COMP2521 Week 3 Tutorial

Balanced Trees, Graph Basics

Jeremy Le

2026T0

Announcements

- Quiz02 due 12pm Wednesday
- Lab02 handmarking (complexity analysis) due this week!
- Lab03 due 12pm Tuesday Week 4
- Assignment due 8pm Monday Week 5

Balancing Trees Recap

Height

Length of the longest path from the root node to any leaf node.

Motivation

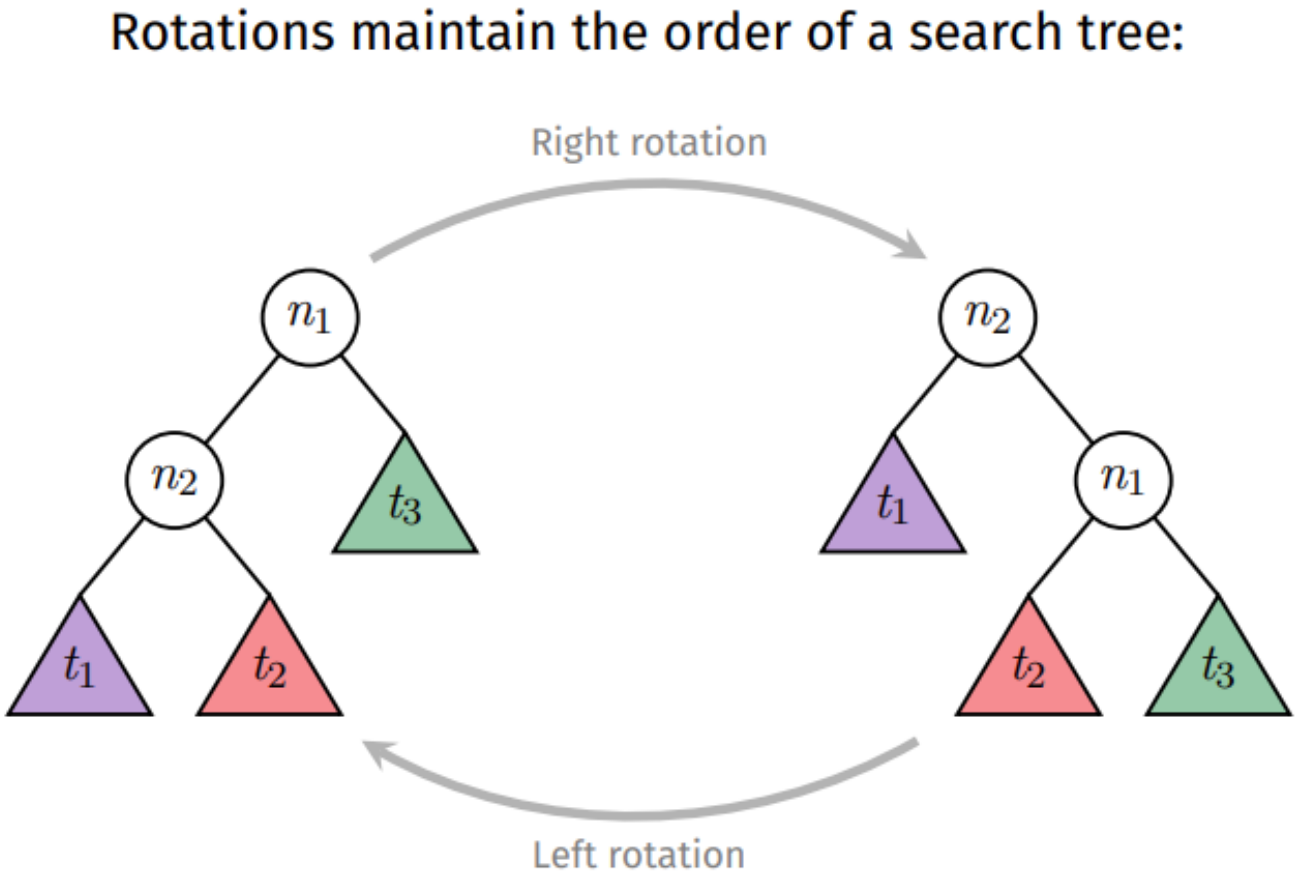
- Balanced trees ensure efficient performance for operations such as search, insertion and deletion.
- Compare the time complexity of insertion on a minimal and maximal height tree on the same number of nodes.

Height Balanced

For every node, the height difference between its left and right subtrees is no more than one. i.e. $|\texttt{height}(l) - \texttt{height}(r)| \leq 1$.

Balancing Operations

There are two types of rotation, left and right rotations and they can be visualised as follows:

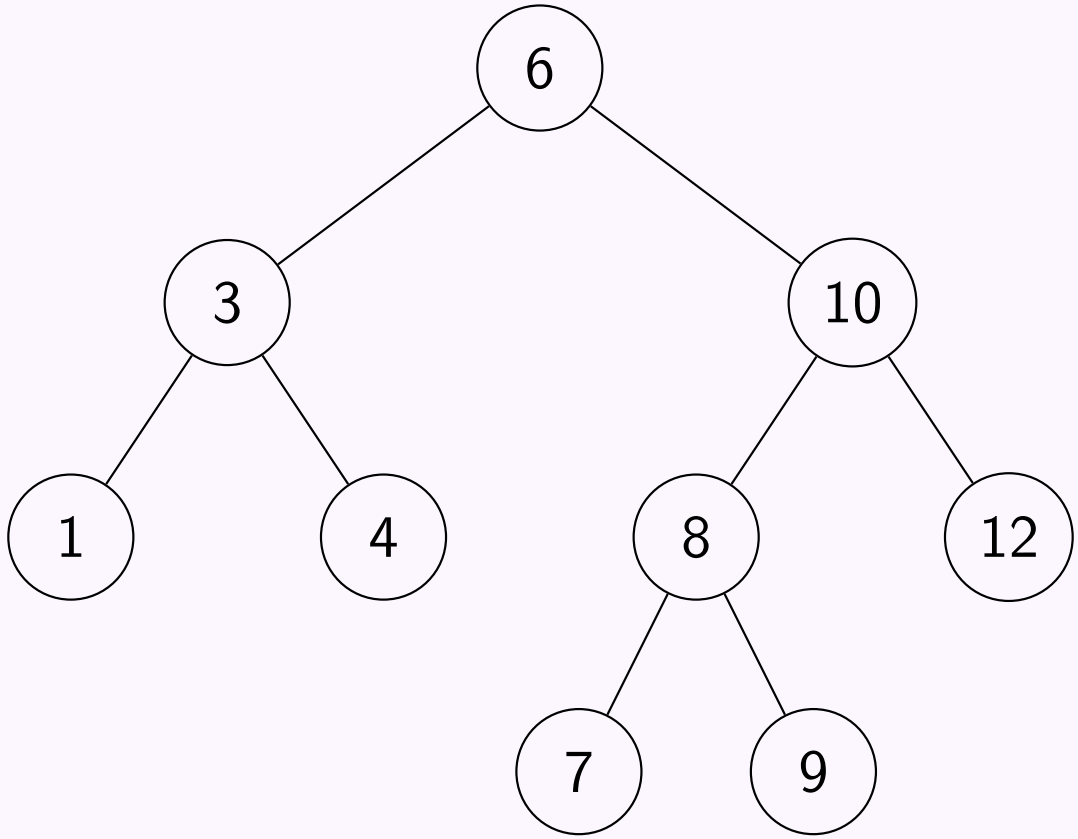


(all values in t_1) $< n_2 <$ (all values in t_2) $< n_1 <$ (all values in t_3)

Balancing Trees — Question 1

Question 1

Show how the following tree would change if we do a right rotation on the node containing 10 followed by a left rotation on the node containing 6.

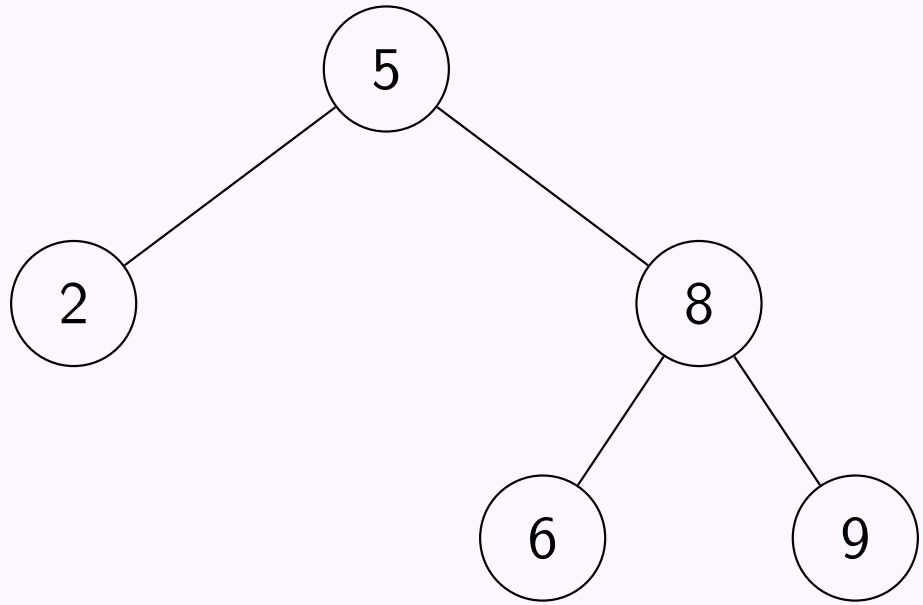


Balancing Trees — Question 1 Working

Balancing Trees — Question 2

Question 2

Show the result of performing the insert-at-root operation with the value 7 in this binary search tree:

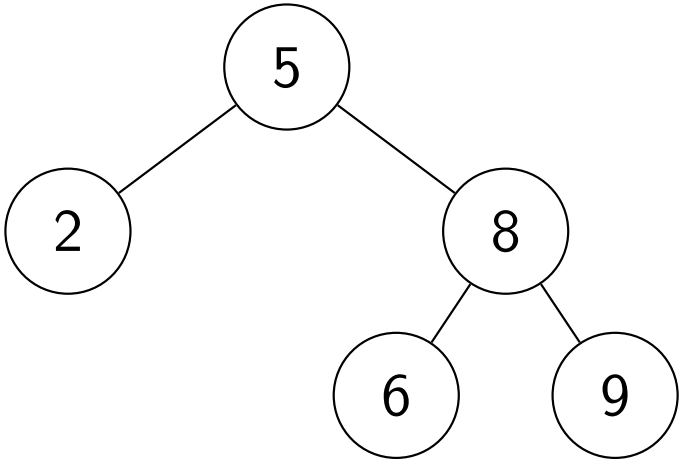


Insert-at-root Operation

Insert-at-root

- Insert into tree as normal
- Perform rotations to lift it to the root
  - If it is the left child of its parent, perform right rotations at its parent
  - If it is the right child of its parent, perform left rotation at its parent
  - Repeat until it is at the root of the tree

Balancing Trees — Question 2 Working



Balancing Trees — Question 2 Working

AVL Trees

AVL Trees

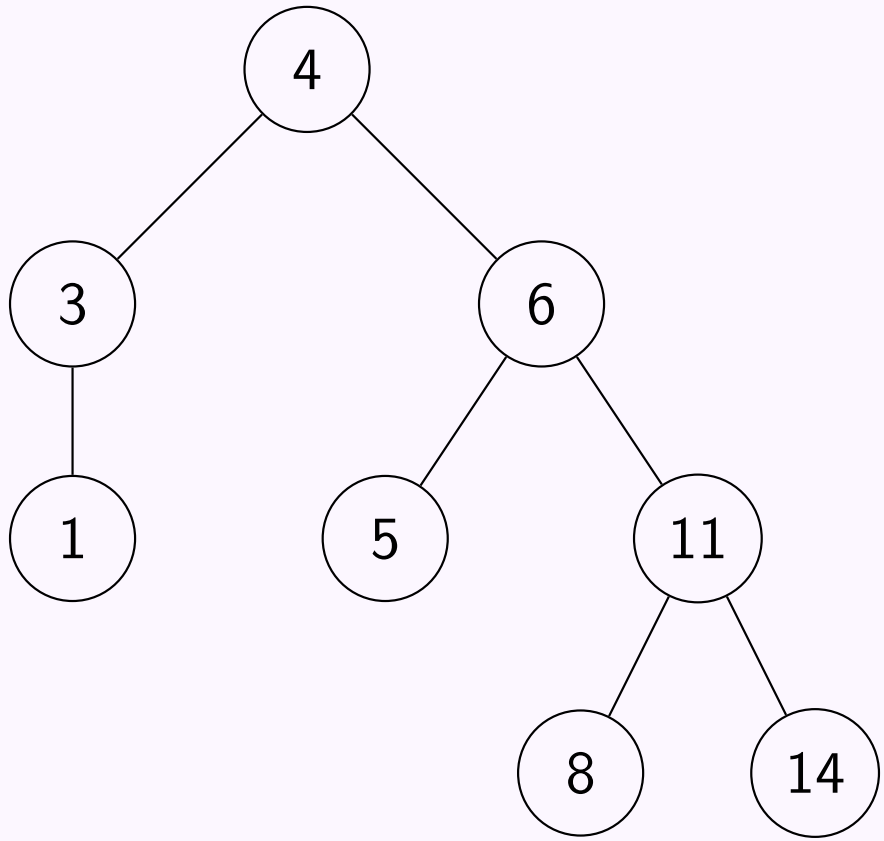
- Self-balancing binary search trees (always balanced)
- Repair any imbalances as soon as they occur (during insertion or deletion)
- Repairs are done locally, not by restructuring entire tree
  - Check balance at each node along the insertion path in reverse
  - Fix imbalances as they are found
- AVL trees use the following node definition:

```
struct node {
    int data;
    int height;
    struct node *left;
    struct node *right;
};
```

AVL Trees — Question 1

Question 1

For the following AVL tree, annotate each node with the height of its subtree.



Now starting from the initial tree each time, insert the following integers: `0 2 10 17`

AVL Trees — Question 1 Working

AVL Trees — Question 2

Question 2

Show how an AVL tree would be constructed if the following values were inserted into an initially empty tree in the order given: `12 10 8 6 4 2`

AVL Trees — Question 2 Working

Graphs

Graphs

- A graph is a data structure consisting of a set of vertices V (nodes) and a set of edges E between pairs of vertices
- Usually denotes $G = (V, E)$ for a graph G
- Edges can either be undirected/directed, unweighted/weighted
- Graphs can have *cycles* or be *acyclic*
- etc... many properties!

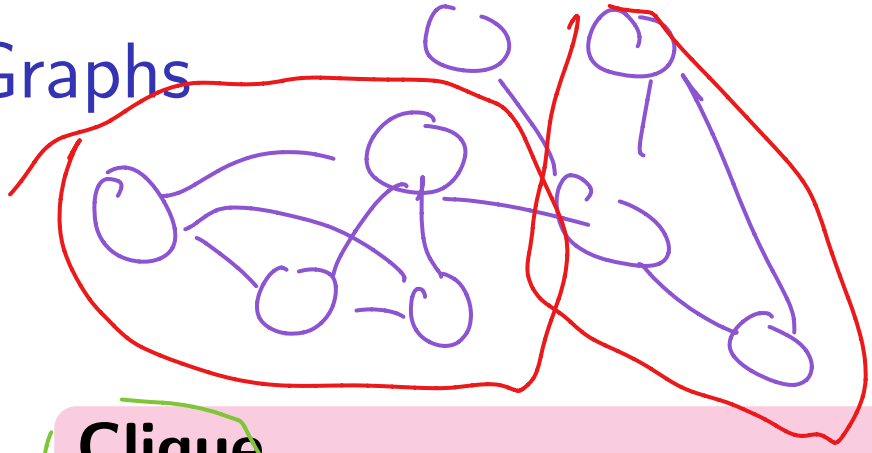
Path

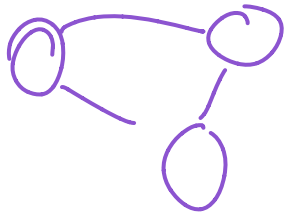
A sequence of edges which joins a sequence of vertices which are all distinct.

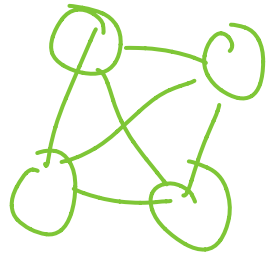
Cycles

A cycle is a path in which only the first and last vertices are equal.

Graphs





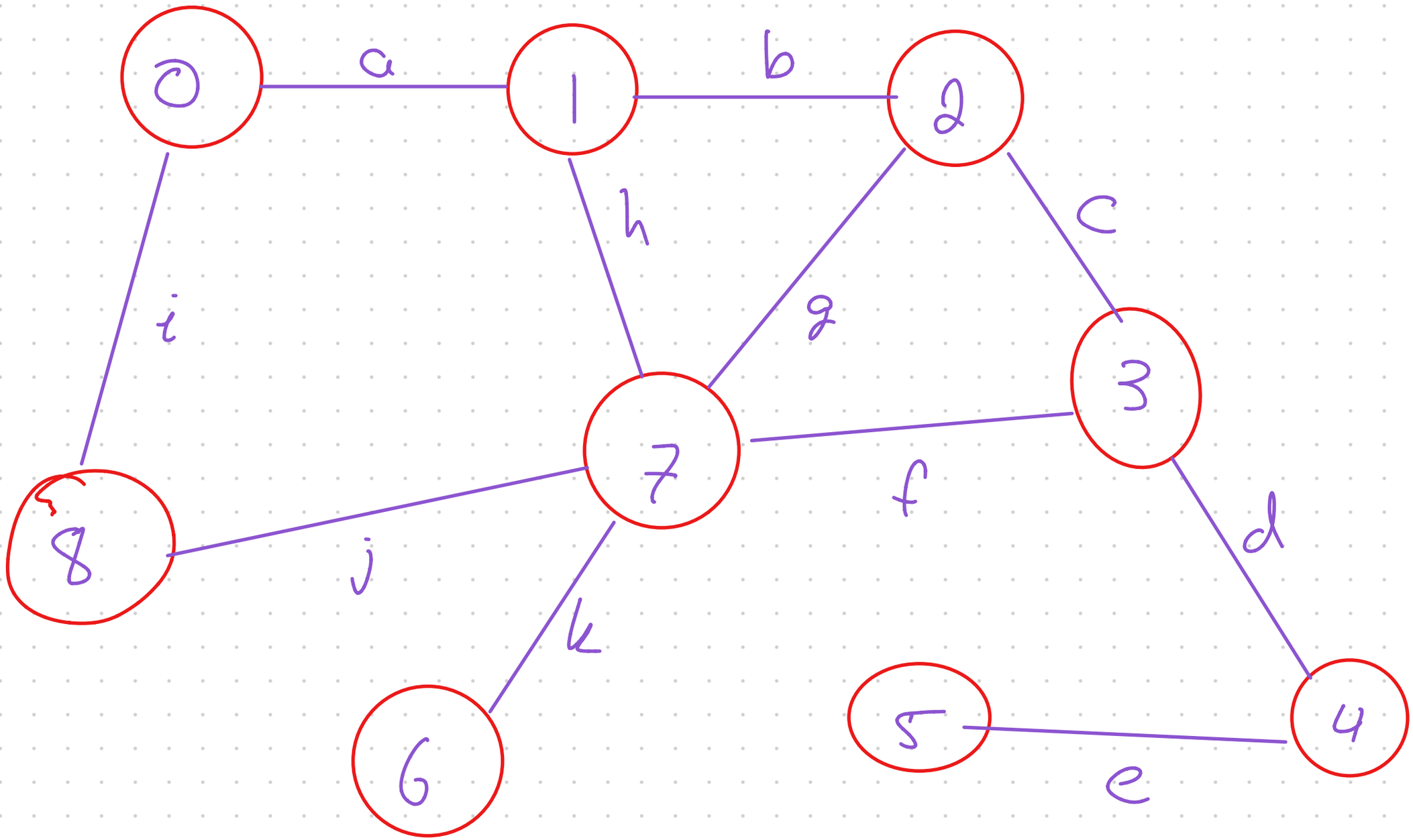


Clique

A complete subgraph. Formally, a subset of vertices of a graph such that every two distinct vertices in the clique are adjacent (share an edge).

Degree

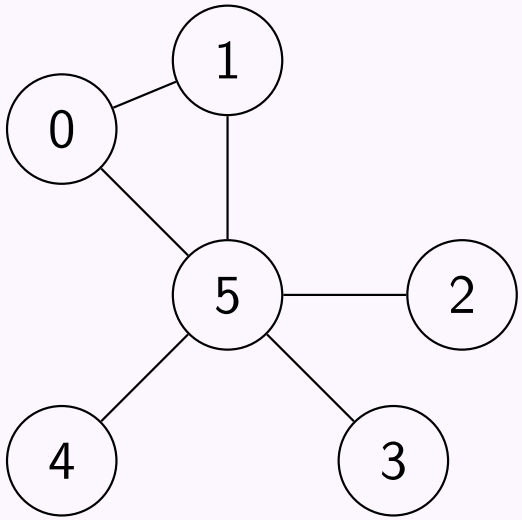
The degree of a vertex is the number of edges that are incident to the vertex.


0
a
1
b
2
h
g
c
i
3
7
f
d
8
j
k
6
5
e
4

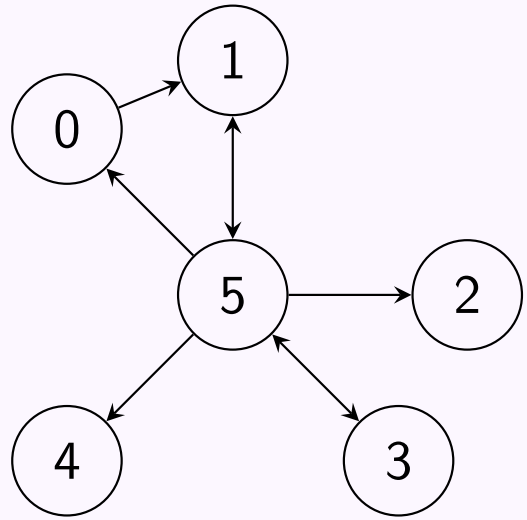
Graphs - Question 2

Question 2

For each of the following graphs:



(i)



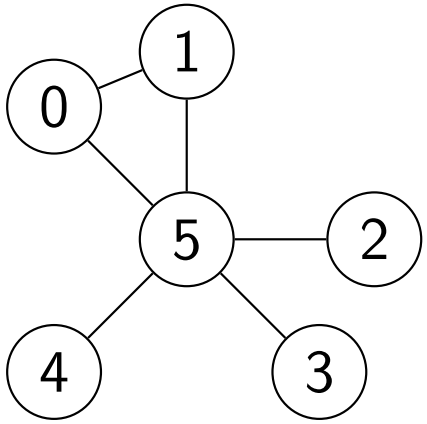
(ii)

Show the concrete data structures if the graph was implemented via:

(a) adjacency matrix representation (assume a full $V \times V$ matrix)

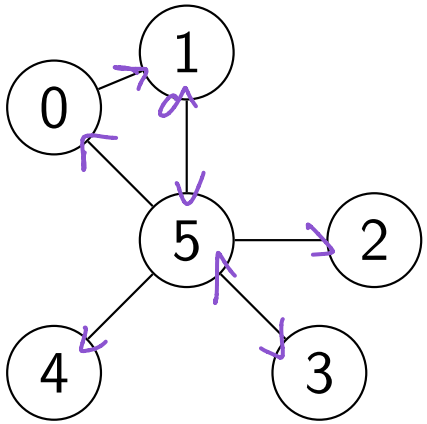
(b) adjacency list representation (if non-directional, include both (v, w) and (w, v))

Graphs - Question 2 Working



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

n^2



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

Graphs - Question 3

Question 3

How is the adjacency matrix for a directed graph different to that for an undirected graph?

Graphs - Question 4

Question 4

Facebook could be considered as a giant "social graph"

(a) What are the vertices? users

(b) What are the edges? friends relationships

(c) Are edges directional? NJ

(d) What does the degree of each vertex represent? num of friends

(e) What kind of graph algorithm could suggest potential friends?

Breadth first search