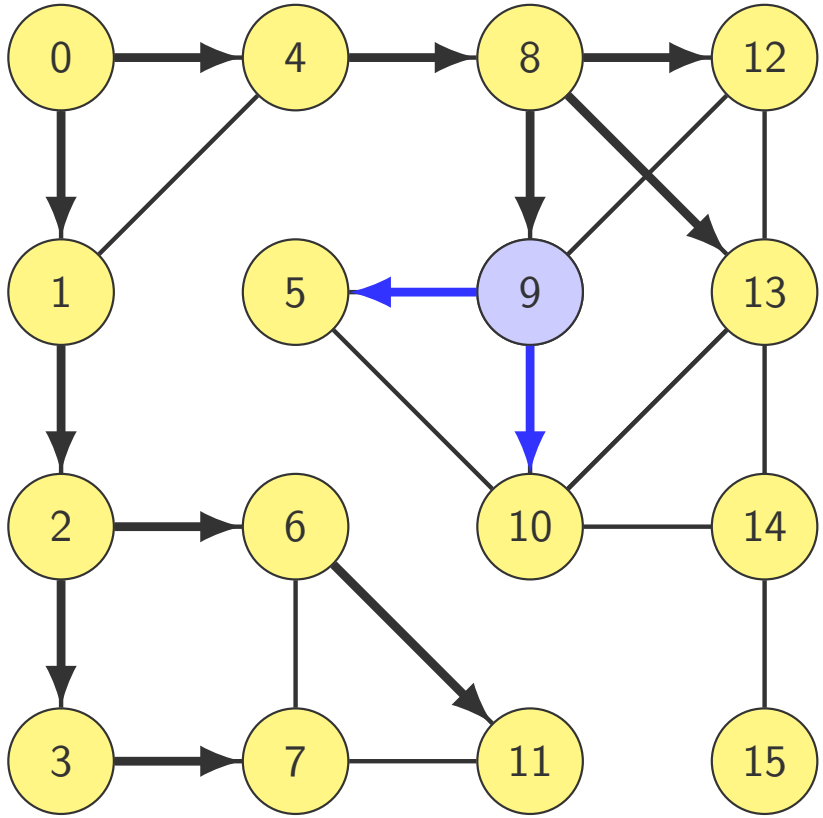


Fila

12	13	7	11

BFS usa fila

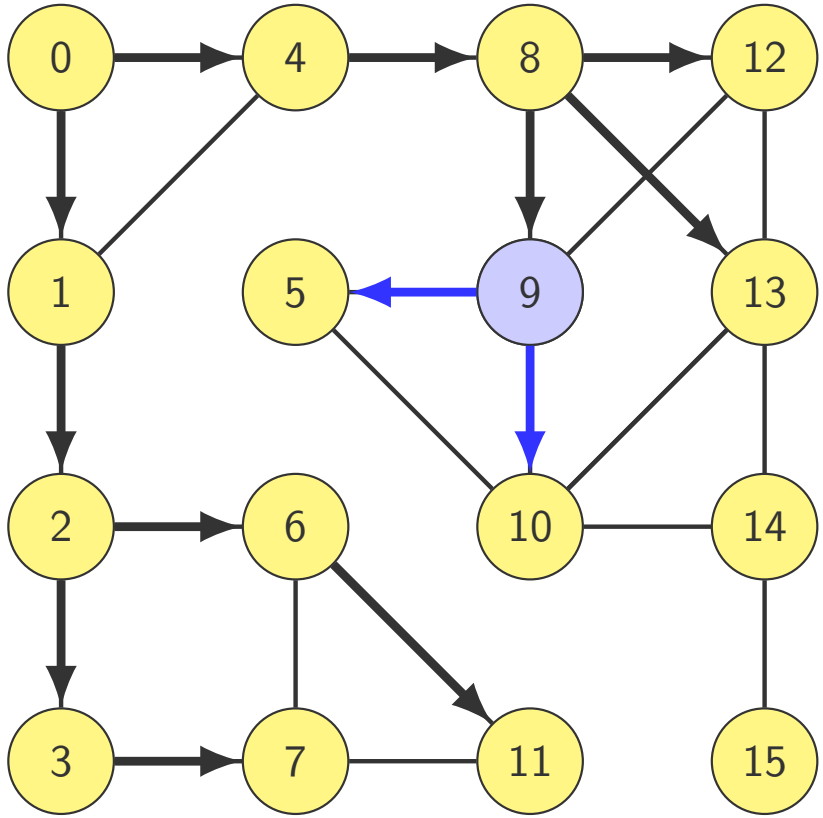
Universidade Federal do Ceará
Campus Quixadá



Fila	12	13	7	11	5

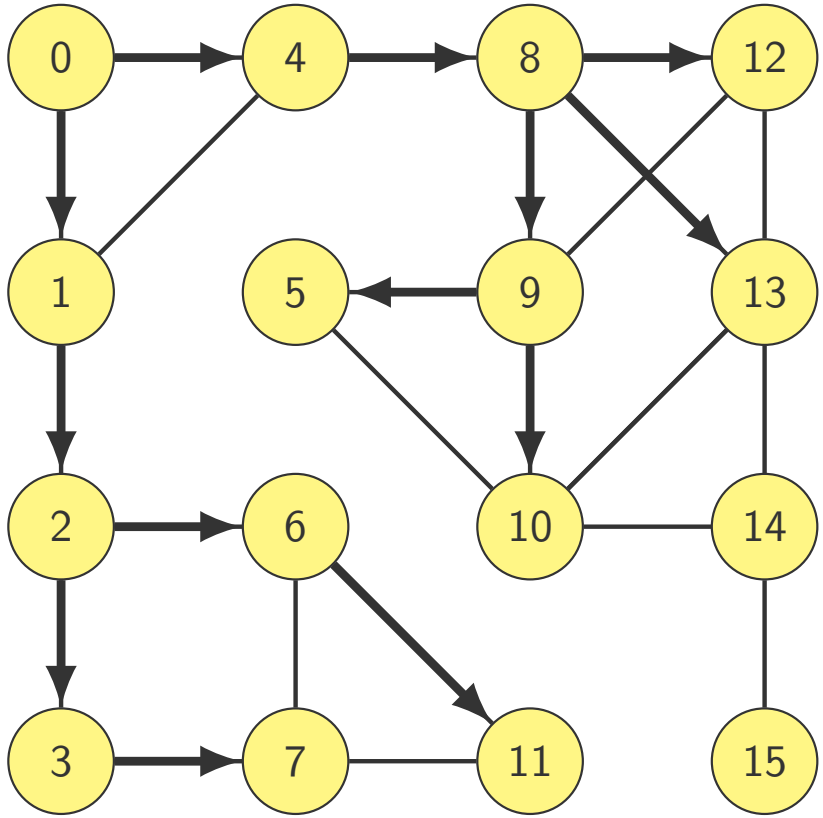
BFS usa fila

Universidade Federal do Ceará
Campus Quixadá



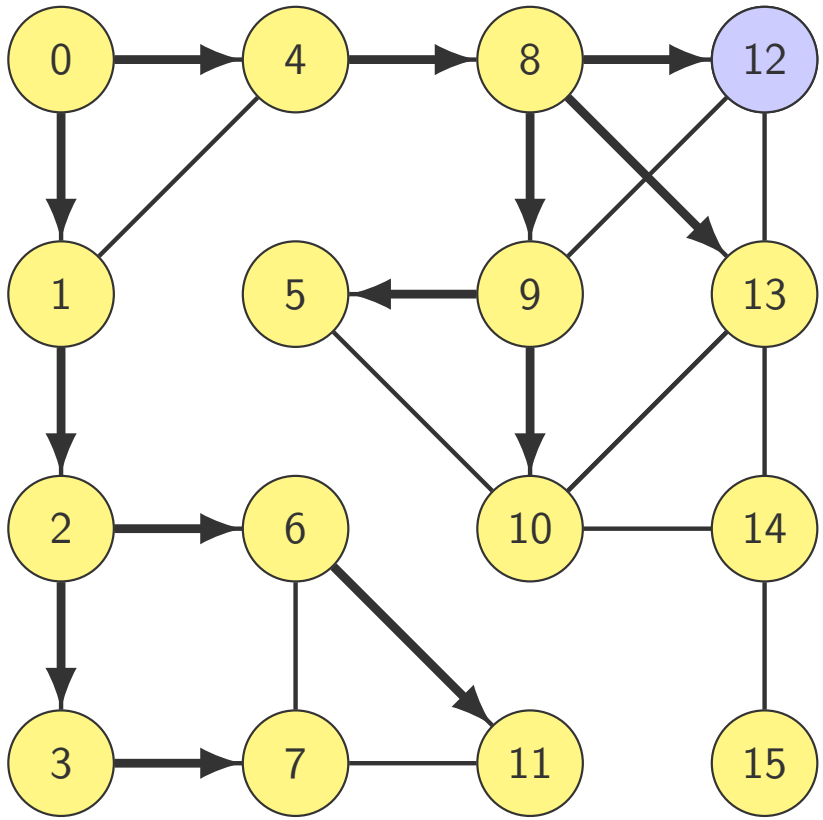
Fila	12	13	7	11	5	10

BFS usa fila



Fila	12	13	7	11	5	10

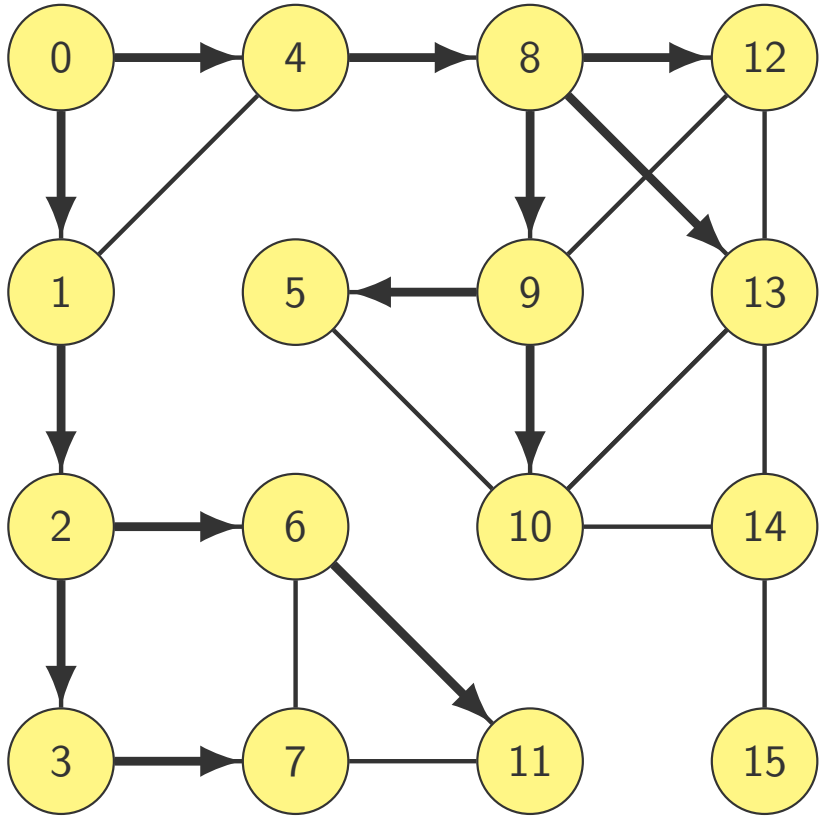
BFS usa fila



Fila

13	7	11	5	10

BFS usa fila

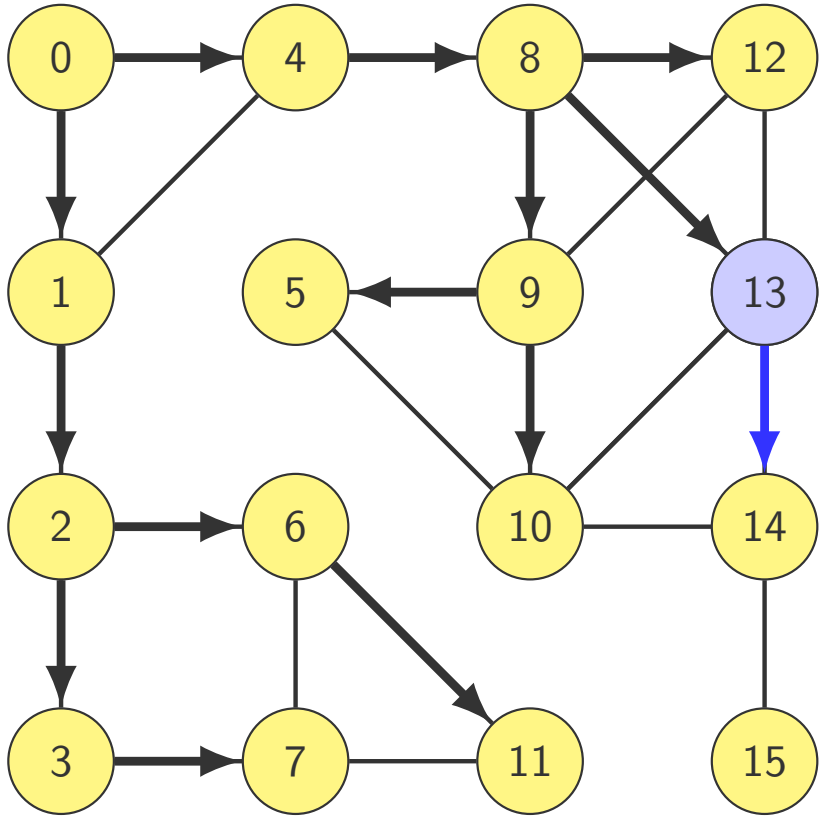


Fila

13	7	11	5	10

BFS usa fila

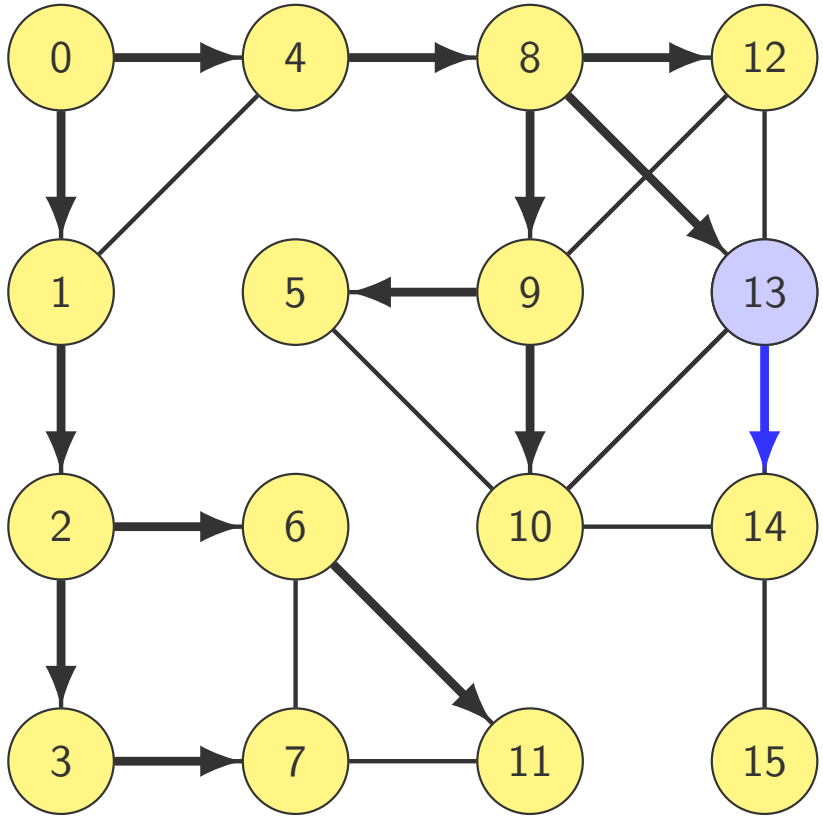
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



Fila

7	11	5	10

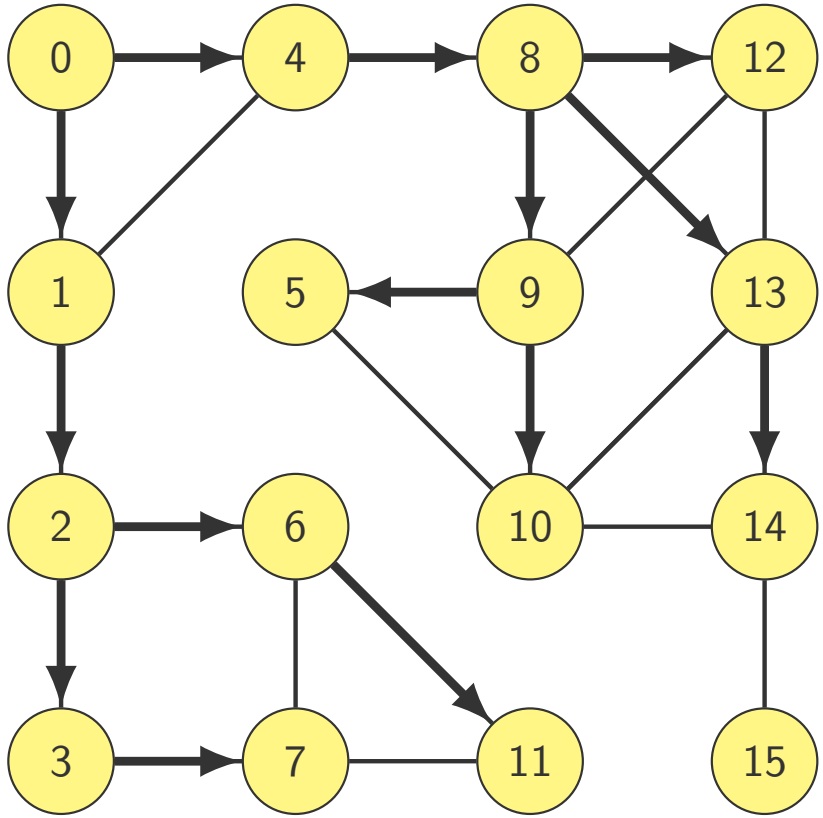
BFS usa fila



Fila

7	11	5	10	14

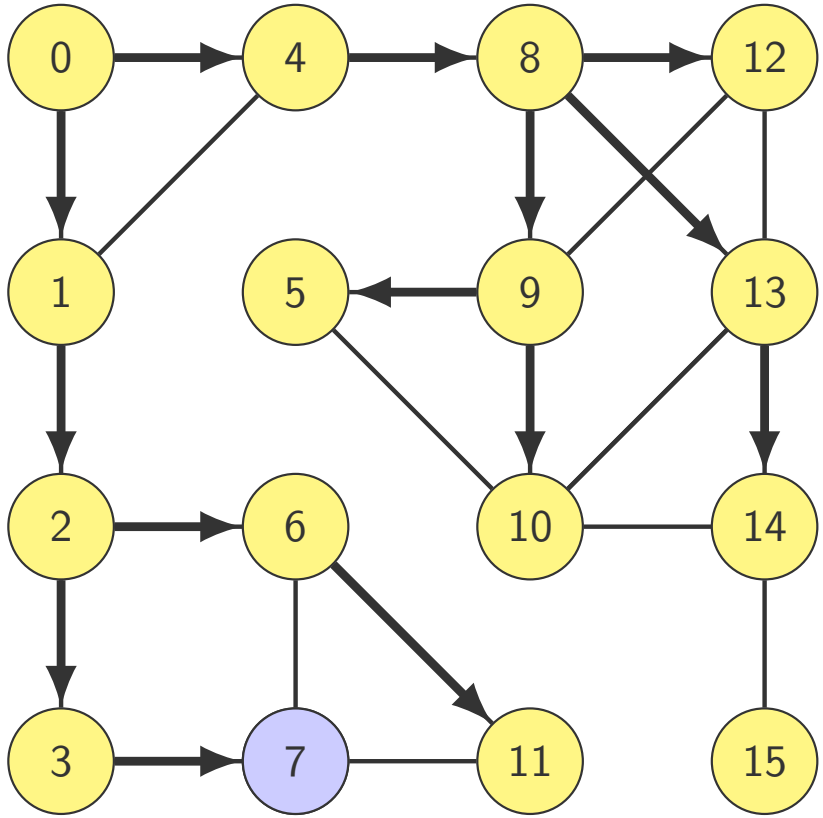
BFS usa fila



Fila

7	11	5	10	14

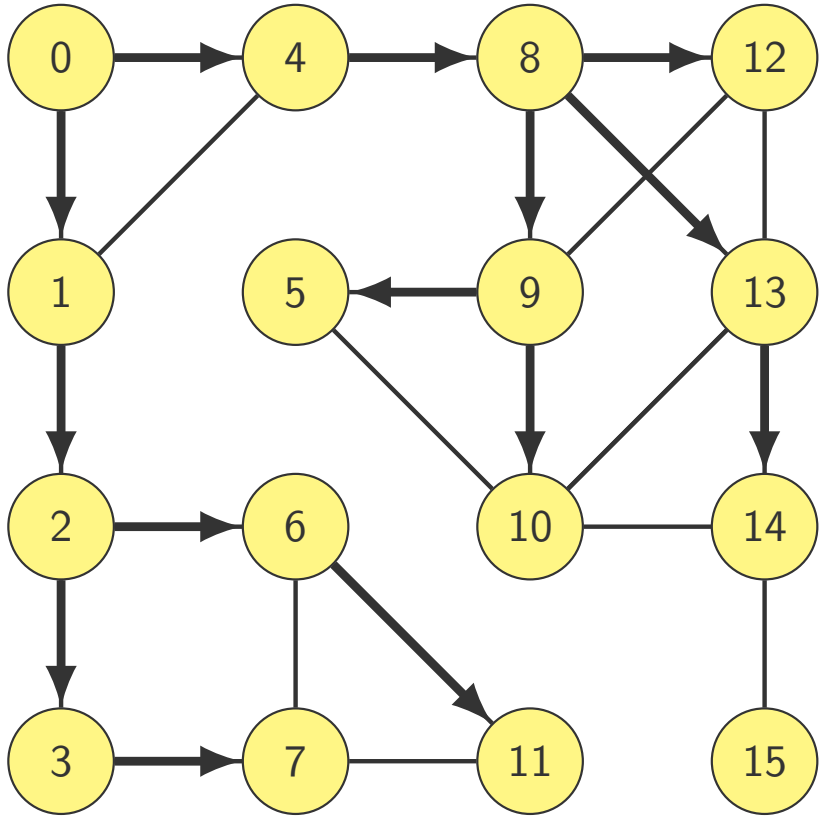
BFS usa fila



Fila

11	5	10	14

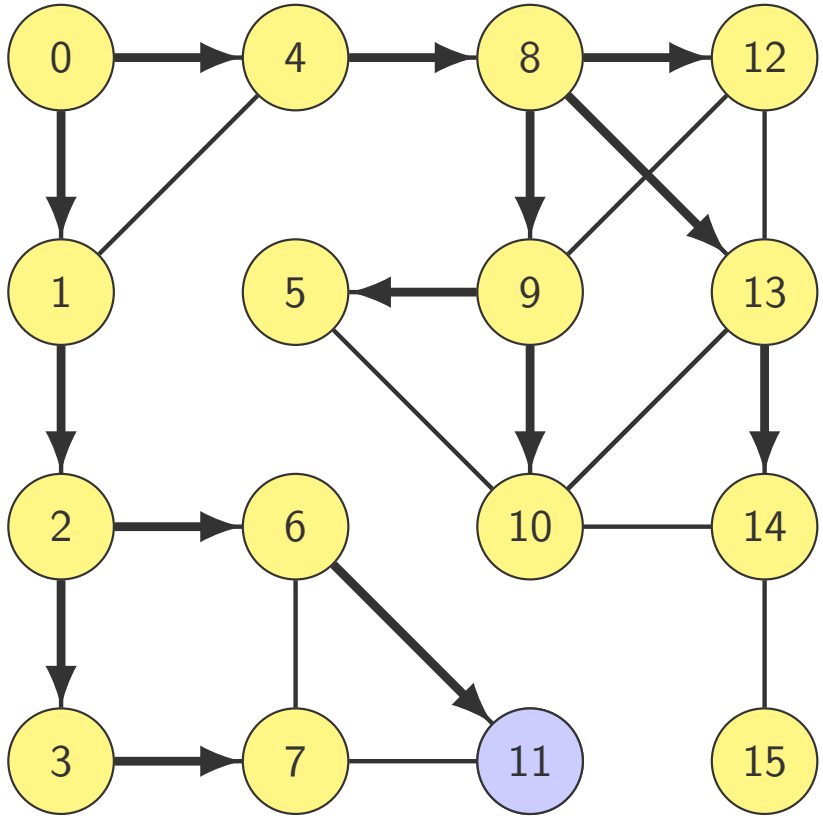
BFS usa fila



Fila

11	5	10	14

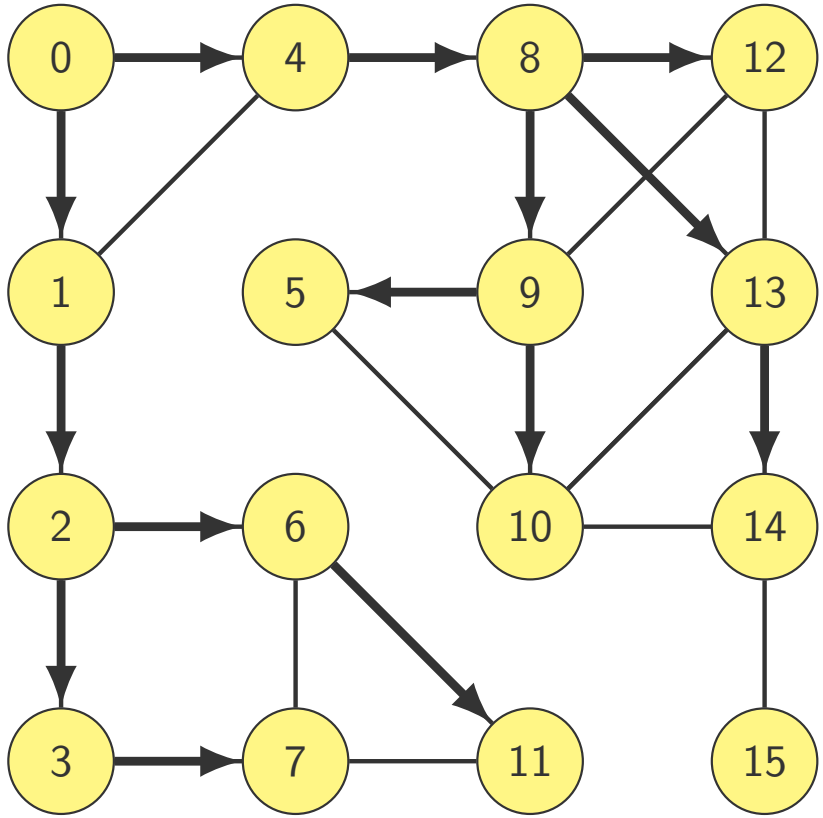
BFS usa fila



Fila

5	10	14

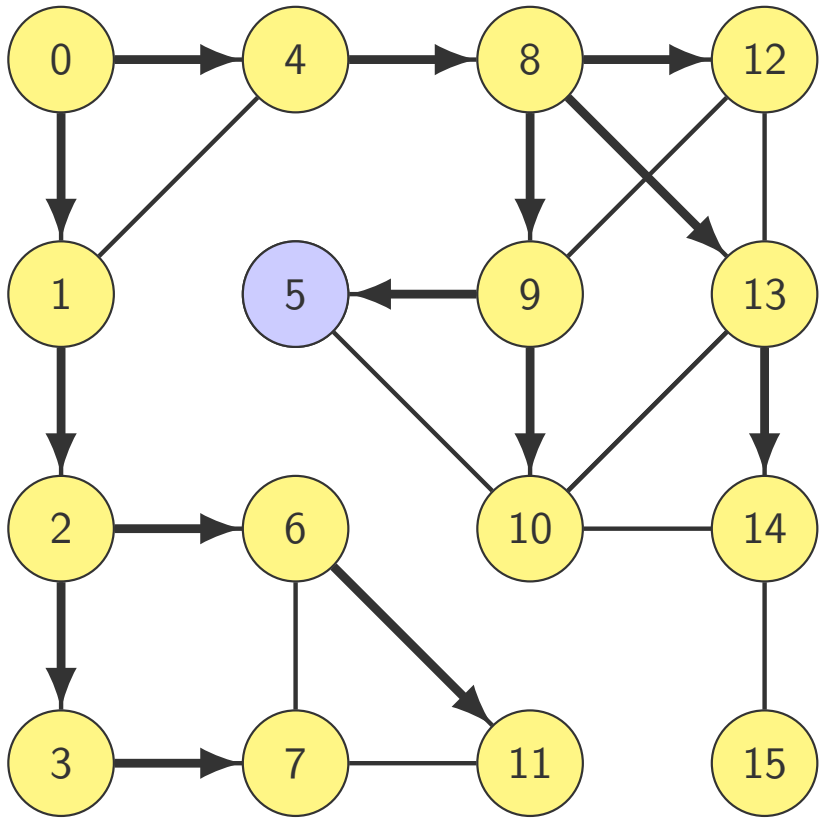
BFS usa fila



Fila

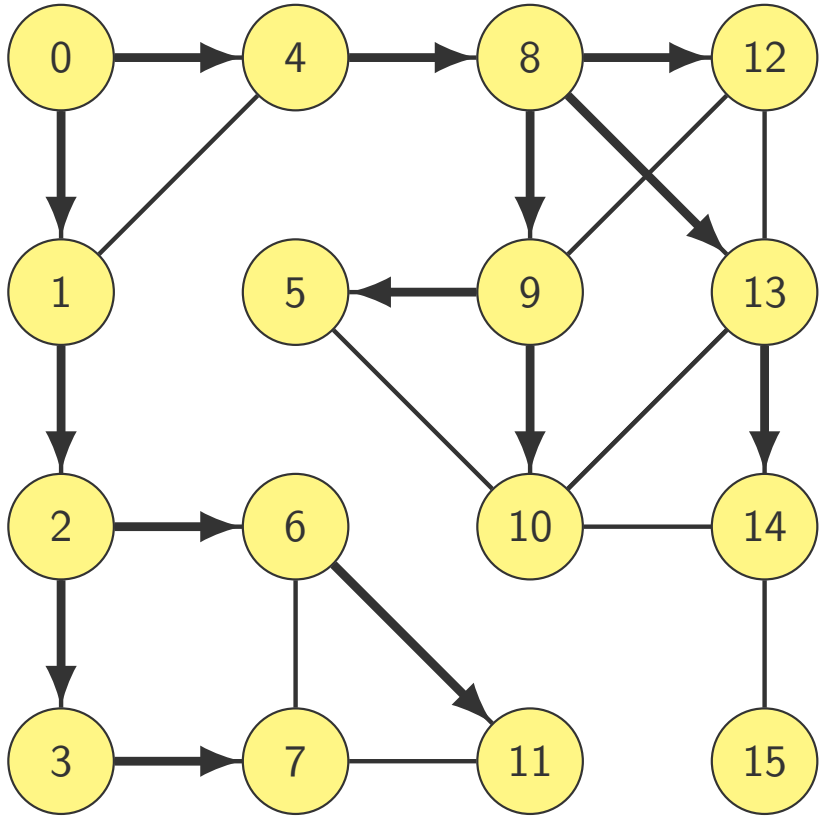
5	10	14

BFS usa fila



Fila | 10 | 14 |

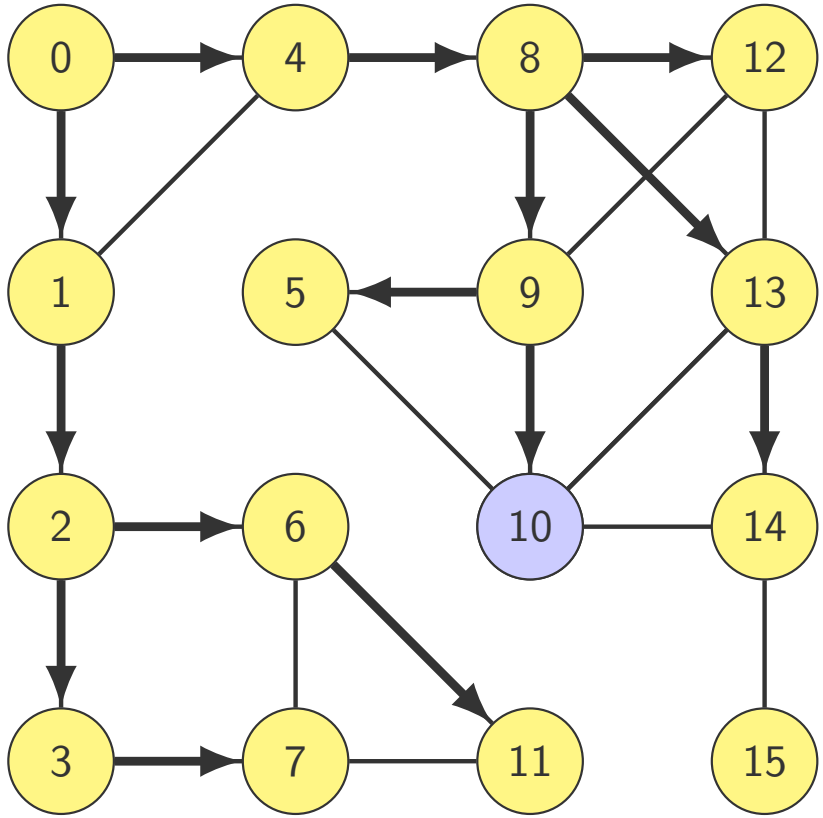
BFS usa fila



Fila | 10 | 14 |

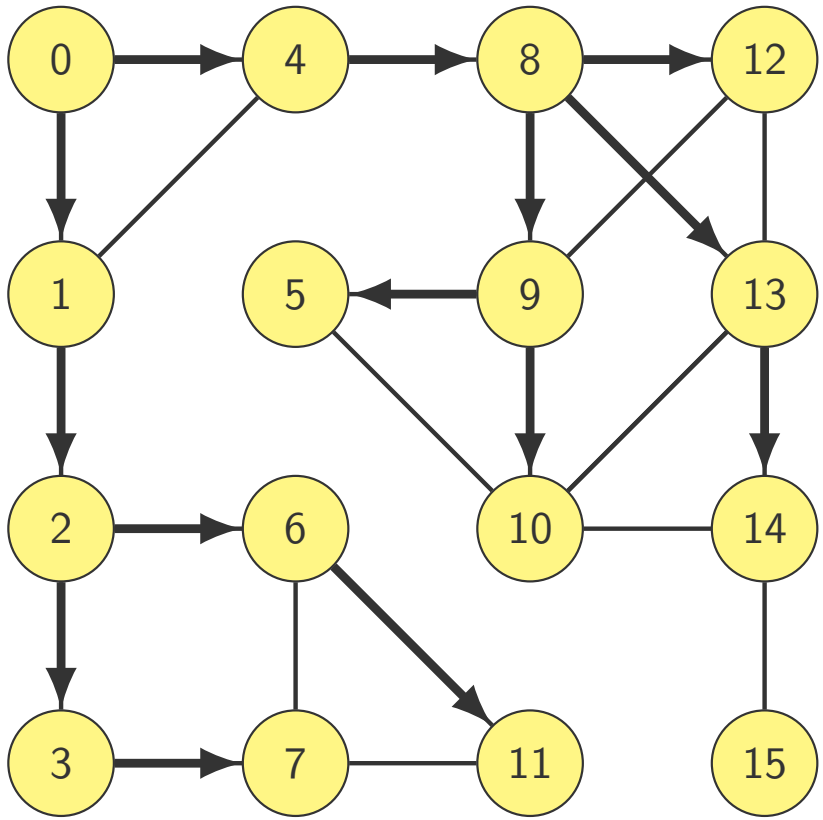
BFS usa fila

Universidade Federal do Ceará
Campus Quixadá



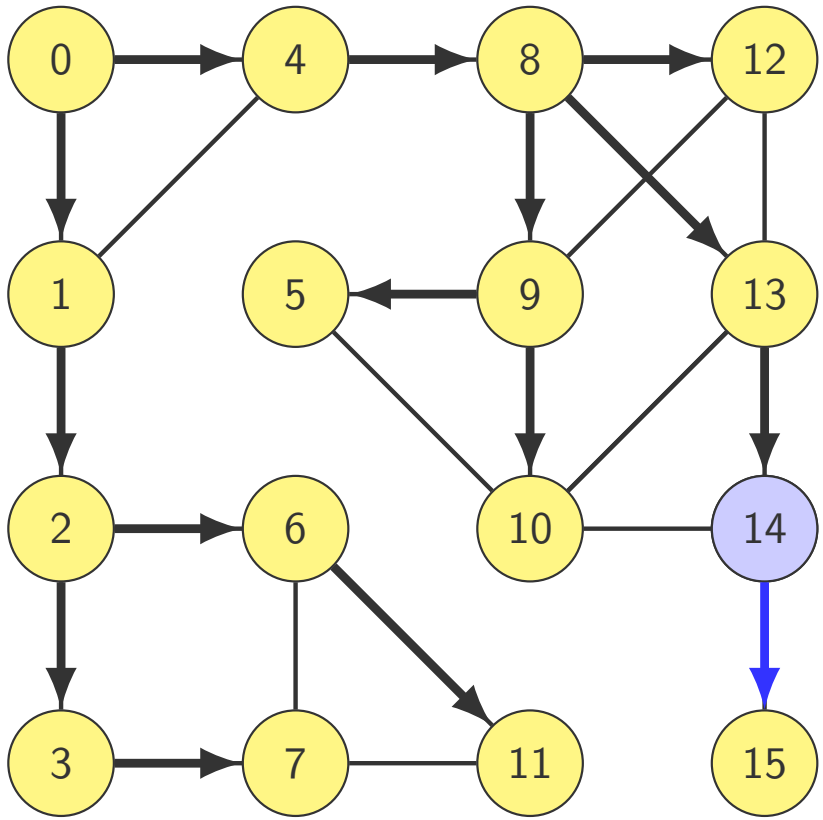
Fila | 14

BFS usa fila



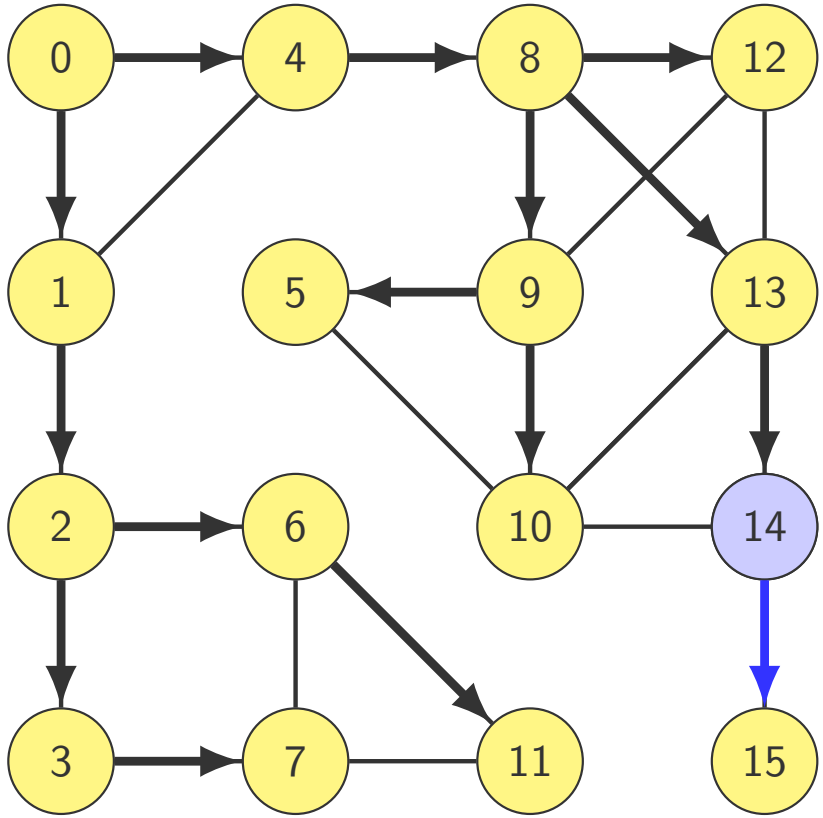
Fila | 14

BFS usa fila



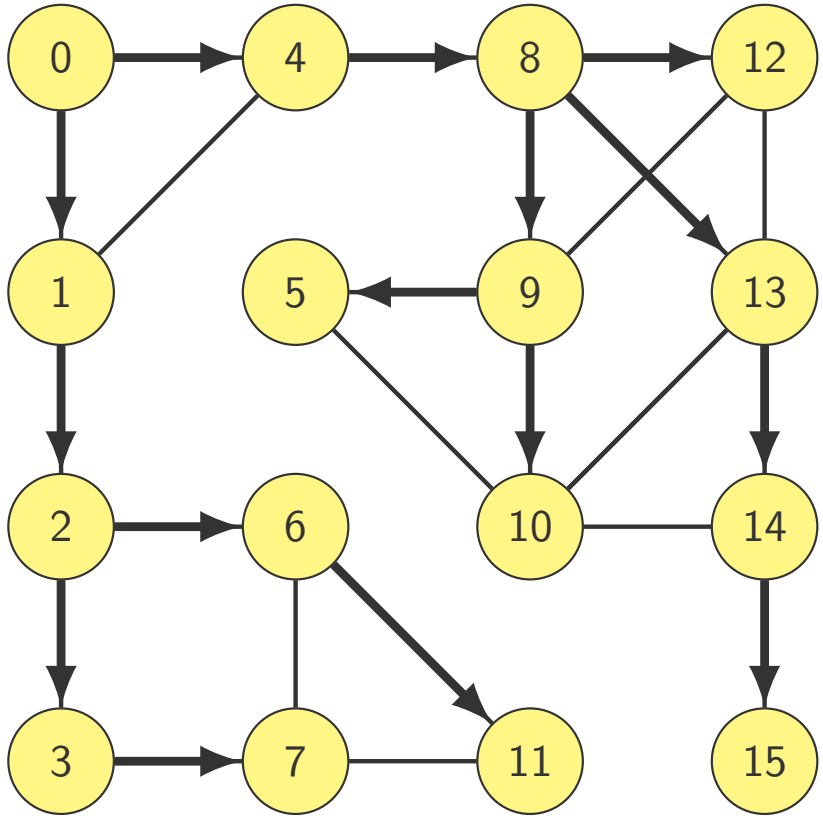
Fila

BFS usa fila



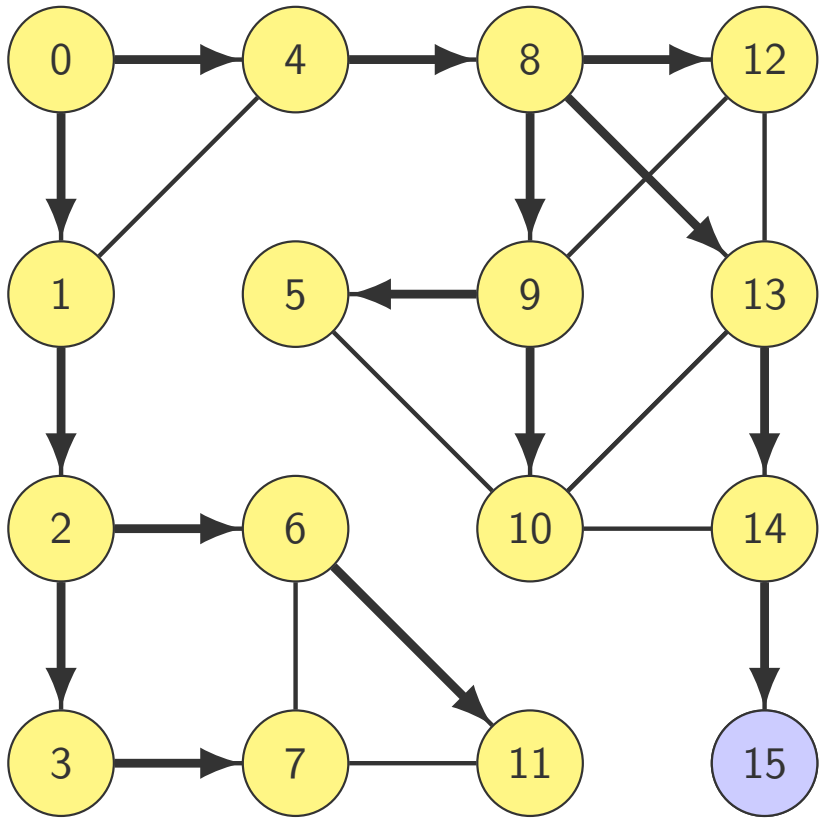
Fila | 15 |

BFS usa fila



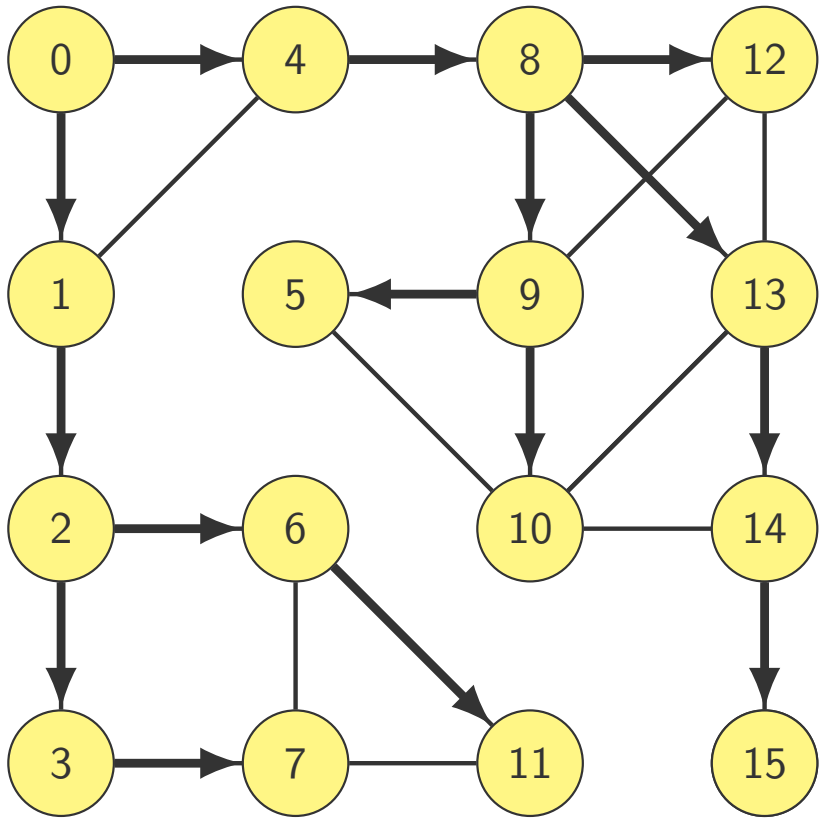
Fila | 15 |

BFS usa fila



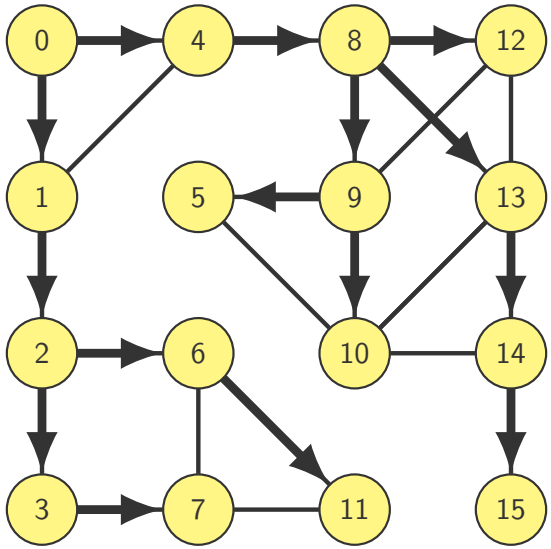
Fila

BFS usa fila



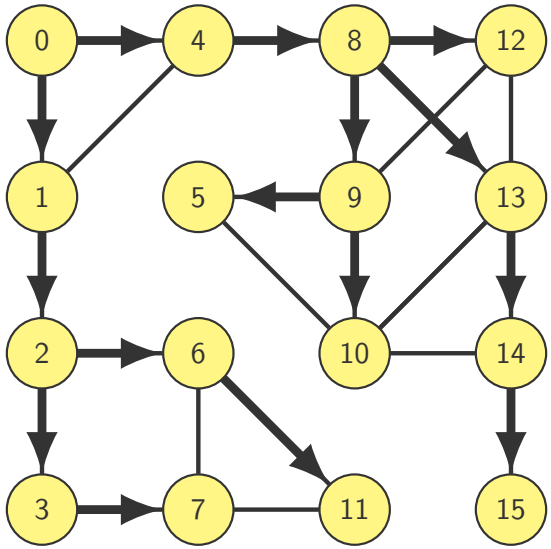
Fila

BFS usa fila



BFS usa fila

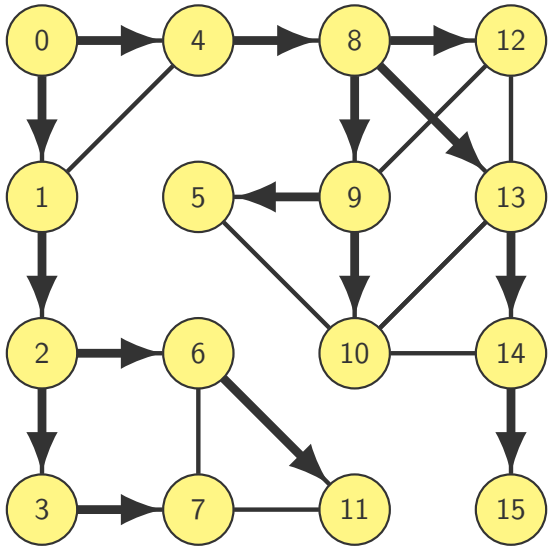
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



Usando uma fila, visitamos primeiro os vértices mais próximos

BFS usa fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

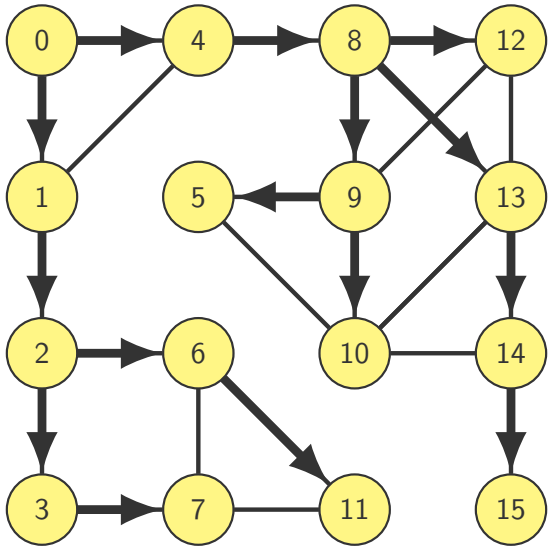


Usando uma fila, visitamos primeiro os vértices mais próximos

- Enfileiramos os vizinhos de 0 (que estão à distância 1)

BFS usa fila

Universidade Federal do Ceará
Campus Quixadá

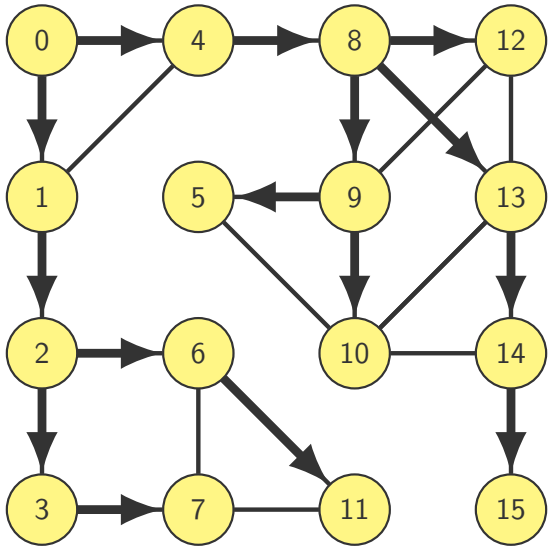


Usando uma fila, visitamos primeiro os vértices mais próximos

- Enfileiramos os vizinhos de 0 (que estão à distância 1)
- Desenfileiramos um de seus vizinhos

BFS usa fila

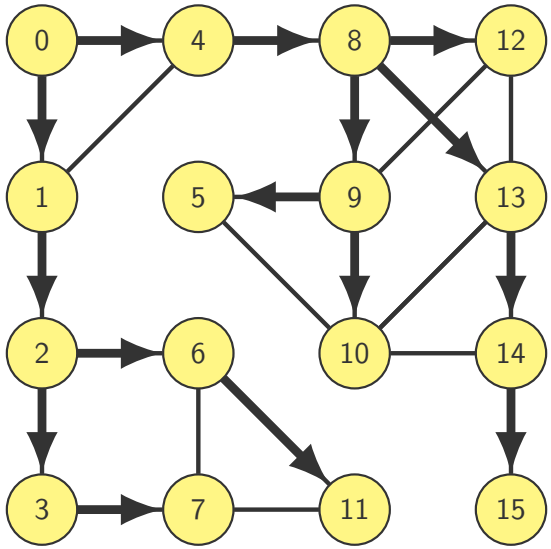
Universidade Federal do Ceará
Campus Quixadá



Usando uma fila, visitamos primeiro os vértices mais próximos

- Enfileiramos os vizinhos de 0 (que estão à distância 1)
- Desenfileiramos um de seus vizinhos
- E enfileiramos os vizinhos deste vértice

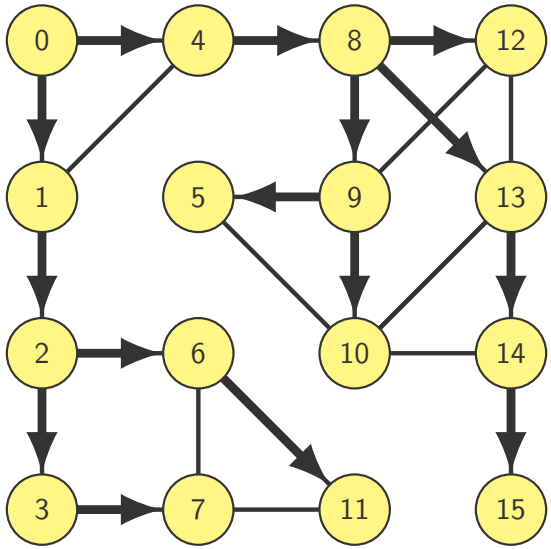
BFS usa fila



Usando uma fila, visitamos primeiro os vértices mais próximos

- Enfileiramos os vizinhos de 0 (que estão à distância 1)
- Desenfileiramos um de seus vizinhos
- E enfileiramos os vizinhos deste vértice
  - que estão a distância 2 de 0

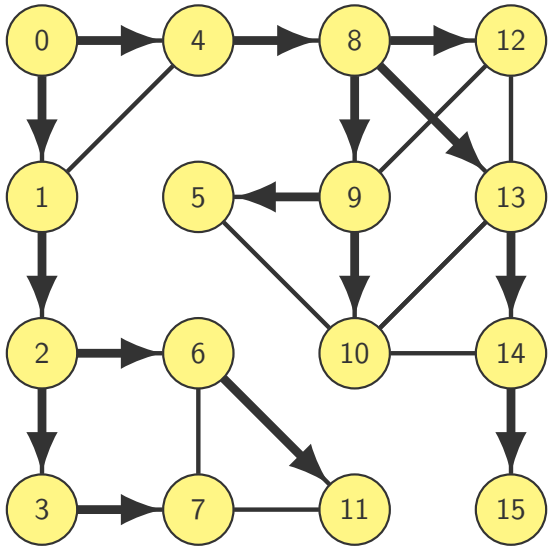
BFS usa fila



Usando uma fila, visitamos primeiro os vértices mais próximos

- Enfileiramos os vizinhos de 0 (que estão à distância 1)
- Desenfileiramos um de seus vizinhos
- E enfileiramos os vizinhos deste vértice
  - que estão a distância 2 de 0
- Assim por diante...

BFS usa fila



Usando uma fila, visitamos primeiro os vértices mais próximos

- Enfileiramos os vizinhos de 0 (que estão à distância 1)
- Desenfileiramos um de seus vizinhos
- E enfileiramos os vizinhos deste vértice
  - que estão a distância 2 de 0
- Assim por diante...

A árvore nos dá um caminho mínimo entre raiz e vértice

Algoritmo BFS

Busca em Largura – Elementos

BFS atribui cores aos vértices:

- Vértice branco: ainda não descoberto.
- Vértice cinza: descoberto, mas não explorado.
- Vértice preto: completamente explorado.

Busca em Largura – Elementos

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

BFS atribui cores aos vértices:

- Vértice branco: ainda não descoberto.
- Vértice cinza: descoberto, mas não explorado.
- Vértice preto: completamente explorado.

BFS descreve via vetor de predecessores π uma árvore de busca em largura.

- $\pi[u]$: pai de u na árvore.

Busca em Largura – Elementos

BFS atribui cores aos vértices:

- Vértice branco: ainda não descoberto.
- Vértice cinza: descoberto, mas não explorado.
- Vértice preto: completamente explorado.

BFS descreve via vetor de predecessores π uma árvore de busca em largura.

- $\pi[u]$: pai de u na árvore.

BFS guarda a distância de um vértice à origem:

- $d[u]$: distância da origem s a u.
- BFS descobre todos os vértices à distância k antes de descobrir qualquer um à distância $k + 1$.

BFS(G, s)

1. **para todo** $u \in V(G)$ **faça**
2. $\quad cor[u] \leftarrow$ BRANCO
3. $\quad \pi[u] \leftarrow$ NIL
4. $\quad d[u] \leftarrow \infty$
5. $cor[s] \leftarrow$ CINZA
6. $d[s] \leftarrow 0$
7. cria fila vazia $Q = \emptyset$
8. Enqueue(Q, s)
9. **enquanto** $Q \neq \emptyset$ **faça**
10. $\quad u \leftarrow$ Dequeue(Q)
11. $\quad$ **para todo** $v \in Adj(u)$ **faça**
12. $\quad\quad$ **se** $cor[v] ==$ BRANCO
13. $\quad\quad\quad cor[v] \leftarrow$ CINZA
14. $\quad\quad\quad d[v] \leftarrow d[u] + 1$
15. $\quad\quad\quad \pi[v] \leftarrow u$
16. $\quad\quad\quad$ Enqueue(Q, v)
17. $\quad cor[u] \leftarrow$ PRETO

Análise de complexidade

Universidade Federal do Ceará
Campus Quixadá

Analisamos de forma agregada

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco
   - enfileiramos cada vértice no máximo uma vez

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco
   - enfileiramos cada vértice no máximo uma vez
   - desenfileiramos cada vértice no máximo uma vez

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco
   - enfileiramos cada vértice no máximo uma vez
   - desenfileiramos cada vértice no máximo uma vez
   - cada operação na fila leva tempo $O(1)$

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco
   - enfileiramos cada vértice no máximo uma vez
   - desenfileiramos cada vértice no máximo uma vez
   - cada operação na fila leva tempo $O(1)$
   - o tempo gasto com a fila é $O(V)$

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco
   - enfileiramos cada vértice no máximo uma vez
   - desenfileiramos cada vértice no máximo uma vez
   - cada operação na fila leva tempo $O(1)$
   - o tempo gasto com a fila é $O(V)$
3. processamos cada vértice uma vez

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco
   - enfileiramos cada vértice no máximo uma vez
   - desenfileiramos cada vértice no máximo uma vez
   - cada operação na fila leva tempo $O(1)$
   - o tempo gasto com a fila é $O(V)$
3. processamos cada vértice uma vez
   - cada lista de adjacências é percorrida uma vez

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco
   - enfileiramos cada vértice no máximo uma vez
   - desenfileiramos cada vértice no máximo uma vez
   - cada operação na fila leva tempo $O(1)$
   - o tempo gasto com a fila é $O(V)$
3. processamos cada vértice uma vez
   - cada lista de adjacências é percorrida uma vez
   - no pior caso, percorremos todas as listas

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco
   - enfileiramos cada vértice no máximo uma vez
   - desenfileiramos cada vértice no máximo uma vez
   - cada operação na fila leva tempo $O(1)$
   - o tempo gasto com a fila é $O(V)$
3. processamos cada vértice uma vez
   - cada lista de adjacências é percorrida uma vez
   - no pior caso, percorremos todas as listas
   - o tempo gasto percorrendo adjacências é $O(E)$

Análise de complexidade

Analisamos de forma agregada

1. o tempo de inicialização é $O(V)$
2. um vértice não volta a ser branco
   - enfileiramos cada vértice no máximo uma vez
   - desenfileiramos cada vértice no máximo uma vez
   - cada operação na fila leva tempo $O(1)$
   - o tempo gasto com a fila é $O(V)$
3. processamos cada vértice uma vez
   - cada lista de adjacências é percorrida uma vez
   - no pior caso, percorremos todas as listas
   - o tempo gasto percorrendo adjacências é $O(E)$

A complexidade da busca em largura é $O(V + E)$

Corretude do algoritmo

Correção do algoritmo

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e

Correção do algoritmo

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e
- $d[v] = dist(s, v)$, para todo $v \in V(G)$.

Correção do algoritmo

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e
- $d[v] = dist(s, v)$, para todo $v \in V(G)$.

Precisamos de dois lemas auxiliares:

Correção do algoritmo

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e
- $d[v] = dist(s, v)$, para todo $v \in V(G)$.

Precisamos de dois lemas auxiliares:

- Lema 1: o caminho de s a v na árvore tem tamanho $d[v]$

Correção do algoritmo

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e
- $d[v] = dist(s, v)$, para todo $v \in V(G)$.

Precisamos de dois lemas auxiliares:

- Lema 1: o caminho de s a v na árvore tem tamanho $d[v]$
- Lema 2: os vértices são inseridos na fila Q em ordem crescente de $d[v]$

Lema 1

Lema 1: Seja T a árvore induzida por π. Se $d[v] < \infty$, então

1. v é um vértice de T,

Lema 1

Lema 1: Seja T a árvore induzida por π. Se $d[v] < \infty$, então

1. v é um vértice de T,
2. o caminho de s a v em T tem comprimento $d[v]$.

Lema 1

Lema 1: Seja T a árvore induzida por π. Se $d[v] < \infty$, então

1. v é um vértice de T,
2. o caminho de s a v em T tem comprimento $d[v]$.

Demonstração:

Lema 1

Lema 1: Seja T a árvore induzida por π. Se $d[v] < \infty$, então

1. v é um vértice de T,
2. o caminho de s a v em T tem comprimento $d[v]$.

Demonstração:

- por indução no número de vezes que executamos **enqueue**

Lema 1

Lema 1: Seja T a árvore induzida por π. Se $d[v] < \infty$, então

1. v é um vértice de T,
2. o caminho de s a v em T tem comprimento $d[v]$.

Demonstração:

- por indução no número de vezes que executamos **enqueue**
- depois que executamos **enqueue** pela primeira vez

Lema 1

Lema 1: Seja T a árvore induzida por π. Se $d[v] < \infty$, então

1. v é um vértice de T,
2. o caminho de s a v em T tem comprimento $d[v]$.

Demonstração:

- por indução no número de vezes que executamos **enqueue**
- depois que executamos **enqueue** pela primeira vez
  - T continha apenas s e valia $d[s] = 0$

Lema 1

Lema 1: Seja T a árvore induzida por π. Se $d[v] < \infty$, então

1. v é um vértice de T,
2. o caminho de s a v em T tem comprimento $d[v]$.

Demonstração:

- por indução no número de vezes que executamos **enqueue**
- depois que executamos **enqueue** pela primeira vez
  - T continha apenas s e valia $d[s] = 0$
  - como $d[s]$ nunca mais muda, isso completa a base

Prova do lema

Considere o instante em que enfileiramos v

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u
- mas u já havia sido enfileirado antes desse instante

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u
- mas u já havia sido enfileirado antes desse instante
- pela hipótese de indução

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u
- mas u já havia sido enfileirado antes desse instante
- pela hipótese de indução
  1. existe um caminho de s a u em T

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u
- mas u já havia sido enfileirado antes desse instante
- pela hipótese de indução
  1. existe um caminho de s a u em T
  2. esse caminho tem comprimento $d[u]$

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u
- mas u já havia sido enfileirado antes desse instante
- pela hipótese de indução
  1. existe um caminho de s a u em T
  2. esse caminho tem comprimento $d[u]$
- mais isso implica que

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u
- mas u já havia sido enfileirado antes desse instante
- pela hipótese de indução
  1. existe um caminho de s a u em T
  2. esse caminho tem comprimento $d[u]$
- mais isso implica que
  1. há caminho de s a v em T, pois $\pi[v] = u$

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u
- mas u já havia sido enfileirado antes desse instante
- pela hipótese de indução
  1. existe um caminho de s a u em T
  2. esse caminho tem comprimento $d[u]$
- mais isso implica que
  1. há caminho de s a v em T, pois $\pi[v] = u$
  2. e esse caminho tem comprimento $d[v]$, pois $d[v] = d[u] + 1$

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u
- mas u já havia sido enfileirado antes desse instante
- pela hipótese de indução
  1. existe um caminho de s a u em T
  2. esse caminho tem comprimento $d[u]$
- mais isso implica que
  1. há caminho de s a v em T, pois $\pi[v] = u$
  2. e esse caminho tem comprimento $d[v]$, pois $d[v] = d[u] + 1$
- e completamos a indução

Prova do lema

Considere o instante em que enfileiramos v

- então v foi descoberto percorrendo os vizinhos de u
- mas u já havia sido enfileirado antes desse instante
- pela hipótese de indução
  1. existe um caminho de s a u em T
  2. esse caminho tem comprimento $d[u]$
- mais isso implica que
  1. há caminho de s a v em T, pois $\pi[v] = u$
  2. e esse caminho tem comprimento $d[v]$, pois $d[v] = d[u] + 1$
- e completamos a indução

Corolário 1: Durante BFS, $d[v] \geq dist(s, v)$ para todo $v \in V$.

Lema 2

Lema 2: Suponha que $\langle v_1, v_2, \ldots, v_r \rangle$ seja a disposição da fila Q em alguma iteração do algoritmo. Então

$$d[v_1] \leq d[v_2] \leq \cdots \leq d[v_r] \leq d[v_1] + 1.$$

Demonstração:

Lema 2

Lema 2: Suponha que $\langle v_1, v_2, \ldots, v_r \rangle$ seja a disposição da fila Q em alguma iteração do algoritmo. Então

$$d[v_1] \leq d[v_2] \leq \cdots \leq d[v_r] \leq d[v_1] + 1.$$

Demonstração:

- por indução no número de iterações do laço enquanto

Lema 2

Lema 2: Suponha que $\langle v_1, v_2, \ldots, v_r \rangle$ seja a disposição da fila Q em alguma iteração do algoritmo. Então

$$d[v_1] \leq d[v_2] \leq \cdots \leq d[v_r] \leq d[v_1] + 1.$$

Demonstração:

- por indução no número de iterações do laço enquanto
- antes da primeira iteração, $Q = \langle s \rangle$ e o lema vale

Prova do lema

Considere uma nova iteração do laço

Prova do lema

Considere uma nova iteração do laço

- antes da iteração a fila era $\langle v_1, v_2, \ldots, v_r \rangle$ (qual a H.I.?)

Prova do lema

Considere uma nova iteração do laço

- antes da iteração a fila era $\langle v_1, v_2, \ldots, v_r \rangle$ (qual a H.I.?)
- na iteração, removemos v_1 e inserimos $v_{r+1}, \ldots, v_{r+t}$

Prova do lema

Considere uma nova iteração do laço

- antes da iteração a fila era $\langle v_1, v_2, \ldots, v_r \rangle$ (qual a H.I.?)
- na iteração, removemos v_1 e inserimos $v_{r+1}, \ldots, v_{r+t}$
- no final da iteração a fila será $\langle v_2, \ldots, v_r, v_{r+1}, \ldots, v_{r+t} \rangle$

Prova do lema

Considere uma nova iteração do laço

- antes da iteração a fila era $\langle v_1, v_2, \ldots, v_r \rangle$ (qual a H.I.?)
- na iteração, removemos v_1 e inserimos $v_{r+1}, \ldots, v_{r+t}$
- no final da iteração a fila será $\langle v_2, \ldots, v_r, v_{r+1}, \ldots, v_{r+t} \rangle$

Inserimos vizinhos de v_1

Prova do lema

Considere uma nova iteração do laço

- antes da iteração a fila era $\langle v_1, v_2, \ldots, v_r \rangle$ (qual a H.I.?)
- na iteração, removemos v_1 e inserimos $v_{r+1}, \ldots, v_{r+t}$
- no final da iteração a fila será $\langle v_2, \ldots, v_r, v_{r+1}, \ldots, v_{r+t} \rangle$

Inserimos vizinhos de v_1

- se v_j é um vértice inserido, então $d[v_j] = d[v_1] + 1$

Prova do lema

Considere uma nova iteração do laço

- antes da iteração a fila era $\langle v_1, v_2, \ldots, v_r \rangle$ (qual a H.I.?)
- na iteração, removemos v_1 e inserimos $v_{r+1}, \ldots, v_{r+t}$
- no final da iteração a fila será $\langle v_2, \ldots, v_r, v_{r+1}, \ldots, v_{r+t} \rangle$

Inserimos vizinhos de v_1

- se v_j é um vértice inserido, então $d[v_j] = d[v_1] + 1$
- pela hipótese de indução

$$d[v_2] \leq \cdots \leq d[v_r] \leq d[v_1] + 1 \leq d[v_2] + 1$$

Prova do lema

Considere uma nova iteração do laço

- antes da iteração a fila era $\langle v_1, v_2, \ldots, v_r \rangle$ (qual a H.I.?)
- na iteração, removemos v_1 e inserimos $v_{r+1}, \ldots, v_{r+t}$
- no final da iteração a fila será $\langle v_2, \ldots, v_r, v_{r+1}, \ldots, v_{r+t} \rangle$

Inserimos vizinhos de v_1

- se v_j é um vértice inserido, então $d[v_j] = d[v_1] + 1$
- pela hipótese de indução

$$d[v_2] \leq \cdots \leq d[v_r] \leq d[v_1] + 1 \leq d[v_2] + 1$$

- portanto

$$d[v_2] \leq \cdots \leq d[v_r] \leq d[v_{r+1}] \leq \cdots \leq d[v_{r+t}] \leq d[v_2] + 1$$

Prova do teorema

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e

Prova do teorema

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e
- $d[v] = dist(s, v)$, para todo $v \in V(G)$.

Prova do teorema

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e
- $d[v] = dist(s, v)$, para todo $v \in V(G)$.

Demonstração

Prova do teorema

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e
- $d[v] = dist(s, v)$, para todo $v \in V(G)$.

Demonstração

- Sabemos que π define uma árvore enraizada em s e, pelo Lema 1, o caminho de s a v na árvore tem comprimento $d[v]$

Prova do teorema

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e
- $d[v] = dist(s, v)$, para todo $v \in V(G)$.

Demonstração

- Sabemos que π define uma árvore enraizada em s e, pelo Lema 1, o caminho de s a v na árvore tem comprimento $d[v]$
- também, se $dist(s, v) = \infty$, então $d[v] = \infty$ pelo Corolário 1

Prova do teorema

Teorema: Seja $G = (V, E)$ um grafo e s um vértice de G. Então, depois de executar bfs(G, s), temos que:

- π define árvore com caminho mínimo de s a v em G e
- $d[v] = dist(s, v)$, para todo $v \in V(G)$.

Demonstração

- Sabemos que π define uma árvore enraizada em s e, pelo Lema 1, o caminho de s a v na árvore tem comprimento $d[v]$
- também, se $dist(s, v) = \infty$, então $d[v] = \infty$ pelo Corolário 1
- resta provar que se $dist(s, v) < \infty$, então $d[v] = dist(s, v)$

Prova do teorema (cont)

Considere um vértice v com $dist(s, v) = k$

Prova do teorema (cont)

Considere um vértice v com $dist(s, v) = k$

- iremos provar que $d[v] = k$ por indução em k

Prova do teorema (cont)

Considere um vértice v com $dist(s, v) = k$

- iremos provar que $d[v] = k$ por indução em k

Caso Base:

Prova do teorema (cont)

Considere um vértice v com $dist(s, v) = k$

- iremos provar que $d[v] = k$ por indução em k

Caso Base:

- se $k = 0$, devemos ter $v = s$ e a afirmação vale.

Prova do teorema (cont)

Considere um vértice v com $dist(s, v) = k$

- iremos provar que $d[v] = k$ por indução em k

Caso Base:

- se $k = 0$, devemos ter $v = s$ e a afirmação vale.

Hipótese indutiva: Suponha que, para todo u com $dist(s, u) < k$, temos que $d[u] = dist(s, u)$

Prova do teorema (cont)

Considere um vértice v com $dist(s, v) = k$

- iremos provar que $d[v] = k$ por indução em k

Caso Base:

- se $k = 0$, devemos ter $v = s$ e a afirmação vale.

Hipótese indutiva: Suponha que, para todo u com $dist(s, u) < k$, temos que $d[u] = dist(s, u)$

Passo indutivo: considere um caminho de s a v de comprimento k

Prova do teorema (cont)

Considere um vértice v com $dist(s, v) = k$

- iremos provar que $d[v] = k$ por indução em k

Caso Base:

- se $k = 0$, devemos ter $v = s$ e a afirmação vale.

Hipótese indutiva: Suponha que, para todo u com $dist(s, u) < k$, temos que $d[u] = dist(s, u)$

Passo indutivo: considere um caminho de s a v de comprimento k

- chame de u o vértice que antecede v nesse caminho

Prova do teorema (cont)

Considere um vértice v com $dist(s, v) = k$

- iremos provar que $d[v] = k$ por indução em k

Caso Base:

- se $k = 0$, devemos ter $v = s$ e a afirmação vale.

Hipótese indutiva: Suponha que, para todo u com $dist(s, u) < k$, temos que $d[u] = dist(s, u)$

Passo indutivo: considere um caminho de s a v de comprimento k

- chame de u o vértice que antecede v nesse caminho
- daí $dist(s, u) = k - 1$ e portanto $d[u] = k - 1$

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Portanto, nesse instante, v era branco ou cinza

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Portanto, nesse instante, v era branco ou cinza

- se v era branco

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Portanto, nesse instante, v era branco ou cinza

- se v era branco
  - v será inserido na fila nessa iteração

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Portanto, nesse instante, v era branco ou cinza

- se v era branco
  - v será inserido na fila nessa iteração
  - e teremos $d[v] = d[u] + 1 = k$

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Portanto, nesse instante, v era branco ou cinza

- se v era branco
  - v será inserido na fila nessa iteração
  - e teremos $d[v] = d[u] + 1 = k$
- se v era cinza

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Portanto, nesse instante, v era branco ou cinza

- se v era branco
  - v será inserido na fila nessa iteração
  - e teremos $d[v] = d[u] + 1 = k$
- se v era cinza
  - v já estava na fila nesse instante

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Portanto, nesse instante, v era branco ou cinza

- se v era branco
  - v será inserido na fila nessa iteração
  - e teremos $d[v] = d[u] + 1 = k$
- se v era cinza
  - v já estava na fila nesse instante
  - então o Lema 2 implica $d[v] \leq d[u] + 1 = k$

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Portanto, nesse instante, v era branco ou cinza

- se v era branco
  - v será inserido na fila nessa iteração
  - e teremos $d[v] = d[u] + 1 = k$
- se v era cinza
  - v já estava na fila nesse instante
  - então o Lema 2 implica $d[v] \leq d[u] + 1 = k$
  - como $k \leq d[v]$, temos $d[v] = k$

Prova do teorema (cont)

Considere o instante em que u foi removido da fila Q

- suponha por contradição que v seja preto
- daí v foi removido de Q antes de u
- então o Lema 2 implica que $d[v] \leq d[u] < k$
- mas o Corolário 1 implica que $k = dist(s, v) \leq d[v]$
- isso é uma contradição, então v não pode ser preto

Portanto, nesse instante, v era branco ou cinza

- se v era branco
  - v será inserido na fila nessa iteração
  - e teremos $d[v] = d[u] + 1 = k$
- se v era cinza
  - v já estava na fila nesse instante
  - então o Lema 2 implica $d[v] \leq d[u] + 1 = k$
  - como $k \leq d[v]$, temos $d[v] = k$
- em qualquer caso, concluímos a indução

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

FIM