2020

1(a)

Algorithm: A finite set of Intruction / Pseudocode to solve a specific problem.

To calculate the efficiency of algorithm, order is more important than the speed of processor.

Lets assume.

~~an algorithm computes 50 nlogn instructions.~~

a problem is given. X algorith solves it in $O(n \log n)$ time & Y algorithm solves it in $O(n^2)$ time.

Let's say, $n = 10^6$

For slow processor, speed is 10^6 instructions /second

" Fast " , " " 10^8 " /~~second~~ "

For ~~problem~~ Algorithm X,

slow processor solves it in = $\frac{n \log n}{10^6}$

$= \frac{10^6 \log(10^6)}{10^6}$

$= 19.93$

≈ 20

For algorithm Y,

fast processor takes it $\frac{10^9}{10^7}$

$= 10^2$ s

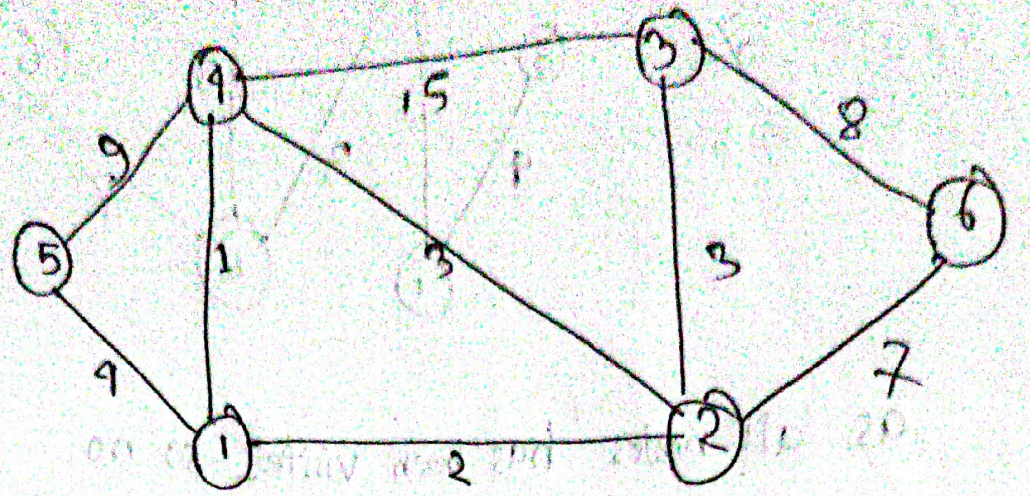
So the problem is being solved in the slow processor in less time than its fast processor even after huge difference in executing instruction speed.

So processor speed is less important than order.

(b)

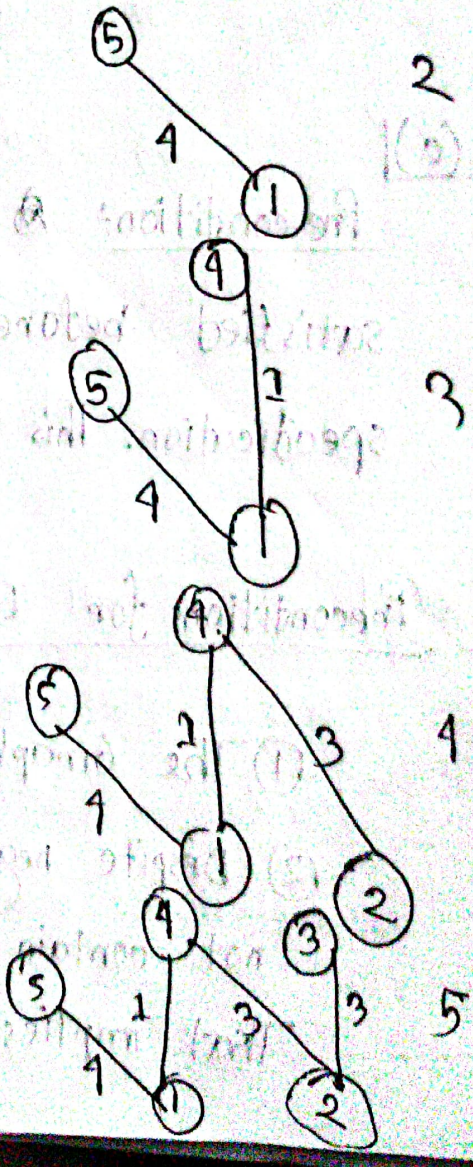
Spanning tree of a graph is a subgraph that contains all the vertices with minimum number of edges and is a tree.

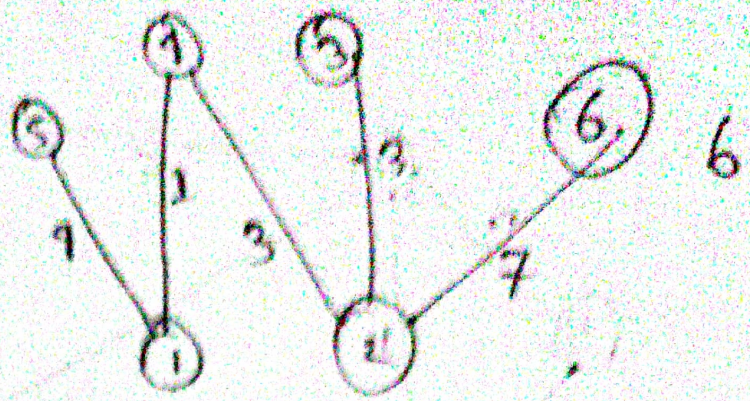
The spanning tree with the minimum weight is called minimum spanning tree.



Applying prim taking (5) as source.

Edges	cost	MST	no. of nodes in MST
5,1	4		2
1,4	1		3
4,2	3		4
2,3	3		5





as all nodes has been visited so no more edges to add.

total cost = 9+1+3+3+7

= 18

1(c)

Pre-condition: conditions that must be satisfied before an operation in a formal specification. This is like explicit constraint.

Preconditions for Bellman-ford Algorithm:

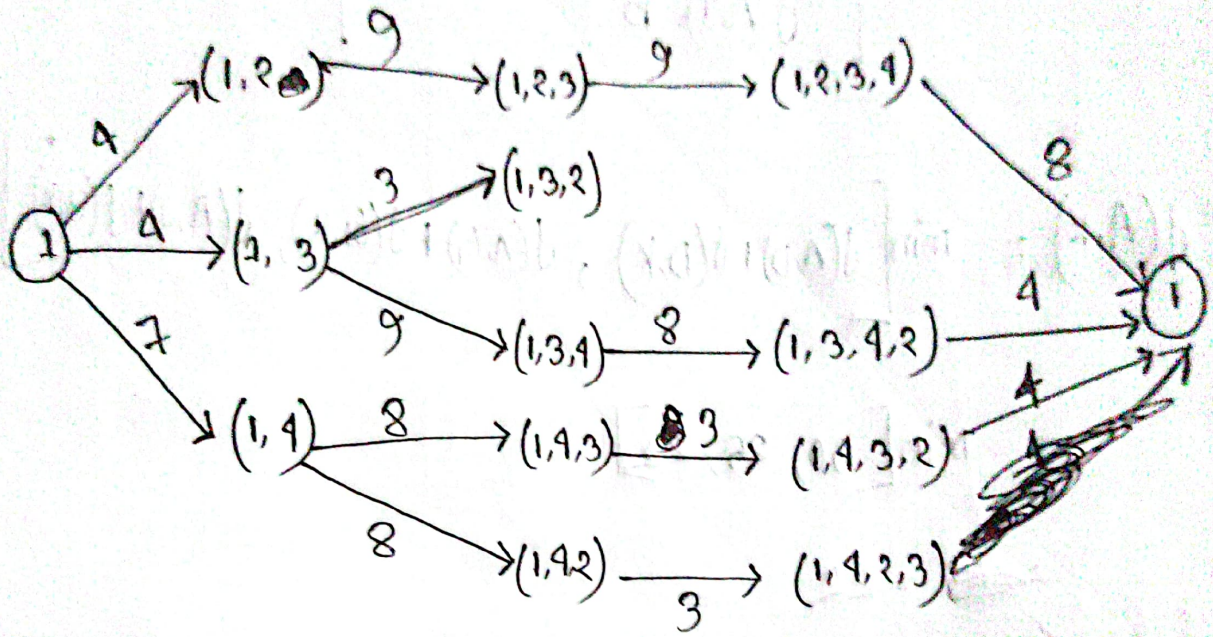
① The Graph must be weighted & directed.

② Despite negative edges, the graph must not contain any negative cycle.

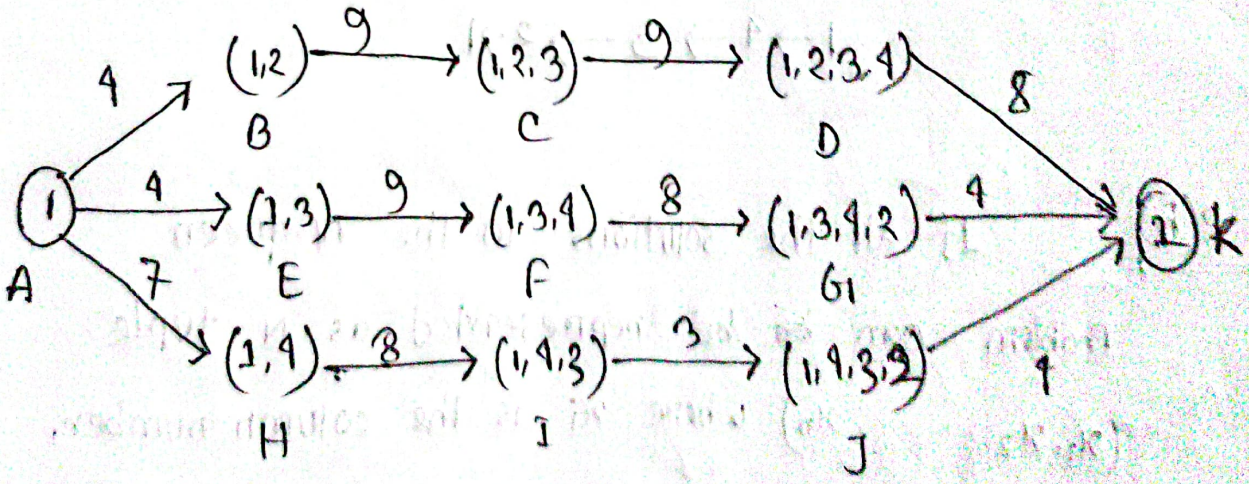
That implies, The shortest path contains a

finite number of edges.

2(a) Here the source and sink in vertex 1



So the Multi Stage Graph is:-



(ii)

$d(A,B) = 7$ $d(A,E) = 4$ $d(A,H) = 3$	$d(A,C) = d(A,B) + d(B,C)$ $= 13$ $d(A,I) = 7+3 = 13$ $d(A,J) = 15$	$d(A,D) = 22$ $d(A,G) = 21$ $d(A,J) = 18$

$$d(A,K) = \min\left[d(A,D)+d(D,K),\ d(A,G)+d(G,K),\ d(A,J)+d(J,K) \right]$$

$$= \min[30, 25, 25]$$

~~25~~ = 22

Path : $A \to H \to I \to J \to K$

$1 \to 4 \to 3 \to 2 \to 1$

2(b)

If all the solutions to the N-queen problem can be represented as N-tuple $(x_1, x_2, \ldots, x_n)$ where x_i is the column number, The explicit constraint is that: $1 \le i \le n$

The implicit constraints:

① No two x_i can be the same.

② Two queens can not be on the same diagonal.

Recursive Agorithm for N-Queen:

```
PLACE(k,i)
{ for j:=1 to k-1 do
    if ((x[j] = i)  → two in the same column
      or (Abs(x[j]-i) = Abs(j-k))   → in the same diagonal
      then return false;
  return true;
}

Procedure NQUEEN(k,n)
{ for i:=1 to n do
  { if PLACE(k,i) then
    { x[k]:= i;
      if (k=n) then write(x[1:n]);
      else NQUEEN(k+1,n);
    }
  }
}
```

Here,

- k is the row no. where Q to be placed
- i is the column number

3(a)

i) $\log(n) + 1000$

$f(x) \leq \log(n) + 1000 \log n$

$\Rightarrow f(x) \leq 1000 \log n$

$\Rightarrow f(x) \leq c \cdot g(x)$

$\therefore O(g(x)) = O(\log n)$

ii) $2^{10} + 35$

$f(x) \leq 2^{10} + 2^{10}$

$\Rightarrow f(x) \leq 2 \cdot 2^{10}$

$\Rightarrow f(x) \leq$ constant

$\therefore O(f(x)) = O(1)$

iii) $n\log n + 15n + 0.002n^2$

$f(x) \leq n^2 + 15n^2 + 0.002n^2$ $[n^2 > n\log n]$

$\Rightarrow f(x) \leq c \cdot n^2$

$\Rightarrow f(x) \leq c \cdot g(x)$

$\therefore O(g(x)) = O(n^2)$

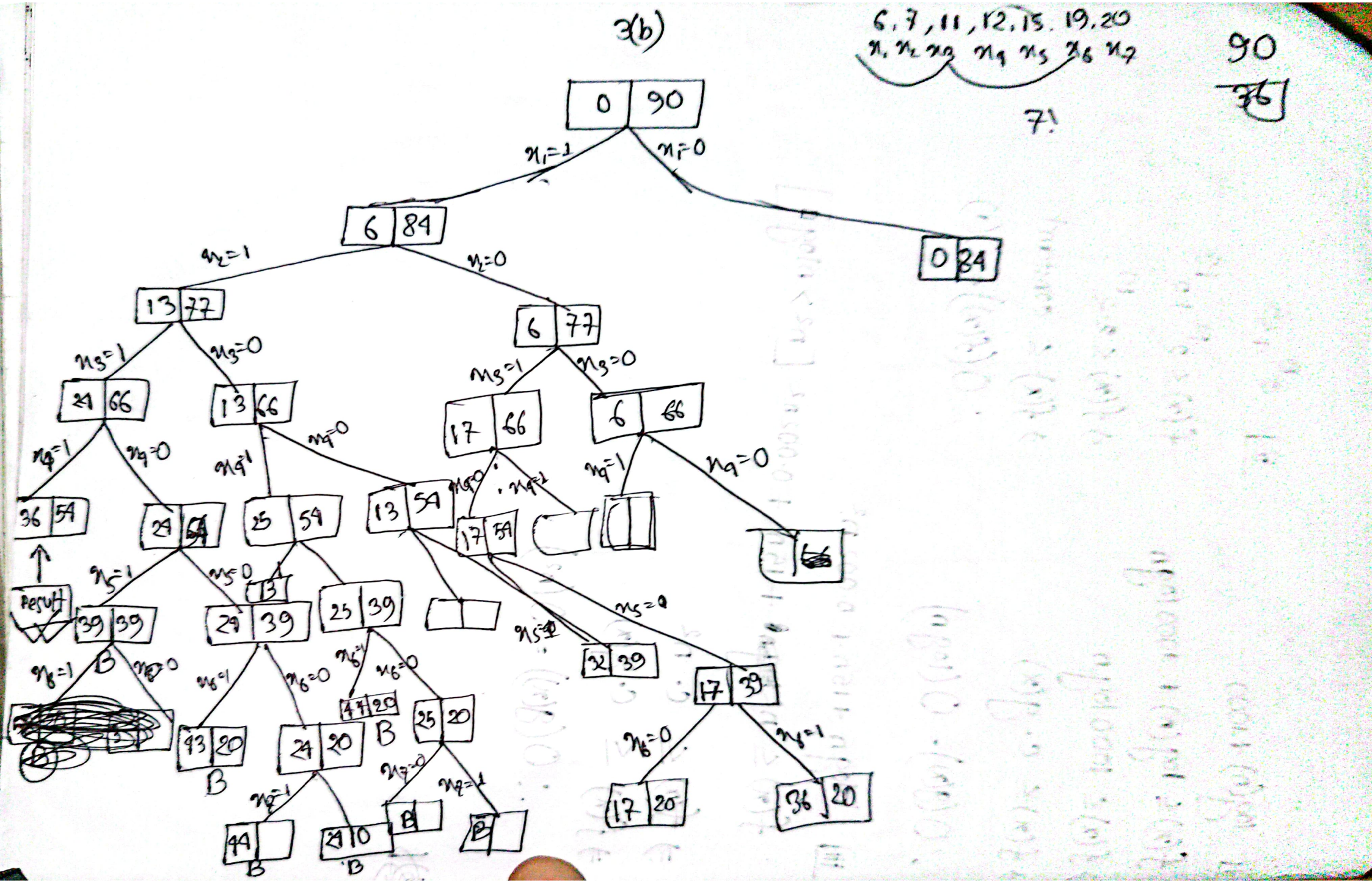
3(b)

3(b)

6, 7, 11, 12, 15, 19, 20
$x_1, x_2, x_3, x_4, x_5, x_6, x_7$

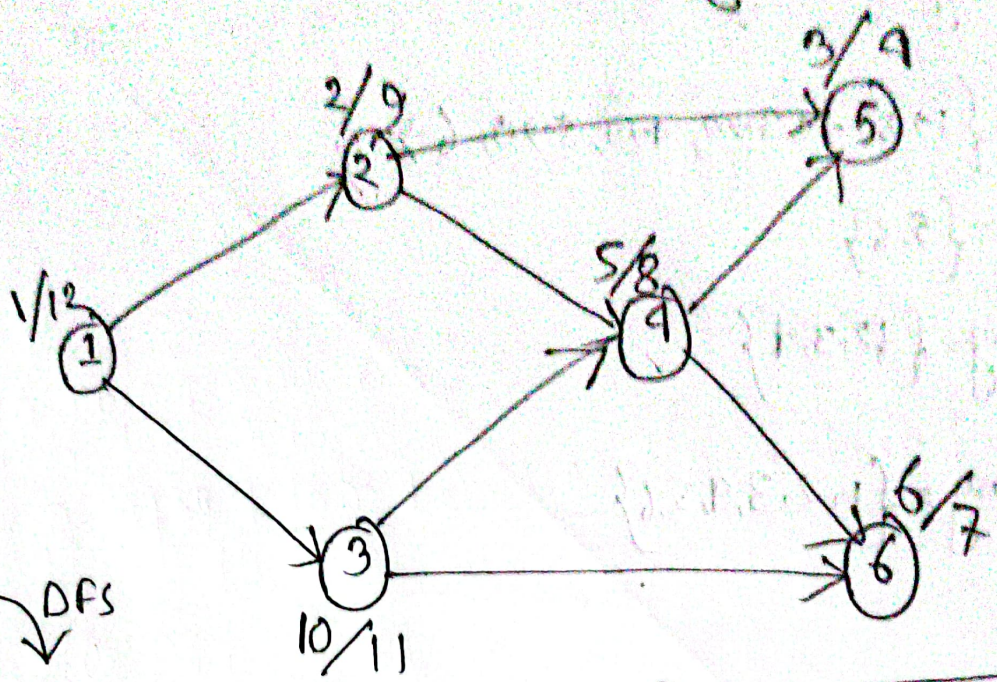
90

71



3(b)

Topological Sort: is a linear ordering of all its vertices such that if G contains an edge(u,v) then u appears before v in the ordering.



By other way → DFS

map = {1→0, 2→1, 3→1, 4→2, 5→2, 6→2}

queue = {1}

ordering = { }

Finishing time:

1→3→2→4→6→5

map = {1→0, 2→0, 3→0, 4→2, 5→2, 6→2}

queue = {2, 3}

ordering {1}

map = {1→0, 2→0, 3→0, 4→2, 5→2, 6→2}

queue = {3}

ordering {1,2}

map: {1→0, 2→0, 3→0, 4→0, 5→1, 6→2}

queue = {4}

Ordering = {1, 2, 3}

map = {1→0, 2→0, 3→0, 4→0, 5→0, 6→2}

queue = {5, 6}

ordering = {1, 2, 3, 4}

ordering = {1, 2, 3, 4, 5, 6}

5(b)

Lower Bound: An estimate of a number of operation need to solve an given problem.

Finding the min and max:

Given a sequence

$X = (x_1, x_2, \ldots, x_n)$ of n distinct numbers,

find indices i and j such that $x_i = \min(X)$ and $x_j = \max(X)$

First we find the minimum in n-1 comparisons and then find the maximum of every thing else in n-2 comparisons. So upper bound of $(n-1+n-2) = 2n-3$

As any algorithm that finds the min & the max certainly finds the max, we found lower bound of $(n-1)$.

We can improve both the upper and the lower bound to $\lceil 3n/2 \rceil - 2$

The upper bound is established by the following algorithm-

* Compare all $\lceil n/2 \rceil$ consecutive pairs of elements x_{2i-1} and x_{2i}
* Put the smaller element into a set S.
* Put the larger element into a set L and if n is odd, put x_n into both L and S.
* Put the larger