ITS202: Algorithms and Data Structure

Graphs

Ms. Sonam Wangmo

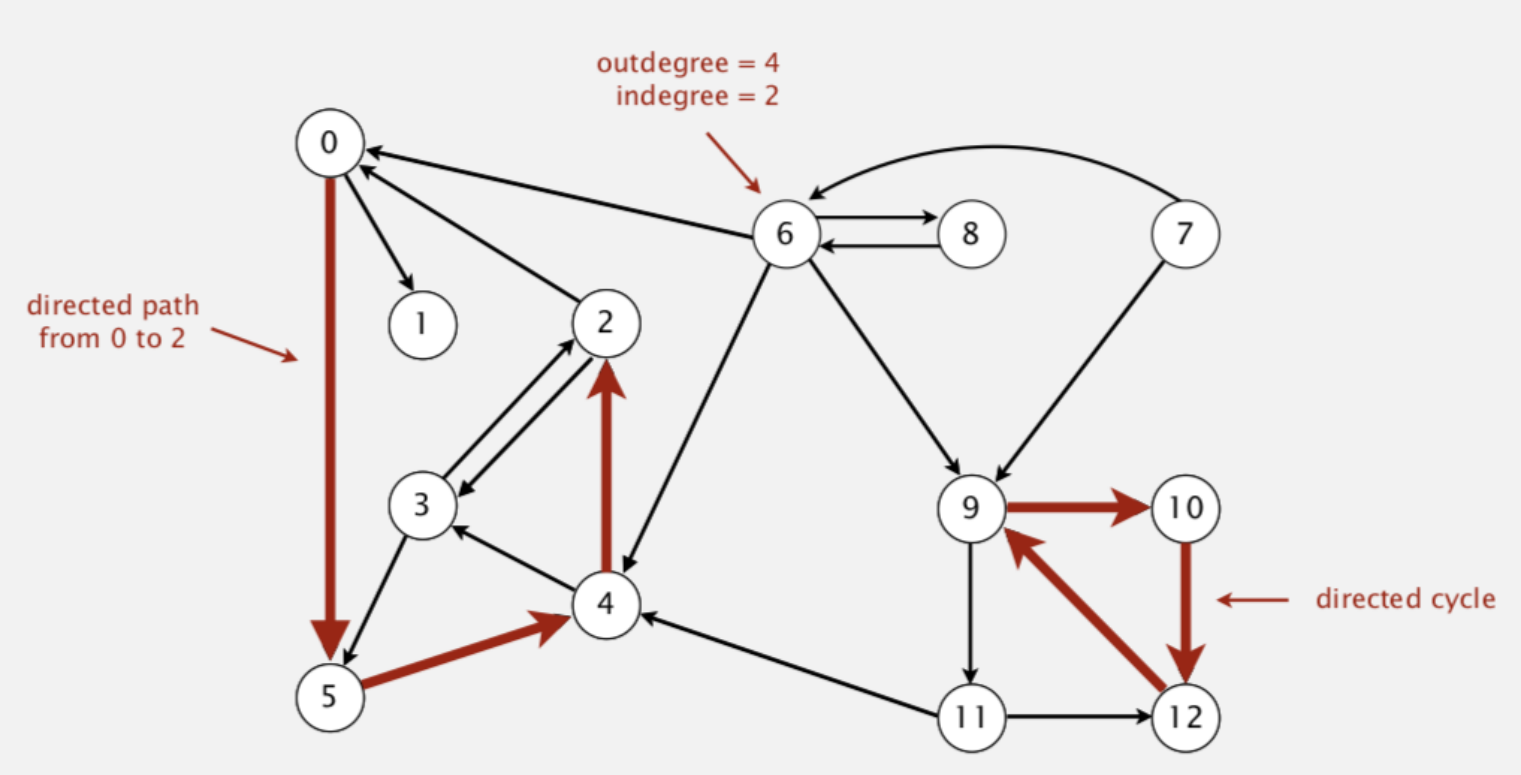
Gyalpozhing College of Information Technology
Royal University of Bhutan

December 3, 2020

Graphs: Digraph

Directed Graph

Digraph. Set of vertices connected pairwise by directed edges.



Graphs: Digraph Application: Road Network

digraph	vertex	directed edge
transportation	street intersection	one-way street
web	web page	hyperlink
food web	species	predator-prey relationship
WordNet	synset	hypernym
scheduling	task	precedence constraint
financial	bank	transaction
cell phone	person	placed call
infectious disease	person	infection
game	board position	legal move
citation	journal article	citation
object graph	object	pointer
inheritance hierarchy	class	inherits from
control flow	code block	jump

Graphs: Some digraph problems

problem	description
s→t path	*Is there a path from s to t ?*
shortest s→t path	*What is the shortest path from s to t ?*
directed cycle	*Is there a directed cycle in the graph ?*

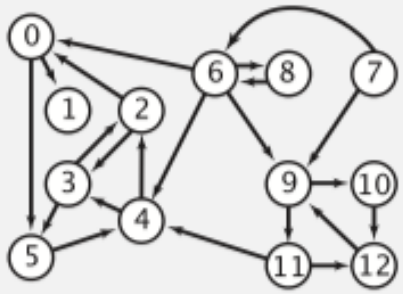
Digraph API

public class Digraph		
	Digraph(int V)	*create an empty digraph with V vertices*
	Digraph(In in)	*create a digraph from input stream*
void	addEdge(int v, int w)	*add a directed edge v→w*
Iterable<Integer>	adj(int v)	*vertices pointing from v*
int	V()	*number of vertices*
int	E()	*number of edges*
Digraph	reverse()	*reverse of this digraph*
String	toString()	*string representation*

Digraph API

tinyDG.txt

```
V
13
22   E
 4  2
 2  3
 3  2
 6  0
 0  1
 2  0
11 12
12  9
 9 10
 9 11
 7  9
10 12
11  4
 4  3
 3  5
 6  8
 8  6
 ⋮
```




```
% java Digraph tinyDG.txt
0->5
0->1
2->0
2->3
3->5
3->2
4->3
4->2
5->4
⋮
11->4
11->12
12->9
```

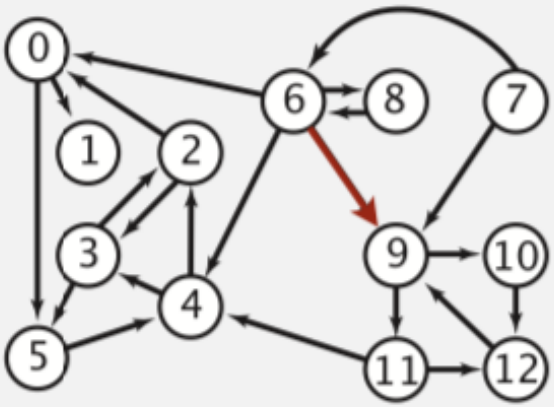
```
In in = new In(args[0]);
Digraph G = new Digraph(in);

for (int v = 0; v < G.V(); v++)
   for (int w : G.adj(v))
      StdOut.println(v + "->" + w);
```

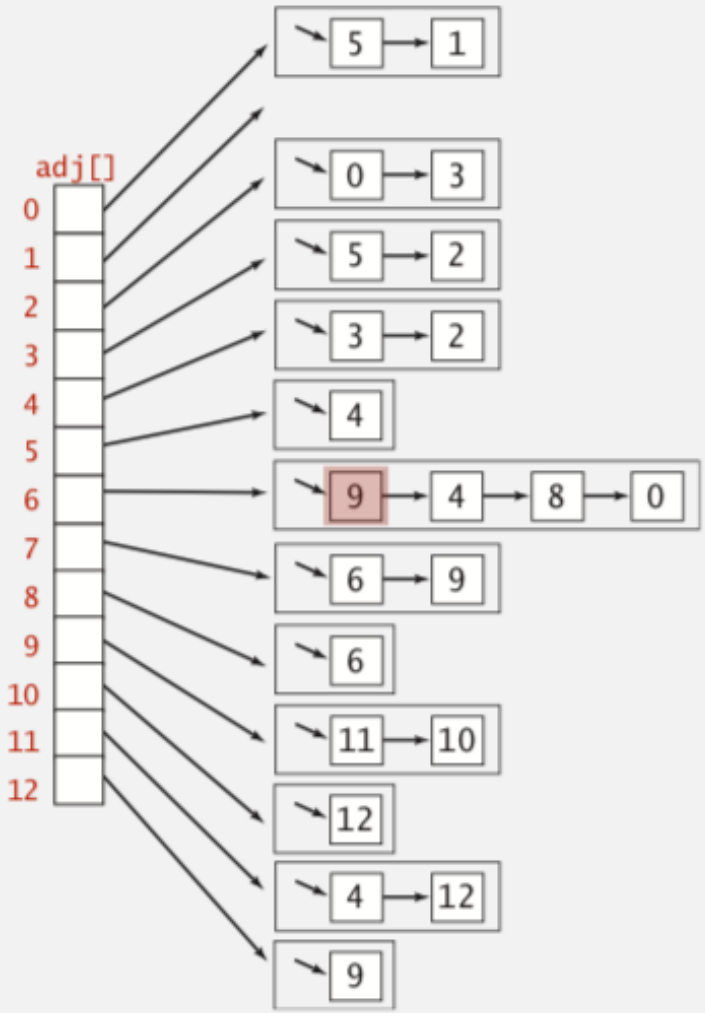
read digraph from input stream

print out each edge (once)

Digraph representation: adjacency lists

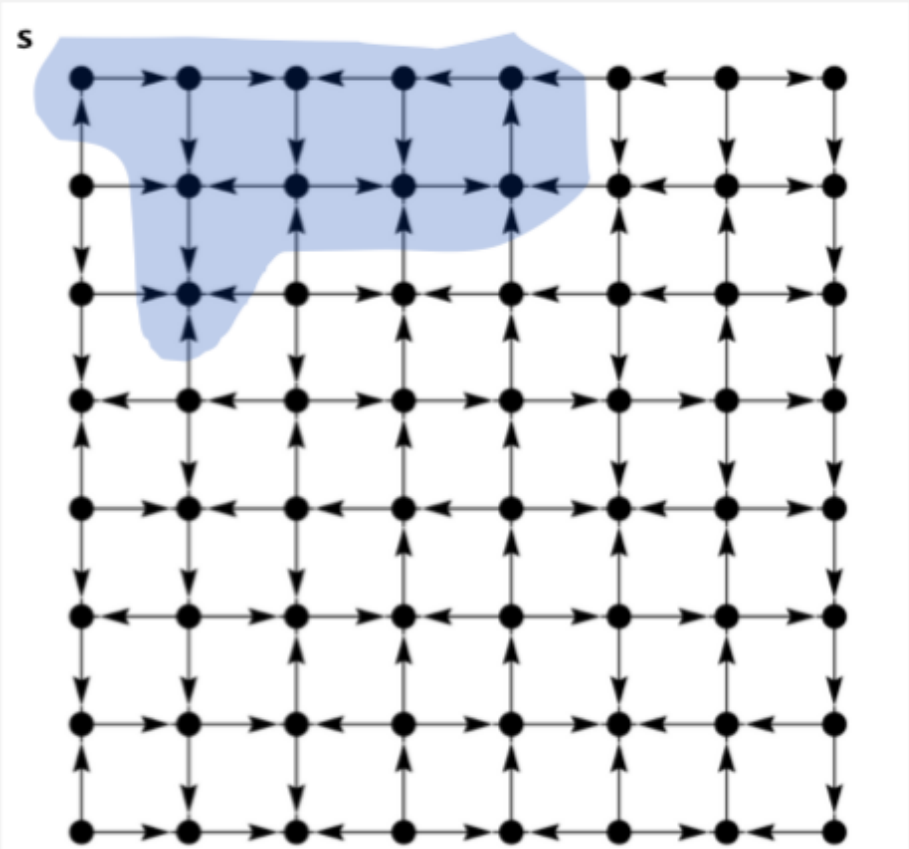
Maintain vertex-indexed array of lists.





Reachability

Problem. Find all vertices reachable from s along a directed path.



Digraph Search: Depth-first search

Same method as for undirected graphs.

- Every undirected graph is a digraph (with edges in both directions).
- DFS is a digraph algorithm.

DFS (to visit a vertex v)

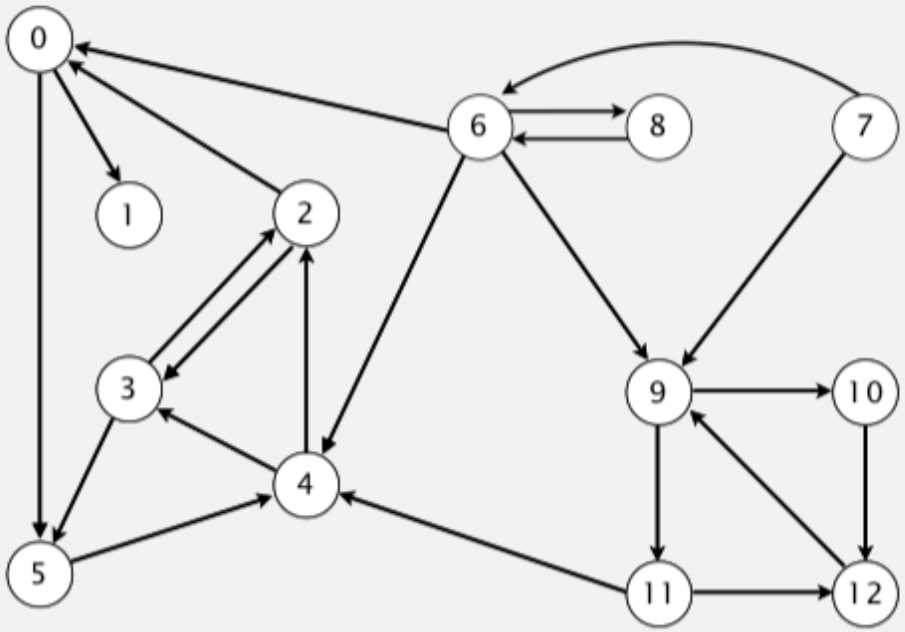
Mark v as visited.

Recursively visit all unmarked vertices w pointing from v.

Digraph Search: Depth-first search

To visit a vertex v :

- Mark vertex v as visited.
- Recursively visit all unmarked vertices pointing from v.



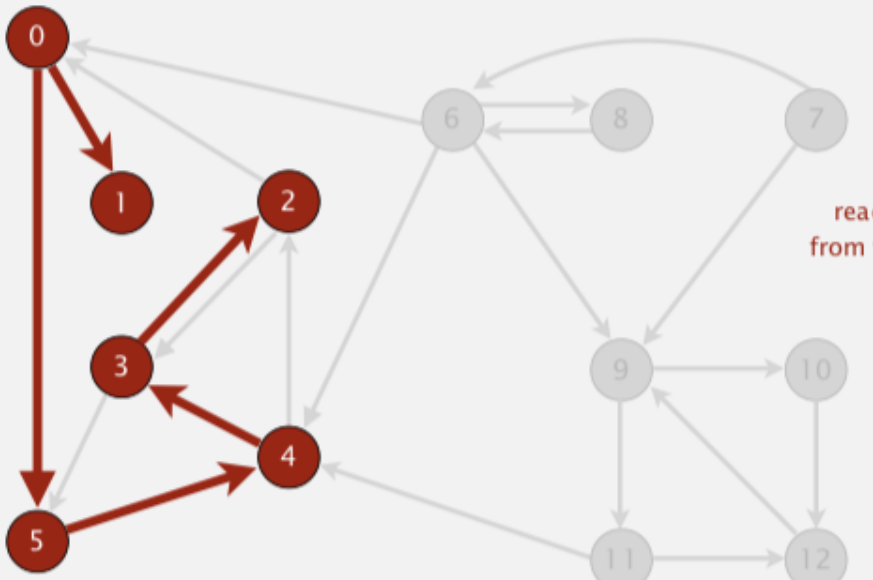
a directed graph

4→2
2→3
3→2
6→0
0→1
2→0
11→12
12→9
9→10
9→11
8→9
10→12
11→4
4→3
3→5
6→8
8→6
5→4
0→5
6→4
6→9
7→6

Digraph Search: Depth-first search

To visit a vertex v:

- Mark vertex v as visited.
- Recursively visit all unmarked vertices pointing from v.



reachable from vertex 0

v	marked[]	edgeTo[]
0	T	–
1	T	0
2	T	3
3	T	4
4	T	5
5	T	0
6	F	–
7	F	–
8	F	–
9	F	–
10	F	–
11	F	–
12	F	–

reachable from 0

Digraph Search: Breadth-first search

Same method as for undirected graphs.

- Every undirected graph is a digraph (with edges in both directions).
- BFS is a digraph algorithm.

BFS (from source vertex s)

Put s onto a FIFO queue, and mark s as visited.

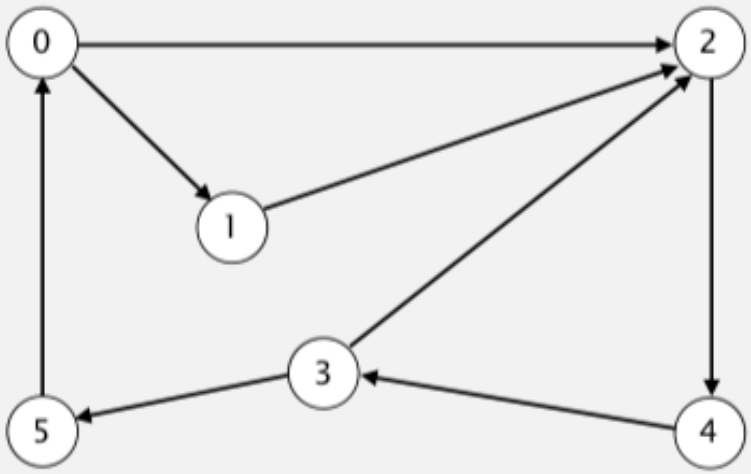
Repeat until the queue is empty:

- **remove the least recently added vertex v**
- **for each unmarked vertex pointing from v:**
  add to queue and mark as visited.

Digraph Search: Breadth-first search

Repeat until queue is empty:

- Remove vertex v from queue.
- Add to queue all unmarked vertices pointing from v and mark them.



graph G

tinyDG2.txt

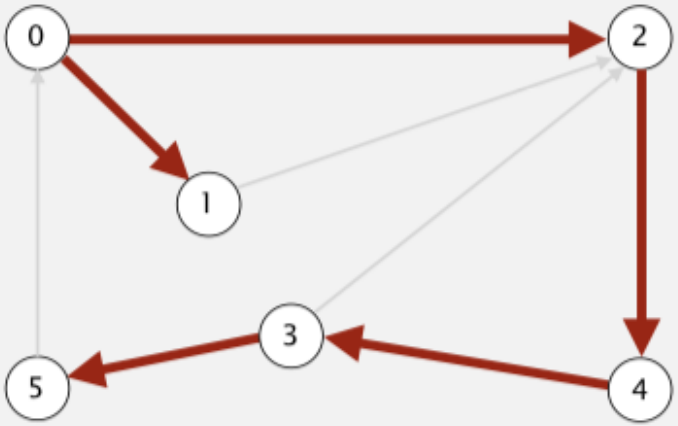
V → 6 ← E

8
5 0
2 4
3 2
1 2
0 1
4 3
3 5
0 2

Digraph Search: Breadth-first search

Repeat until queue is empty:

- Remove vertex v from queue.
- Add to queue all unmarked vertices pointing from v and mark them.



v	edgeTo[]	distTo[]
0	–	0
1	0	1
2	0	1
3	4	3
4	2	2
5	3	4

done