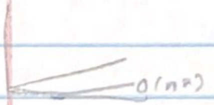
algorithm → ↑order

i ← 1 to 5

j ← 1 to i



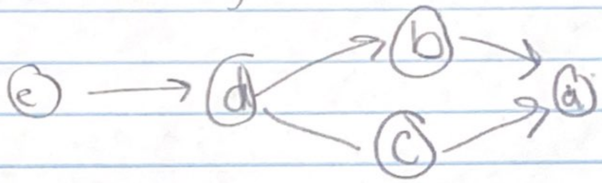
max heap

1 deletes the root

2 reorders the remaing

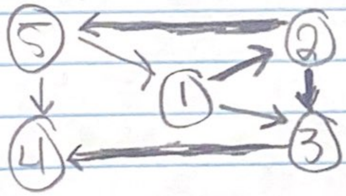
3

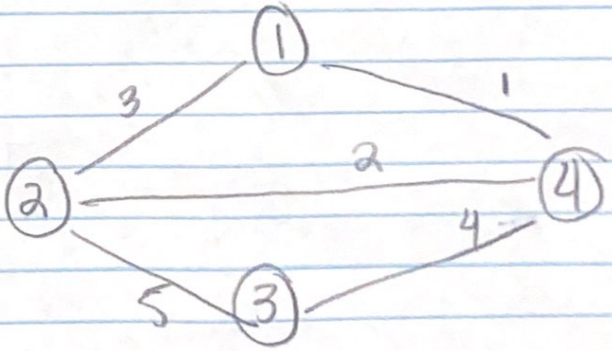
1 2 3 4 5



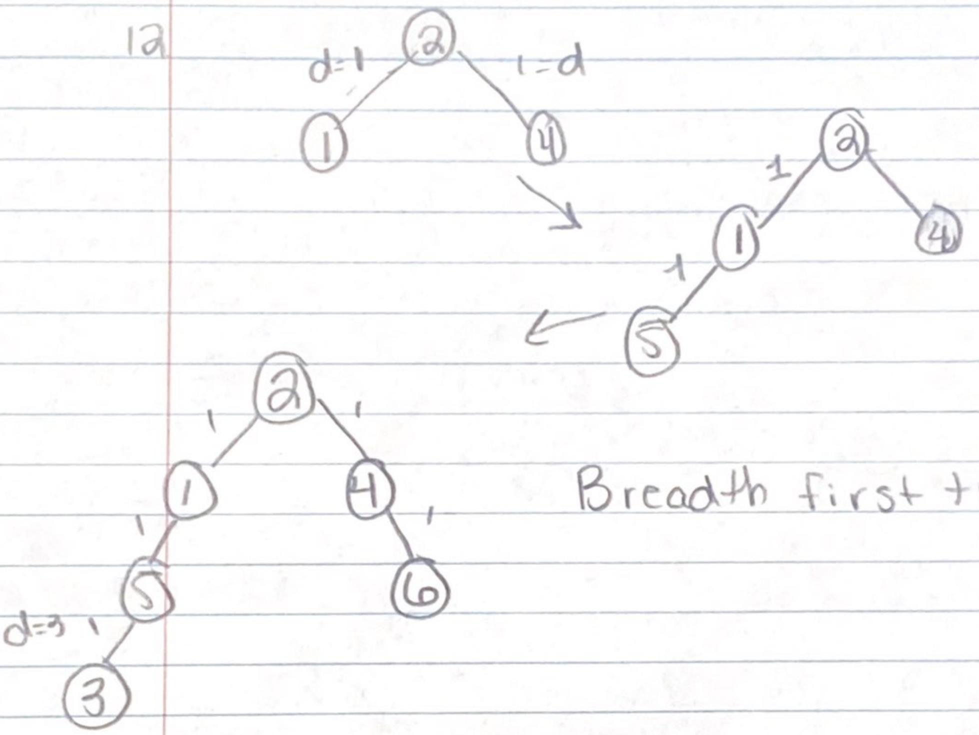
Quick sort

partition;



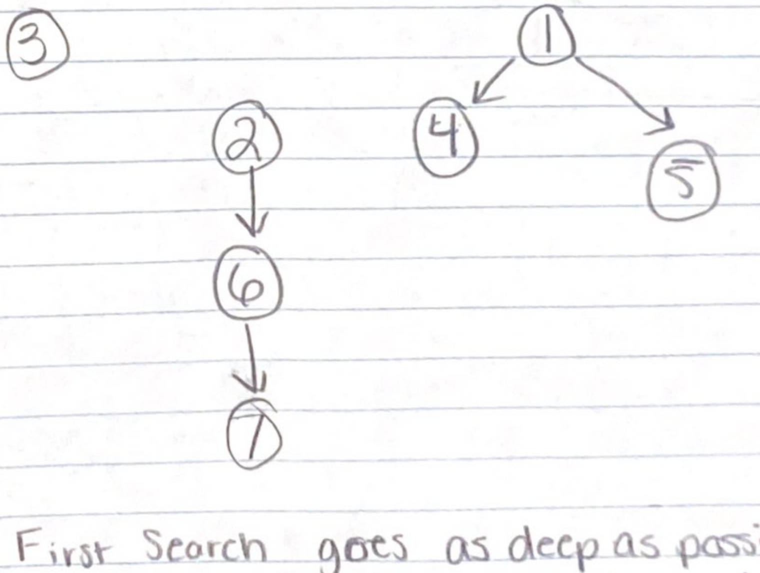


12



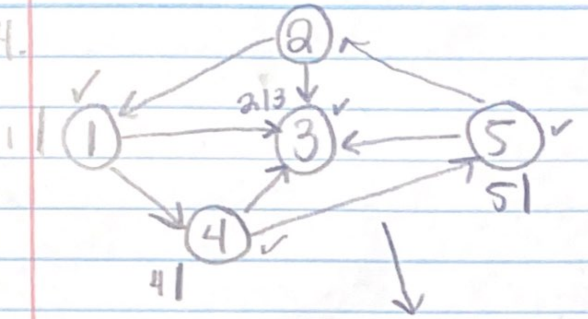
Breadth first tree

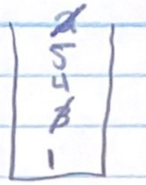
13.

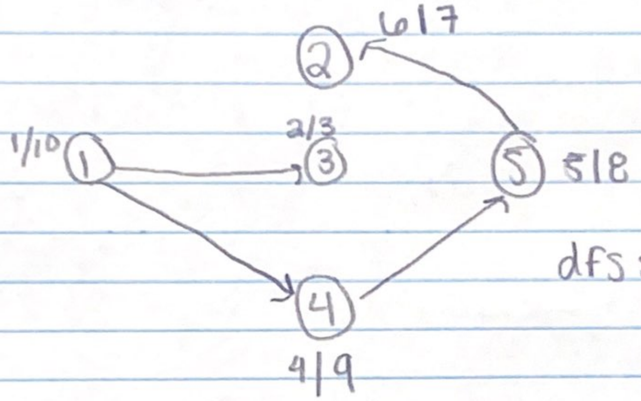


Depth First Search goes as deep as possible You then back track if you can't go deeper. If you get back to the starting vertex with vertices still to be visited. It will be a forest. Shift to the unvisited vertex and continue until all are visited.

14.







(final tree)

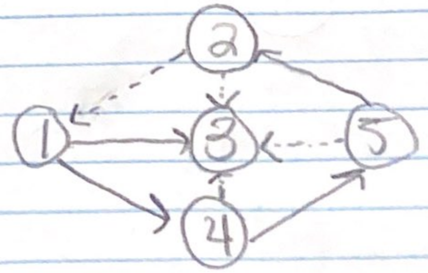
dfs: 1 3 4 5 2

Tree edge: obtained by applying DFS on the graph

Forward edge: It is an edge (u,v): V is descendat but isn't part of DFS tree.

Backword edge: It is an edge (u, v): V is ancestors of node but not part of the tree.

Cross edge: It is an edge which connects two nodes so they do not have any ancestors and descendat bond between them

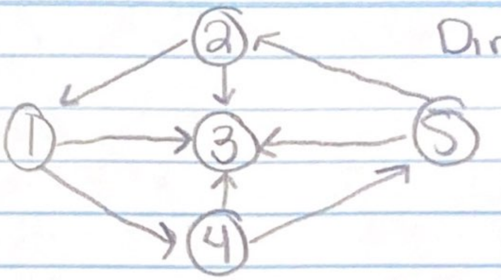


Tree edges: 13, 14, 45, 52

Forward edge: 21, 2, 3

Backword E: 43

Cross edge: 53



Directed Acyclic Graph

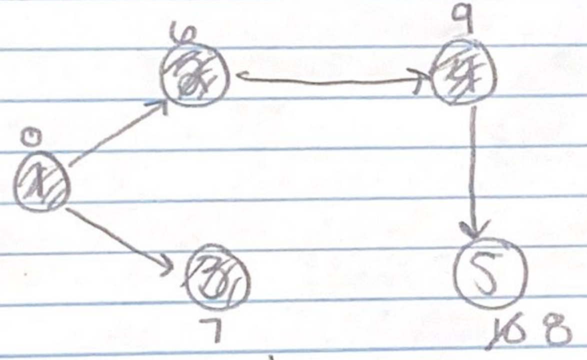
D.A.G : a digraph it should contain any cycle

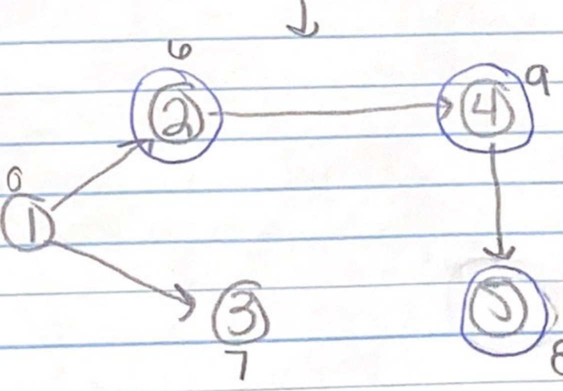
there is a cycle in this graph

1 → 4 → 5 → 2 → 1

which means that it isn't D.A.G, so there is no application of topological sort.

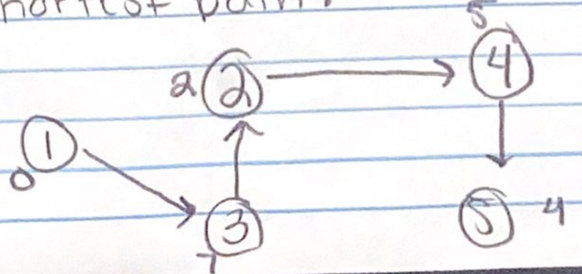
15.





incorrect values of nodes 2, 4, 5

Shortest path:



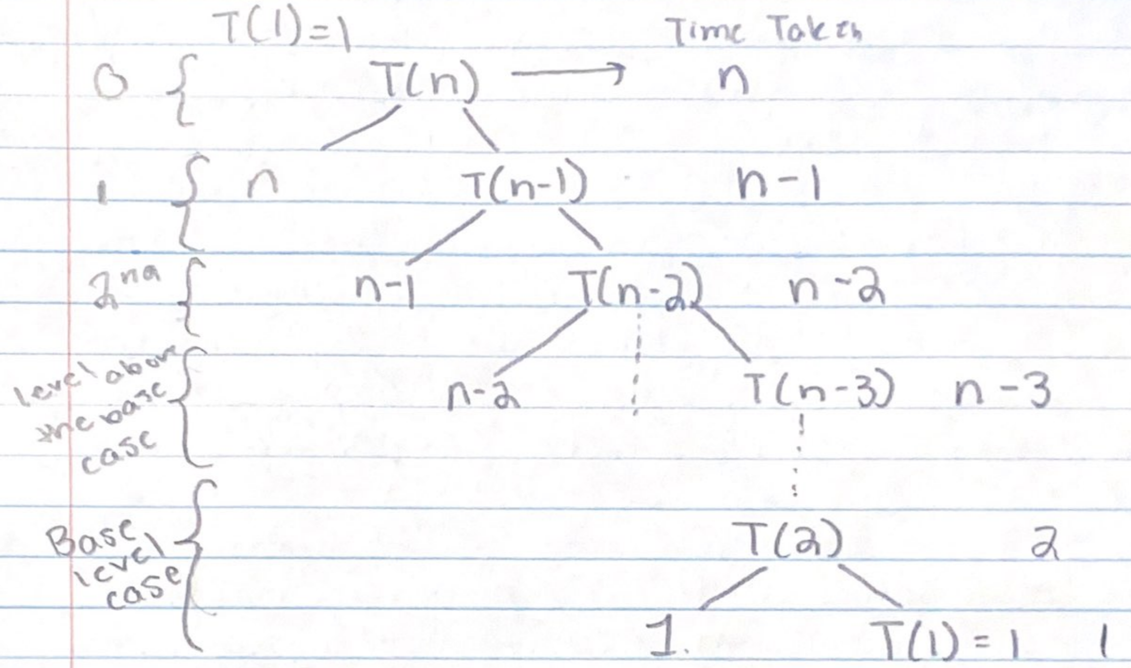
16 (a). Yes, the problem of finding shortest path in a graph is in P due to there being a polynomial time algorithm which is used to solve it.

(b.) For problems in class NP that are not inside P, there is an infinite hierarchy of problems in NP in between the ones in P and the ones that are NP complete. This characteristic is from the Ladner's Theorem.

(c.) The NP in the phrase "NP-complete" stands non-deterministic polynomial time. This is a problem that is solved in Polynomial time with the non-deterministic turning machine NP-complete means it is NP and any problem has a "reduction" from y to X; which is a polynomial time algorithm that turns any instance y in to the intanse of X.

17. $T(n) = T(n-1) + n$

$T(1) = 1$



Total Time :

$T(n) = (n-1) + (n-2) + \text{- - -} + 2 + 1$

$= 1 + 2 + \text{- - - -} + n$

$= \frac{n(n+1)}{2}$ → Required polynomial

$O(n^2)$ Time

18. $T(n) = T(n-1) + n$; $T(1) = 1$

Base case Proof:

$$T(1) = T(1-1) + 1$$
$$= T(0) + 1$$
$$= 0 + 1$$
$$= 1$$
$$T(1) = 1$$

Inductive Step:

$T(n) = T(n-1) + n$ first

$T(n-1) = T(n-2) + (n-1)$ Second

$T(n-2 = T(n-3) + (n-2)$ third

Substitute Second into First

$T(n) = T(n-2) + (n-1) + n$ fourth

Substitute Third into fourth

$$T(n) = T(n-3) + (n-2) + (n-1) + n$$
$$T(n) = T(n-4) + (n-3) + (n-2) + (n-1) + n$$

$$\vdots$$

$$T(n) = T(n-k) + (n-(k-1)) + (n-(k-2)) + \cdots$$
$$(n-3) + (n-2) + (n-1)$$
$$= 5$$

Base condition

$$n - k = 1 \rightarrow n = k + 1 \rightarrow k = n - 1$$

Substitute k

$$T(n) = T(n-(n-1) + (n-(n-1)-1) + (n-((n-1)-2)) + \cdots$$
$$+ (n-3) + (n-2) + (n-1) + n$$

$$T(n) = \frac{n(n+1)}{2} = O(n^2)$$

$$= \frac{(n-1)(n-1+1)}{2} = \frac{n(n-1)}{2}$$

$$= \frac{n(n-1)}{2} \leq c(n-1)^2 \rightarrow T(n-1) \leq c(n-1)^2$$

19. Greedy Algorithm Design:

- The technique is being produced part by part and the first benefit is that it is being counter while at the same time, selecting the next part.

Example: Prim's algorithm.

Inductive Algarithm Design:

- The technique in this "if-then" mechanism is being praduced by the set of classification rules.

Example: an Algorithm for face recognition.