(a) If $f(n) = O(g(n))$ and $g(n) = O(f(n))$ then $f(n) = g(n)$.

FALSE

Take $f(n) = n^2$ $\quad g(n) = n^2 + 2$

(b) $\log_2 n = \Theta(\log_8 n)$

TRUE

$\log_2 n = 3 \log_8 n$

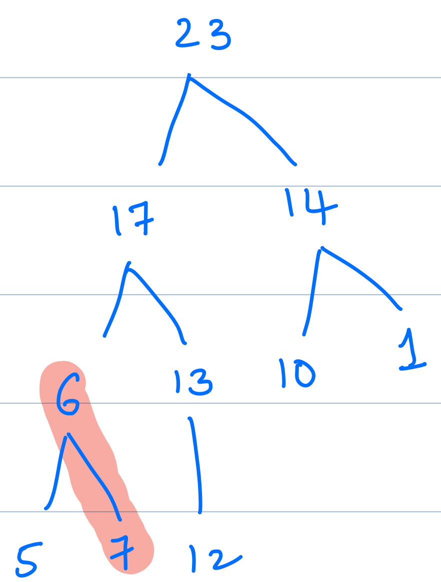
$c_1 \log_8 n \leq \log_2 n \leq c_2 \log_8 n$

Take $c_1 = 1$ and $c_2 = 3$, $n_0 = 1$

ie, $\log_8 n \leq \log_2 n \leq 3 \log_8 n \quad \forall\, n \geq 1$

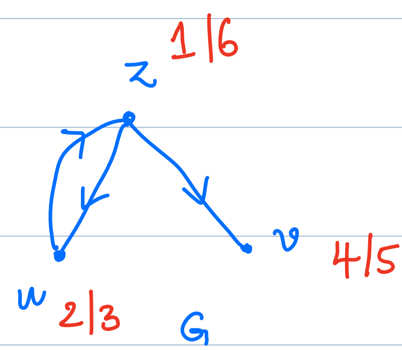
(c) An array constructed with the values $[23, 17, 14, 6, 13, 10, 1, 5, 7, 12]$ is a max heap.

FALSE



(d) If a directed graph G contains a path from u to v, then any depth-first search must result in $v.d \leq u.f$. Here $v.d$ represents discover time of v and $u.f$ represents finish time of u.

FALSE



There is a Path from u to v, But when we do DFS from z, we get $v.d = 4$ and $u.f = 3$

ie, $v.d \geq u.f$.

2. Given an adjacency-list representation of a directed graph, where each vertex maintains an array of its outgoing edges (but not its incoming edges), how long does it take, in the worst case, to compute the in-degree of a given vertex? As usual, we use n and m to denote the number of vertices and edges, respectively, of the given graph. Also, let k denote the maximum in-degree of a vertex. (Recall that the in-degree of a vertex is the number of edges that enter it). Justify your answer. [3]

We must at least read all edges of G.

If an edge e is not read then e may contribute to an indegree of the given vertex.

On the other hand we can compute in-degree of any vertex by reading all the edges

$\therefore$ Running time $= \Theta(m+n)$

Q3. (a)

$$T(n) = 3T(n/4) + n^2 \quad , \quad T(1) = 1$$

Soln using master theorem

$$a = 3, \quad b = 4 \qquad f(n) = n^2$$

$$n^{\log_4 3} = n^{0.79}$$

$$n^2 = \Omega\left(n^{0.79+\varepsilon}\right) \qquad \text{take } \varepsilon = 0.1$$

$$a f(n/b) = 3\left(\frac{n}{4}\right)^2 = \frac{3}{16} n^2 \leq c f(n),$$

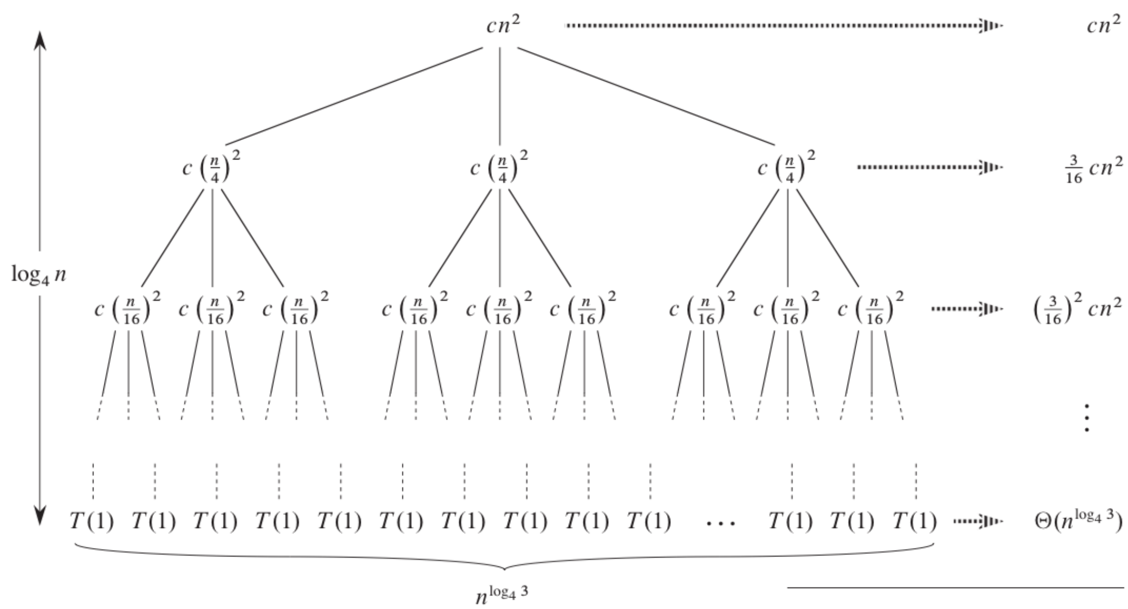
where $c = 3/16$

From Case 3 of Master theorem

$$T(n) = \Theta(f(n)) = \Theta(n^2)$$

Q3 (b) $T(n) = 3T(n/4) + cn^2$

Final recursion tree looks as follows



(d)

Total: $O(n^2)$

$$\text{Total cost} = cn^2 + \frac{3}{16}cn^2 + \dots + \left(\frac{3}{16}\right)^{\log_4^n - 1} + \Theta(n^{\log_4^3})$$

$$= \sum_{i=0}^{\log_4^n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4^3})$$

$$< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4^3})$$

$$= \frac{1}{1-\frac{3}{16}} cn^2 + \Theta(n^{\log_4^3})$$

$$= \frac{16}{13} cn^2 + \Theta(n^{\log_4^3})$$

$$= O(n^2)$$

4. Given an array $A[1, \cdots, n]$, we say a pair $(A[i], A[j])$ is an inversion if $i < j$ but $A[i] > A[j]$. Design an $O(n \log n)$ algorithm to count the number of inversion pairs. For example the array $[2, 4, 1, 3, 9]$ has three inversions $(2, 1), (4, 1), (4, 3)$. Clearly write all the details of your algorithm. [6]

This Problem was discussed in the Class.

(Small modification to merge sort gives the soln to the above problem)

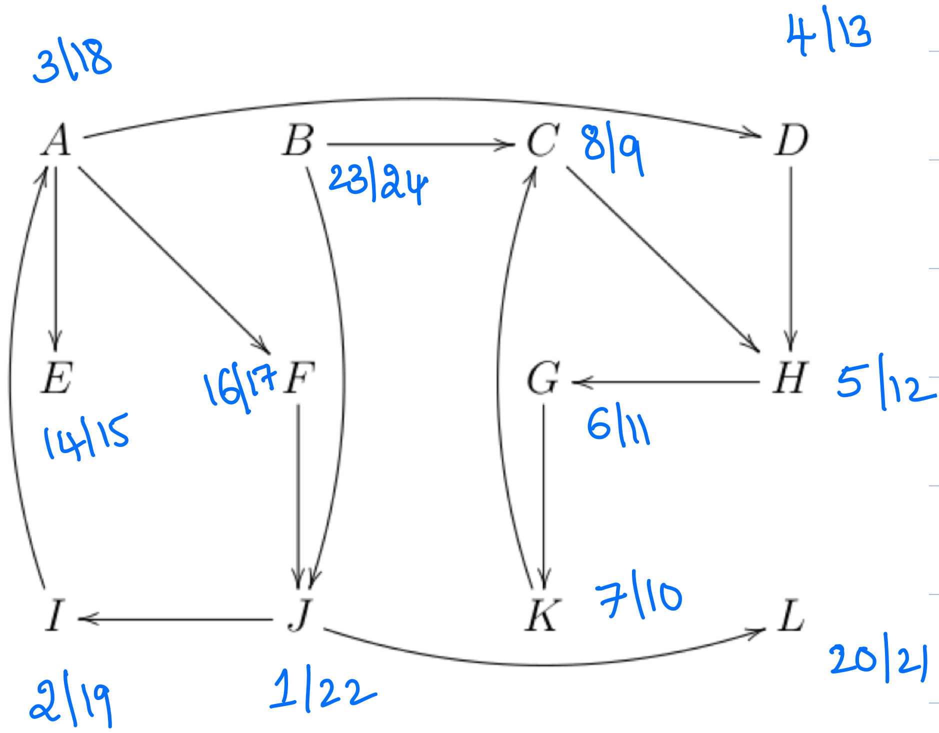
Q5 The Problem is known as

" Searching in Rotated Sorted array"

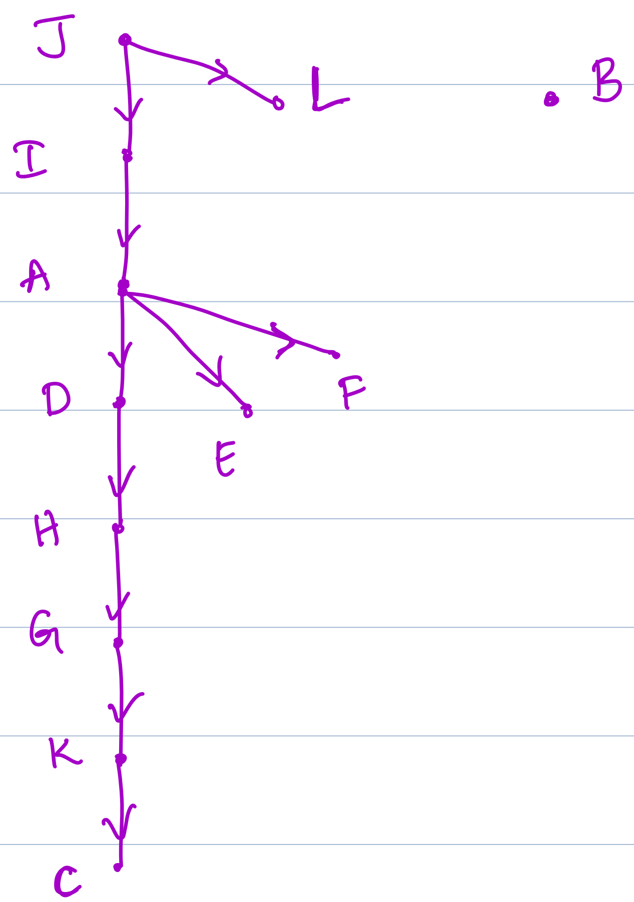
Solution Can be obtained by using binary Search

Runningtime: $O(\log_2 n)$

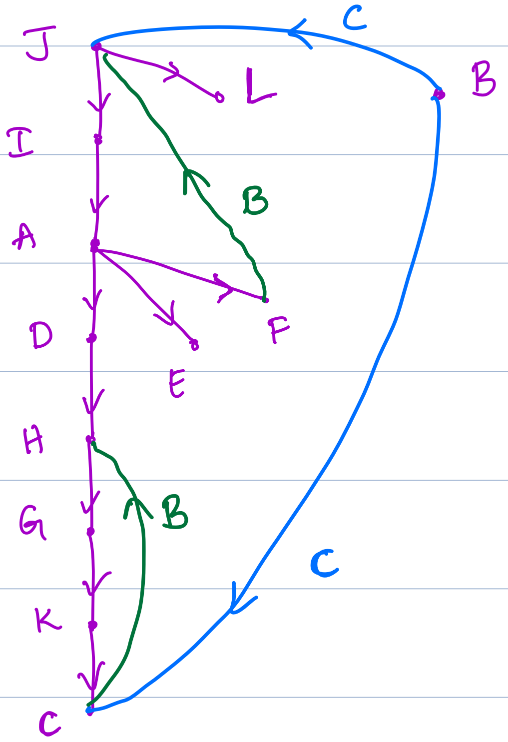
Q6


A
B
C
D
E
F
G
H
I
J
K
L
3|18
4|13
8|9
23|24
16|17
14|15
6|11
5|12
7|10
20|21
2|19
1|22

DFS Forest



Classification of edges



Unmarked edges are tree edges