GRAPHS

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

Centro de Informática UFPE

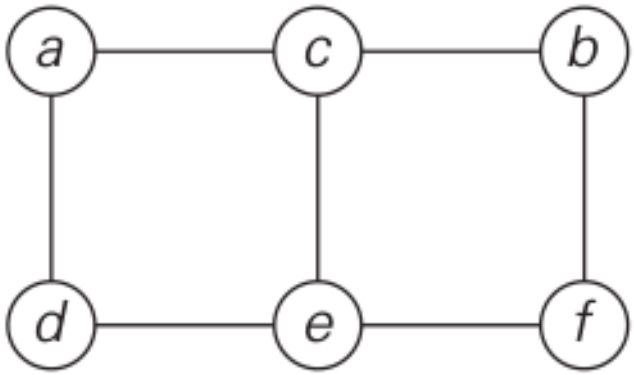
Universidade Federal de Pernambuco

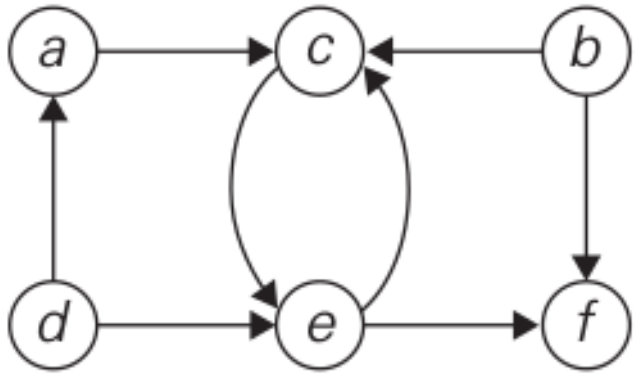
Agenda

Introduction[1]

Graph: a collection of nodes (vertices), some of them connected by edges (arcs) – $G = (V, E)$, where $V \neq \emptyset$, and $E \in V \leftrightarrow V$

- Undirected / Directed graphs (digraphs)





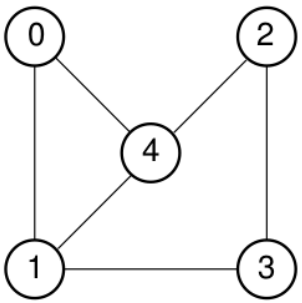
Centro de Informática UFPE

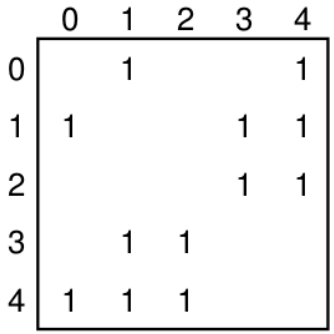
Universidade Federal de Pernambuco

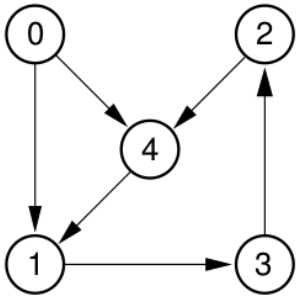
[1] Source: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

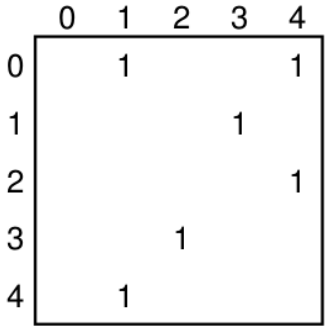
Possible implementations (unweighted graphs)[2]

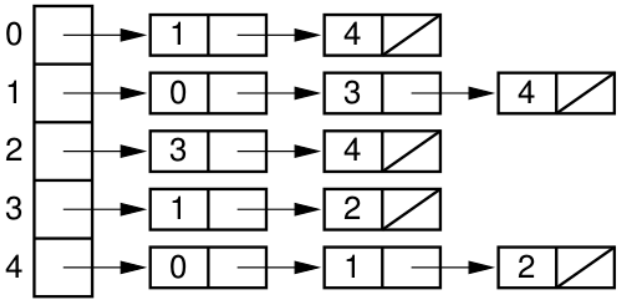
Adjacency matrix or adjacency list

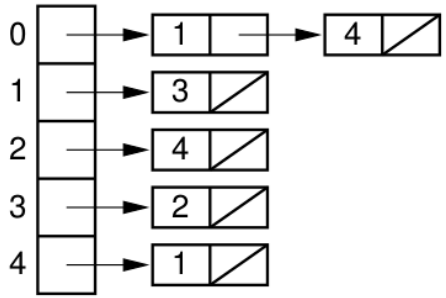








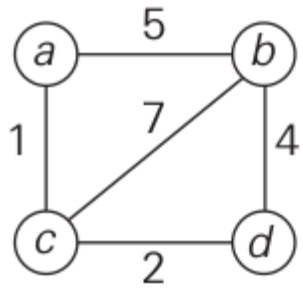




[2] Source: C. Shaffer. Data Structures and Algorithm Analysis. 2013.

Possible implementations (weighted graphs)[4]

Adjacency matrix or adjacency list



	a	b	c	d
a	∞	5	1	∞
b	5	∞	7	4
c	1	7	∞	2
d	∞	4	2	∞

a	→ b, 5 → c, 1
b	→ a, 5 → c, 7 → d, 4
c	→ a, 1 → b, 7 → d, 2
d	→ b, 4 → c, 2

[3]

Choosing the most appropriate implementation

- Space efficiency: dense graph *vs.* sparse graph
- Time efficiency (graph traversal): $\Theta(|V|^2)$ *vs.* $\Theta(|V| + |E|)$

[3] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

[4] Source: C. Shaffer. Data Structures and Algorithm Analysis. 2013.

Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph ADT

Operations

- int n(G g);
- int e(G g);
- int first(G g, int v);
- int next(G g, int v, int w);
- void setEdge(G g, int i, int j, int wt);
- void delEdge(G g, int i, int j);
- boolean isEdge(G g, int i, int j);
- int weight(G g, int i, int j);
- void setMark(G g, int v, int val);
- int getMark(G g, int v);

Graph ADT implemented as an adjacency matrix

Composite type:

```
1  int[][] matrix ;                          // adjacency matrix
2  int numEdge ;                              // number of edges
3  int[] Mark ;                          // auxiliary marking array
```

Algorithm: G create_graph(int n)

```
1  g.Mark ← new int[n];
2  g.matrix ← new int[n][n];
3  g.numEdge ← 0;
4  return g;
```

Graph ADT implemented as an adjacency matrix

Algorithm: int first(G g, int v)

```
1  for i ← 0 to n(g) − 1 do
2      if g.matrix[v][i] ≠ 0 then return i;
3  return n(g);
```

Algorithm: int next(G g, int v, int w)

```
1  for i ← w + 1 to n(g) − 1 do
2      if g.matrix[v][i] ≠ 0 then return i;
3  return n(g);
```

Implicit assumption: 0 denotes the absence of an edge between two nodes

Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph ADT implemented as an adjacency matrix

Algorithm: void setEdge(G g, int i, int j, int wt)

```
1  if wt = 0 then error ;
2  if g.matrix[i][j] = 0 then g.numEdge++;
3  g.matrix[i][j] ← wt;
```

Algorithm: void delEdge(G g, int i, int j)

```
1  if g.matrix[i][j] ≠ 0 then g.numEdge--;
2  g.matrix[i][j] ← 0;
```

Implicit assumption: 0 denotes the absence of an edge between two nodes

Centro de Informática UFPE

Universidade Federal de Pernambuco

Agenda

Graph traversals

In general terms (attention to unconnected graphs):

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED then
5          traverse(g, v);
```

Possible traversals

- DFS: depth-first search
- BFS: breadth-first search

Centro de Informática UFPE

Universidade Federal de Pernambuco

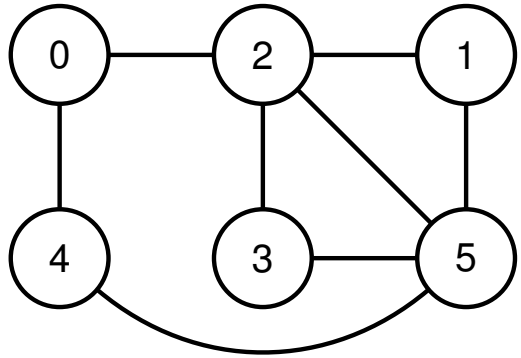
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	×	×	×	×	×	×

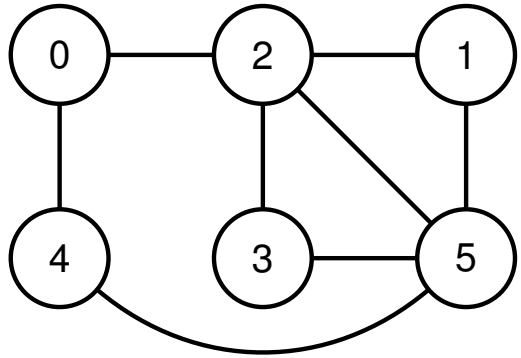
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	×	×	×	×	×	×

DFS calls:
- DFS(_, 0)

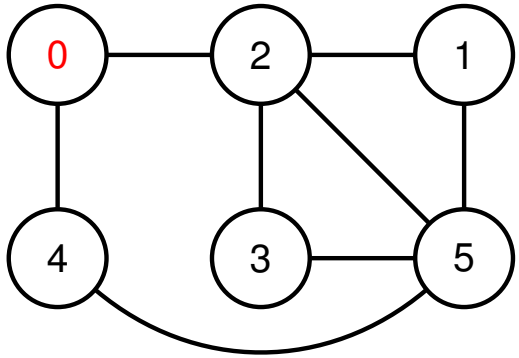
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	×	×	×	×	×

DFS calls:
- DFS(_, 0)

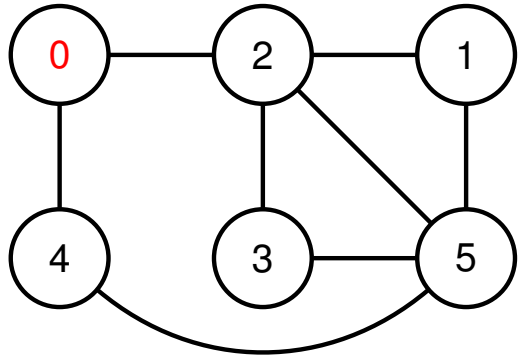
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	×	×	×	×	×

DFS calls:
- DFS(_, 0)
- DFS(_, 2)

Centro de Informática UFPE

Universidade Federal de Pernambuco

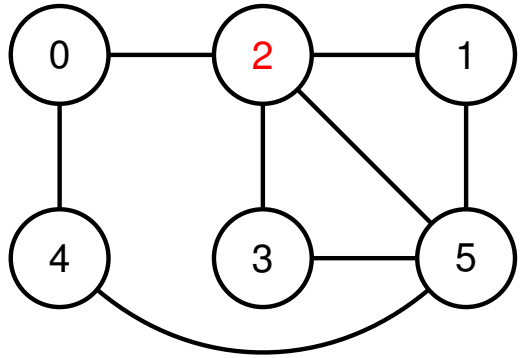
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	×	✓	×	×	×

DFS calls:
- DFS(_, 0)
- DFS(_, 2)

Centro de Informática UFPE

Universidade Federal de Pernambuco

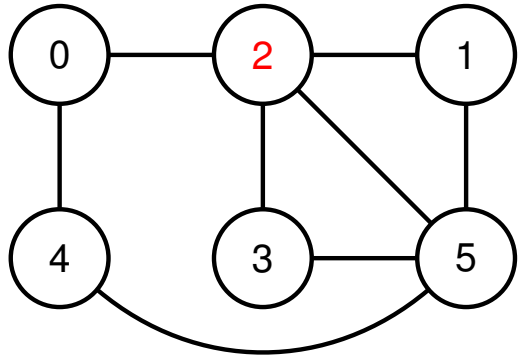
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	×	✓	×	×	×

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)

Centro de Informática UFPE

Universidade Federal de Pernambuco

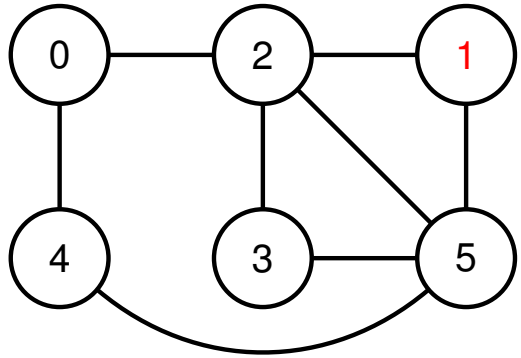
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	×	×	×

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)

Centro de Informática UFPE

Universidade Federal de Pernambuco

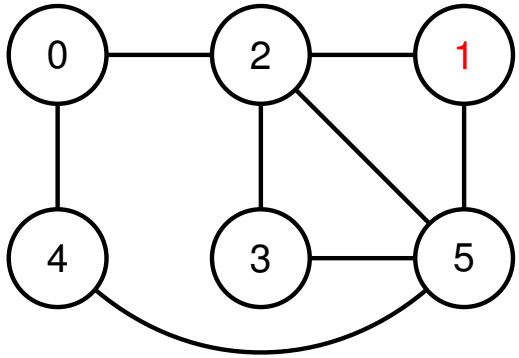
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	×	×	×

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)
- DFS(_, 5)

Centro de Informática UFPE

Universidade Federal de Pernambuco

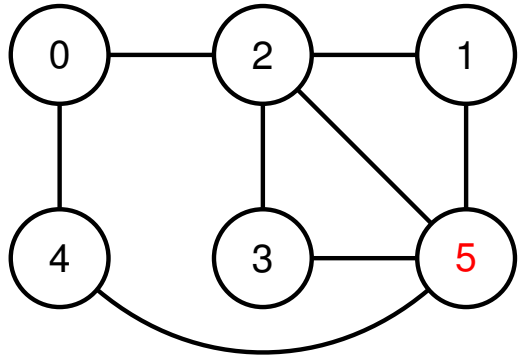
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	×	×	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)
- DFS(_, 5)

Centro de Informática UFPE

Universidade Federal de Pernambuco

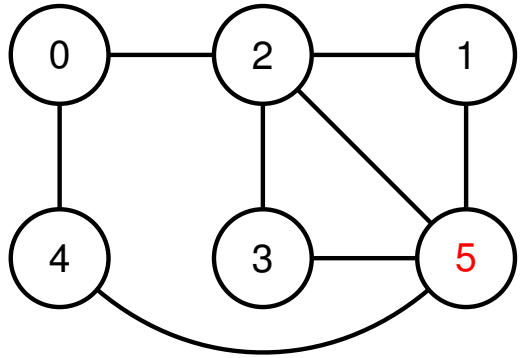
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	×	×	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)
- DFS(_, 5)
- DFS(_, 3)

Centro de Informática UFPE

Universidade Federal de Pernambuco

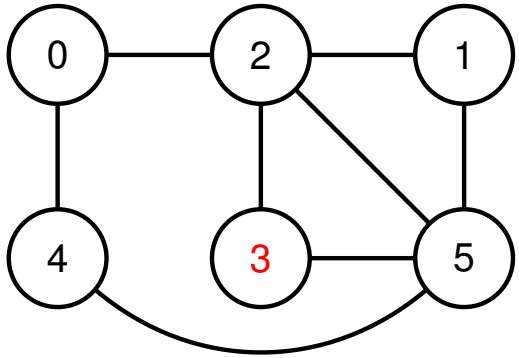
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	×	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)
- DFS(_, 5)
- DFS(_, 3)

Centro de Informática UFPE

Universidade Federal de Pernambuco

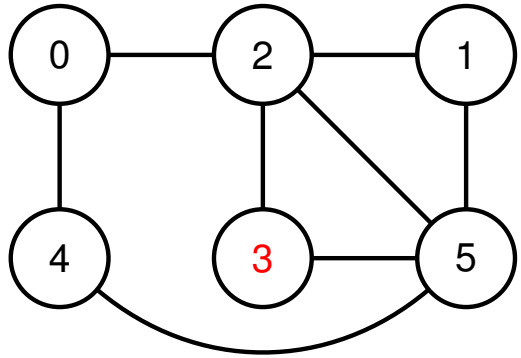
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	×	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)
- DFS(_, 5)
- DFS(_, 3)

Centro de Informática UFPE

Universidade Federal de Pernambuco

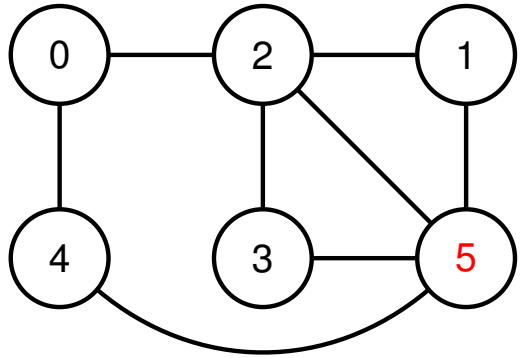
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	×	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)
- DFS(_, 5)
- DFS(_, 4)

Centro de Informática UFPE

Universidade Federal de Pernambuco

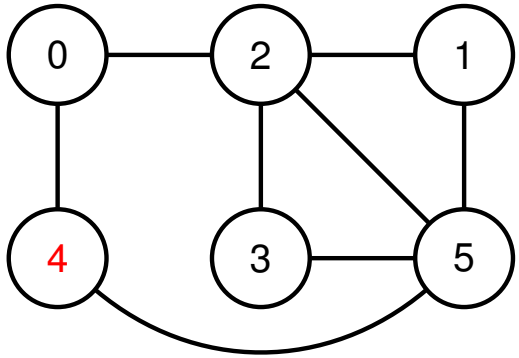
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)
- DFS(_, 5)
- DFS(_, 4)

Centro de Informática UFPE

Universidade Federal de Pernambuco

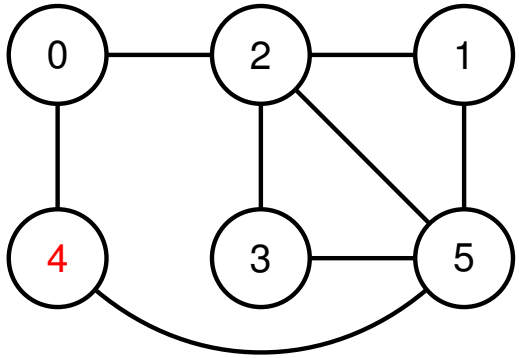
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)
- DFS(_, 5)
- DFS(_, 4)

Centro de Informática UFPE

Universidade Federal de Pernambuco

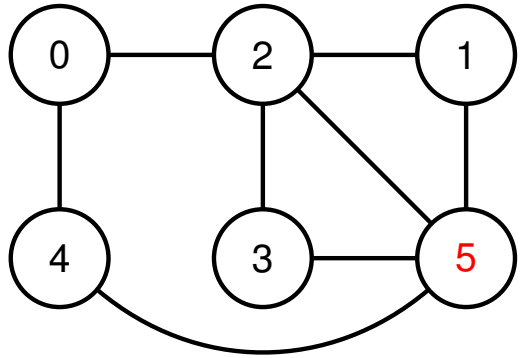
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)
- DFS(_, 5)

Centro de Informática UFPE

Universidade Federal de Pernambuco

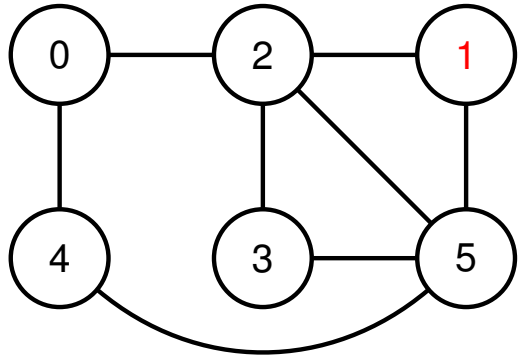
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)
- DFS(_, 1)

Centro de Informática UFPE

Universidade Federal de Pernambuco

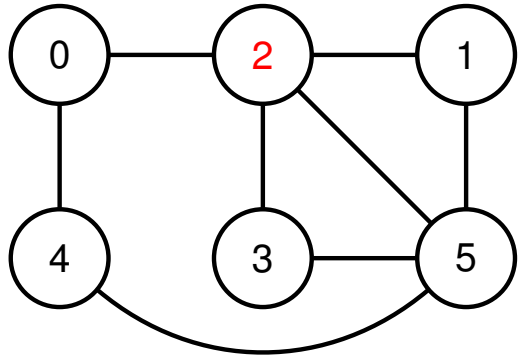
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

DFS calls:
- DFS(_, 0)
- DFS(_, 2)

Centro de Informática UFPE

Universidade Federal de Pernambuco

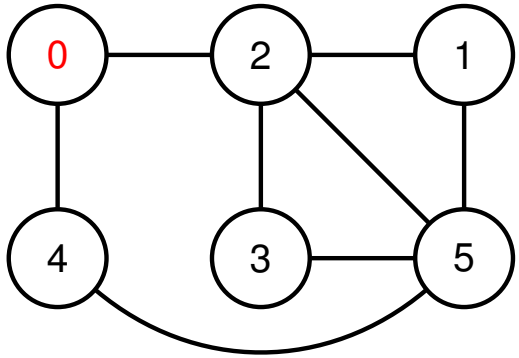
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

DFS calls:
- DFS(_, 0)

Centro de Informática UFPE

Universidade Federal de Pernambuco

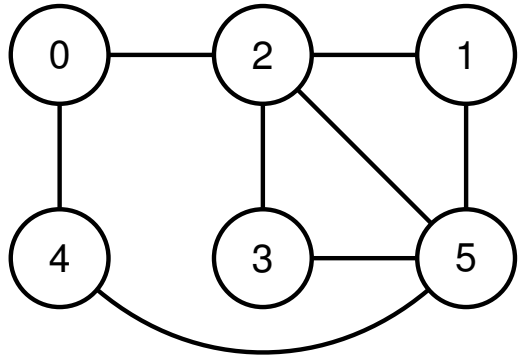
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

Centro de Informática UFPE

Universidade Federal de Pernambuco

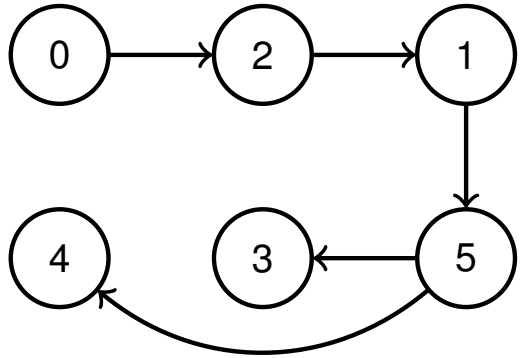
Graph traversals: DFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then DFS(g, v) ;
```

Algorithm: void DFS(G g, int v)

```
1  preVisit(g, v);
2  setMark(g, v, VISITED);
3  w ← first(g, v);
4  while w < n(g) do
5      if getMark(g, w) = UNVISITED
         then DFS(g, w) ;
6      w ← next(g, v, w);
7  posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

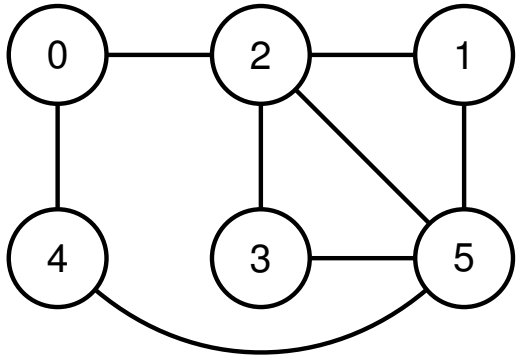
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then BFS(g, v) ;
```

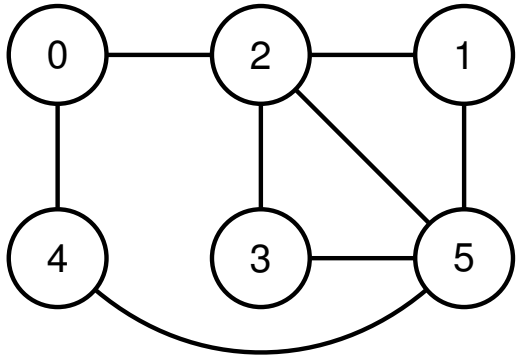


	0	1	2	3	4	5
Mark	×	×	×	×	×	×

Graph traversals: BFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then BFS(g, v) ;
```



	0	1	2	3	4	5
Mark	×	×	×	×	×	×

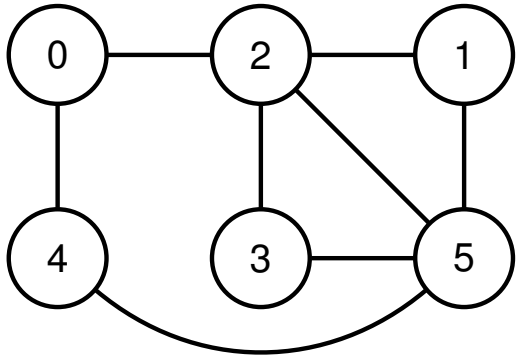
BFS calls:

- BFS(_, 0)

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1   Q ← create_queue();
2   enqueue(Q, start);
3   setMark(g, start, VISITED);
4   while length(Q) > 0 do
5       v ← dequeue(Q);
6       preVisit(g, v);
7       w ← first(g, v);
8       while w < n(g) do
9           if getMark(g, w) =
              UNVISITED then
10              setMark(g, w, VISITED);
11              enqueue(Q, w);
12          w ← next(g, v, w);
13      posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	×	×	×	×	×

BFS calls:
- BFS(_, 0)

Queue: 0

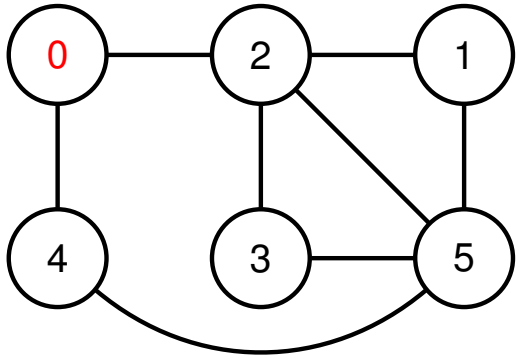
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1   Q ← create_queue();
2   enqueue(Q, start);
3   setMark(g, start, VISITED);
4   while length(Q) > 0 do
5       v ← dequeue(Q);
6       preVisit(g, v);
7       w ← first(g, v);
8       while w < n(g) do
9           if getMark(g, w) =
              UNVISITED then
10              setMark(g, w, VISITED);
11              enqueue(Q, w);
12          w ← next(g, v, w);
13      posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	×	×	×	×	×

BFS calls:
- BFS(_, 0)

Queue: -

$v = 0$

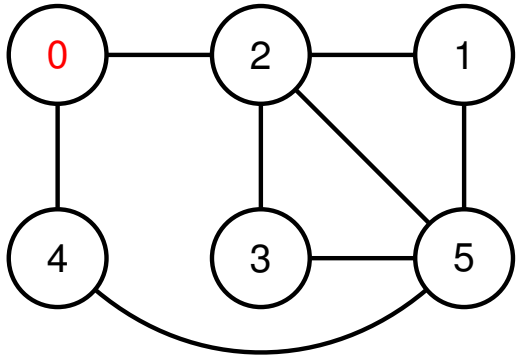
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1   Q ← create_queue();
2   enqueue(Q, start);
3   setMark(g, start, VISITED);
4   while length(Q) > 0 do
5       v ← dequeue(Q);
6       preVisit(g, v);
7       w ← first(g, v);
8       while w < n(g) do
9           if getMark(g, w) =
              UNVISITED then
10              setMark(g, w, VISITED);
11              enqueue(Q, w);
12          w ← next(g, v, w);
13      posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	×	✓	×	✓	×

BFS calls:
- BFS(_, 0)

Queue: 2, 4

$v = 0$

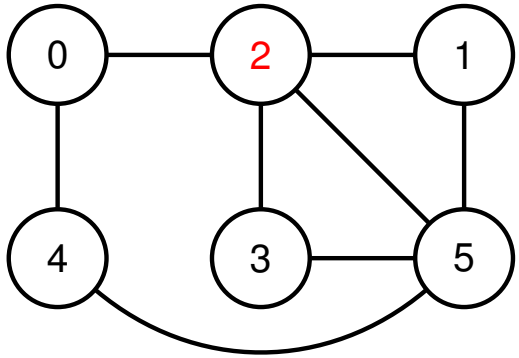
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1   Q ← create_queue();
2   enqueue(Q, start);
3   setMark(g, start, VISITED);
4   while length(Q) > 0 do
5       v ← dequeue(Q);
6       preVisit(g, v);
7       w ← first(g, v);
8       while w < n(g) do
9           if getMark(g, w) =
              UNVISITED then
10              setMark(g, w, VISITED);
11              enqueue(Q, w);
12          w ← next(g, v, w);
13      posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	×	✓	×	✓	×

BFS calls:
- BFS(_, 0)

Queue: 4

$v = 2$

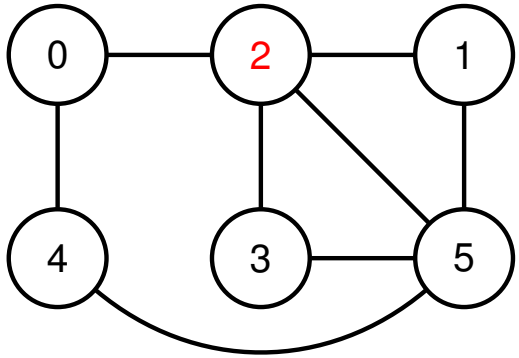
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1  Q ← create_queue();
2  enqueue(Q, start);
3  setMark(g, start, VISITED);
4  while length(Q) > 0 do
5      v ← dequeue(Q);
6      preVisit(g, v);
7      w ← first(g, v);
8      while w < n(g) do
9          if getMark(g, w) =
             UNVISITED then
10             setMark(g, w, VISITED);
11             enqueue(Q, w);
12         w ← next(g, v, w);
13     posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

BFS calls:
- BFS(_, 0)

Queue: 4, 1, 3, 5

$v = 2$

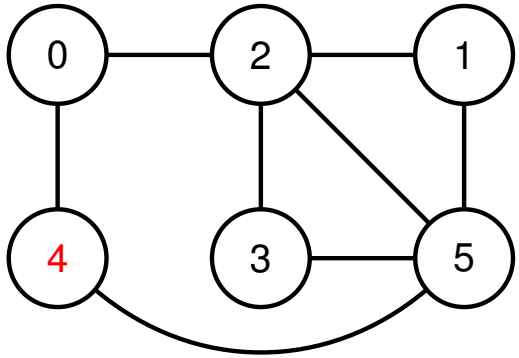
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1   Q ← create_queue();
2   enqueue(Q, start);
3   setMark(g, start, VISITED);
4   while length(Q) > 0 do
5       v ← dequeue(Q);
6       preVisit(g, v);
7       w ← first(g, v);
8       while w < n(g) do
9           if getMark(g, w) =
              UNVISITED then
10              setMark(g, w, VISITED);
11              enqueue(Q, w);
12          w ← next(g, v, w);
13      posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

BFS calls:
- BFS(_, 0)

Queue: 1, 3, 5

$v = 4$

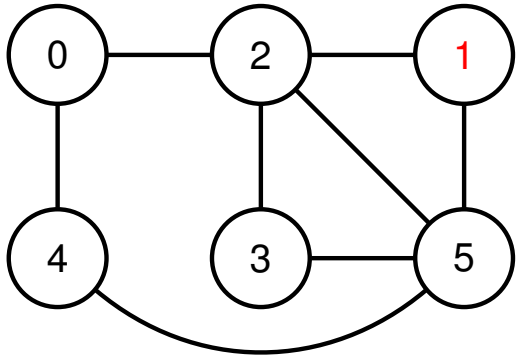
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1   Q ← create_queue();
2   enqueue(Q, start);
3   setMark(g, start, VISITED);
4   while length(Q) > 0 do
5       v ← dequeue(Q);
6       preVisit(g, v);
7       w ← first(g, v);
8       while w < n(g) do
9           if getMark(g, w) =
              UNVISITED then
10              setMark(g, w, VISITED);
11              enqueue(Q, w);
12          w ← next(g, v, w);
13      posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

BFS calls:
- BFS(_, 0)

Queue: 3, 5

$v = 1$

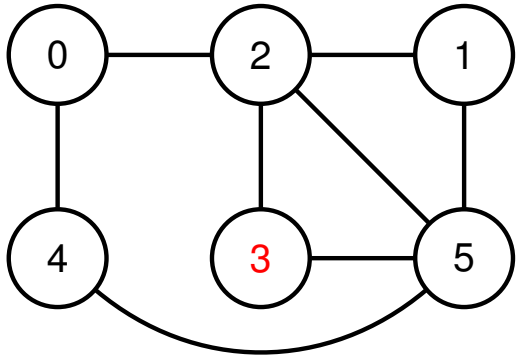
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1   Q ← create_queue();
2   enqueue(Q, start);
3   setMark(g, start, VISITED);
4   while length(Q) > 0 do
5       v ← dequeue(Q);
6       preVisit(g, v);
7       w ← first(g, v);
8       while w < n(g) do
9           if getMark(g, w) =
              UNVISITED then
10              setMark(g, w, VISITED);
11              enqueue(Q, w);
12          w ← next(g, v, w);
13      posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

BFS calls:
- BFS(_, 0)

Queue: 5

$v = 3$

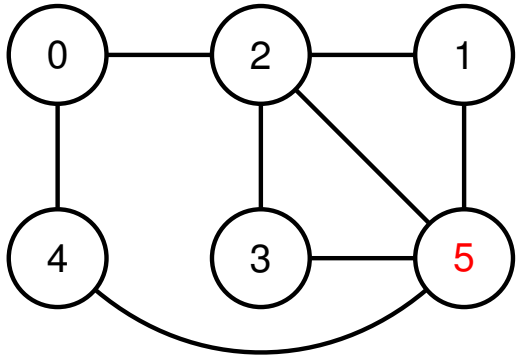
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1   Q ← create_queue();
2   enqueue(Q, start);
3   setMark(g, start, VISITED);
4   while length(Q) > 0 do
5       v ← dequeue(Q);
6       preVisit(g, v);
7       w ← first(g, v);
8       while w < n(g) do
9           if getMark(g, w) =
              UNVISITED then
10              setMark(g, w, VISITED);
11              enqueue(Q, w);
12          w ← next(g, v, w);
13      posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

BFS calls:
- BFS(_, 0)

Queue: -

$v = 5$

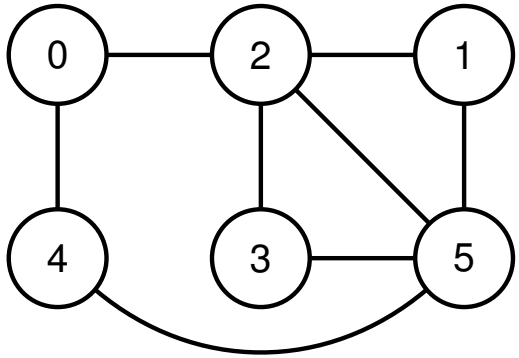
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void
BFS(G g, int start)

```
1   Q ← create_queue();
2   enqueue(Q, start);
3   setMark(g, start, VISITED);
4   while length(Q) > 0 do
5       v ← dequeue(Q);
6       preVisit(g, v);
7       w ← first(g, v);
8       while w < n(g) do
9           if getMark(g, w) =
              UNVISITED then
10              setMark(g, w, VISITED);
11              enqueue(Q, w);
12          w ← next(g, v, w);
13      posVisit(g, v);
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

BFS calls:
- BFS(_, 0)

Queue: -

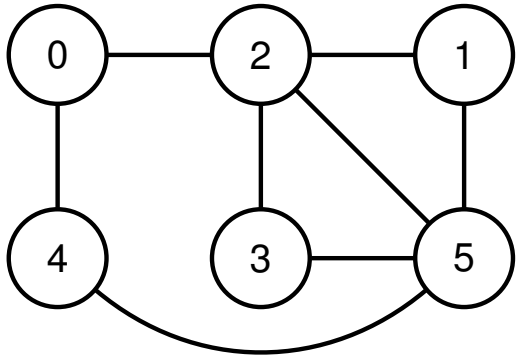
Centro de Informática UFPE

Universidade Federal de Pernambuco

Graph traversals: BFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then BFS(g, v) ;
```

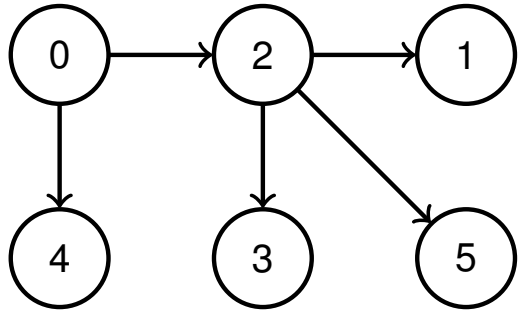


	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

Graph traversals: BFS

Algorithm: void graphTraverse(G g)

```
1  for v ← 0 to n(g) − 1 do
2      setMark(g, v, UNVISITED);
3  for v ← 0 to n(g) − 1 do
4      if getMark(g, v) = UNVISITED
         then BFS(g, v) ;
```



	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓

Centro de Informática UFPE

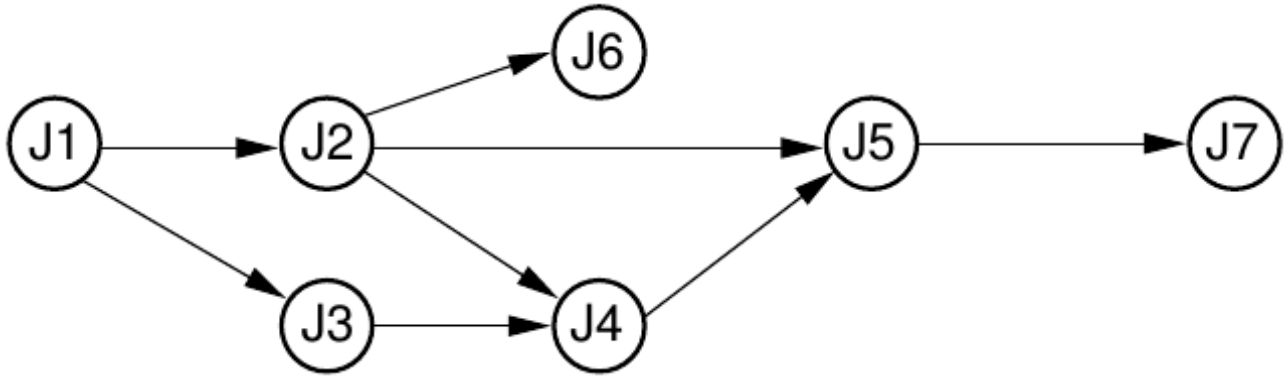
Universidade Federal de Pernambuco

Agenda

DFS: topological sorting[5]

Let G be a DAG (directed acyclic graph), find a solution considering the informed prerequisites

- Assumption: $u \rightarrow v$ means that u is a prerequisite of v



[5] Source: C. Shaffer. Data Structures and Algorithm Analysis. 2013.

DFS: topological sorting

Algorithm: void toposort(G g, int v, STACK s)

```
1  setMark(g, v, VISITED);
2  w ← first(g, v);
3  while w < n(g) do
4      if getMark(g, w) = UNVISITED then
5          toposort(g, w, s);
6      w ← next(g, v, w);
7  push(s, v);            // we will have in s a solution
```

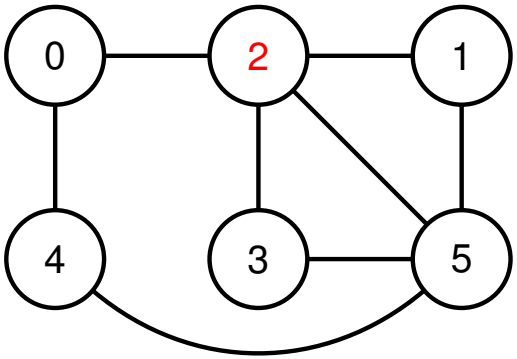
For the given example: J1, J3, J2, J6, J4, J5, J7

BFS: shortest path on unweighted graphs

Similar to the original BFS code

- Start the search on the origin
- When the destination is reached, stop the search
- To reconstruct the path: auxiliary array to store predecessors

Let the origin be 0 and the destination be 5, solution = $0 \rightarrow 2 \rightarrow 5$



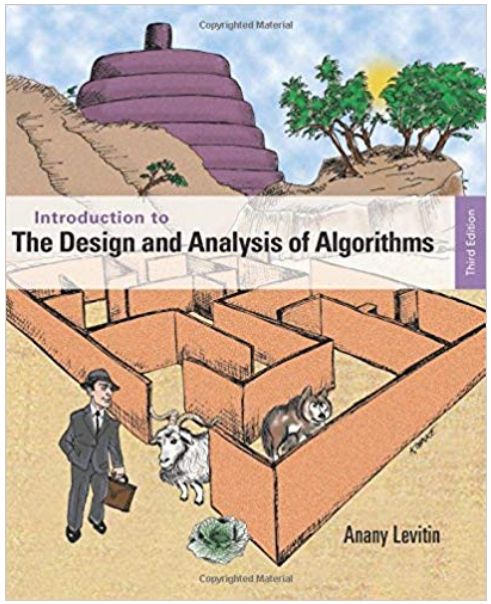
	0	1	2	3	4	5
Mark	✓	✓	✓	✓	✓	✓
Pred	-	2	0	2	0	2

Centro de Informática UFPE

Universidade Federal de Pernambuco

Agenda

Bibliography



Chapter 1 (pp. 28–31)
Chapter 3 (pp. 122–128)
Chapter 4 (pp. 138–141)
Anany Levitin.
Introduction to the Design and Analysis of Algorithms.
3rd edition. Pearson. 2011.

Chapter 11 (pp. 371–388)
Clifford Shaffer.
Data Structures and Algorithm Analysis. Dover, 2013.

Centro de Informática UFPE

Universidade Federal de Pernambuco

GRAPHS

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

Centro de Informática UFPE

Universidade Federal de Pernambuco