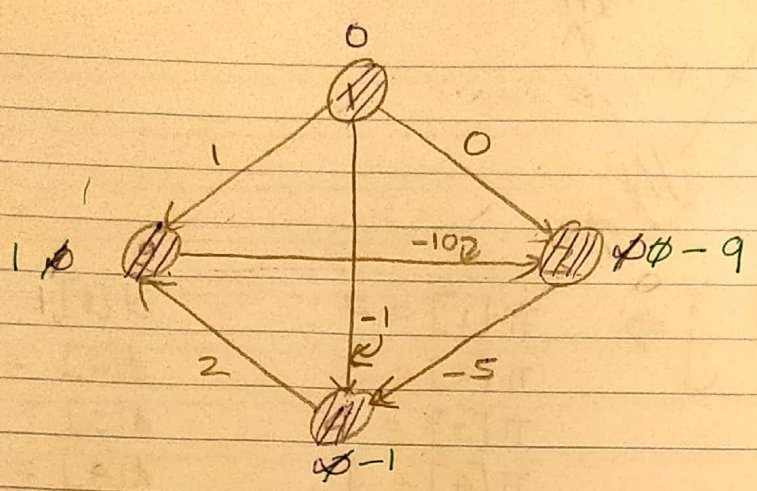
ASSIGNMENT 8

ARSHITHA BASAVARAJ

② ③ ④ form a negative weight cycle

1a)



Queue	ID	1st Iteration	2nd Iteration	3rd Iteration	4th Iteration
4	∞	-1 → min	~~X~~	~~X~~	~~X~~
3	∞	0	0	-9 → min	~~X~~
2	∞	1	1 → min	~~X~~	~~X~~
1	0	~~X~~	~~X~~	~~X~~	~~X~~
	min				

Order of Traversal

① ④ ② ③

$\pi[1] = \text{NULL}$ $d[1] = 0$

$\pi[2] = 1$ $d[2] = 1$

$\pi[3] = 2$ $d[3] = -9$

$\pi[4] = 1$ $d[4] = -1$

} Bykstra's output

Actual Output

$\pi[1] = \text{NULL}$ $d[1] = 0$

$\pi[2] = 1$ $d[2] = 1$

$\pi[3] = 2$ $d[3] = -9$

$\pi[4] = 3$ $d[4] = -14$

1b) To find the shortest path in a weighted undirected graph (with weights 1 or 2) in $\Theta(V+E)$ time we can use BFS.

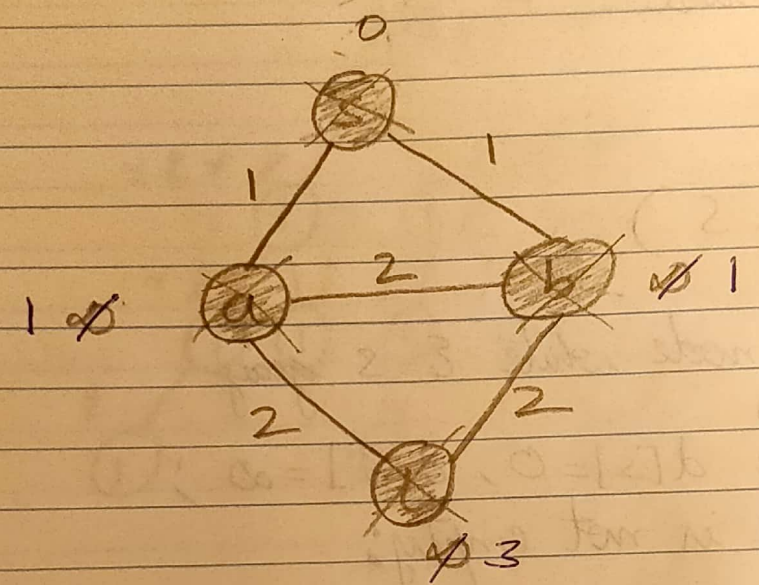
Pseudo code

BFS (G, S)

$\Theta(V)$ color each node white & s gray
$\Theta(1)$ enqueue (S)
$\Theta(V)$ $\pi[s] = \text{null}$; $d[s] = 0$, $d[v] = \infty$;
while queue is not empty:
    $\Theta(1)$ h = head (queue)
    $\Theta(E)$ for each neighbor n of h
        $\Theta(V)$ if n is white & $d(n) > d(h) + w(h,n)$
            color it gray
            $d[n] = d(h) + w(h,n)$
            $\pi[n] = h$
            $\Theta(V)$ enqueue (n).
    $\Theta(V)$ dequeue (h), color h black

$\Theta(V+E)$

Applying the above pseudo-code on the following graph,



Queue

~~s~~
~~a~~
~~b~~
~~t~~

$\pi[s] = null \quad d[s] = 0$
$\pi[a] = s \quad d[a] = 1$
$\pi[b] = s \quad d[b] = 1$
$\pi[t] = a \quad d[t] = 3$

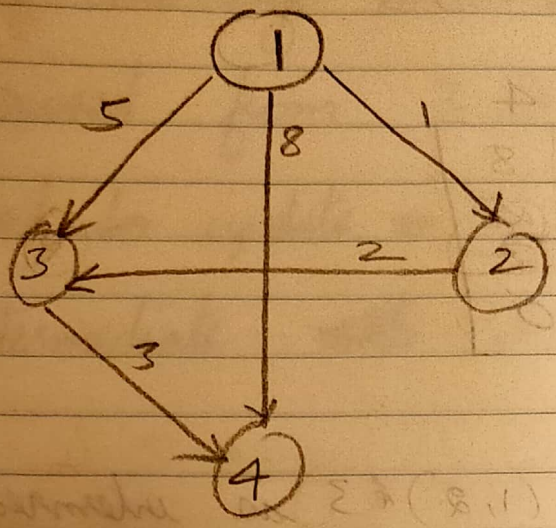
∴ Shortest path from s to t is

$s \rightarrow a \rightarrow t$

Runtime is $\Theta(V+E)$

27

a.



$D^0 \rightarrow$ adjacency matrix

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

$D^1 \rightarrow$ may use vertex 1 as intermediate vertex

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

D^2 - may use (1), 2 as intermediate vertex

	1	2	3	4
1	0	1	3	8
2	∞	0	2	∞
3	∞	∞	0	3
4	∞	∞	∞	0

D^3 - may use (1, 2) & 3 as intermediate vertex

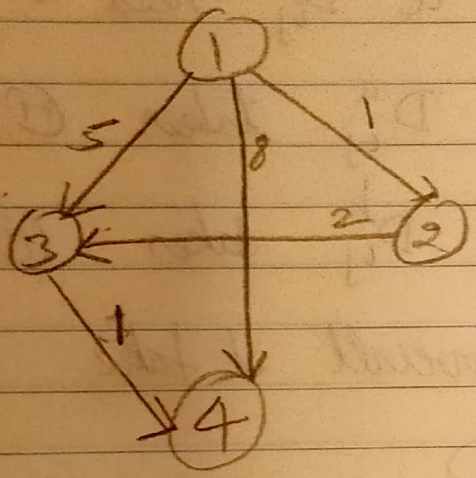
	1	2	3	4
1	0	1	3	6
2	∞	0	2	5
3	∞	∞	0	3
4	∞	∞	∞	0

D^4 - may use (1, 2, 3) 4 are intermediate vertex

	1	2	3	4
1	0	1	3	6
2	∞	0	2	5
3	∞	∞	0	3
4	∞	∞	∞	0

D^4 is the final matrix we get if we applied Floyd-Warshall on the above shown graph.

Now, if the edge between (3,4) were to be decreased from 3 to 1, then we'll only need to update D^3 & D^4 that use ③ & ④ as intermediate vertices.



It can be verified that D^1 & D^2 don't change

D^3 will now be

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 3 & 4 \\ \infty & 0 & 2 & 3 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix} \end{array}$$

∴ in this graph no edge goes out of ④

D^4 will be same as D^3.

∴ Updating the matrix D will take

$\Theta(V^2)$

Algorithm

① If w of e(a,b) is decreased, then only D_{ij}^{a} & D_{ij}^{b} need to be ~~update~~ updated.

② Updating D_{ij}^{a} takes $\Theta(V^2)$

③ Updating D_{ij}^{b} takes $\Theta(V^2)$

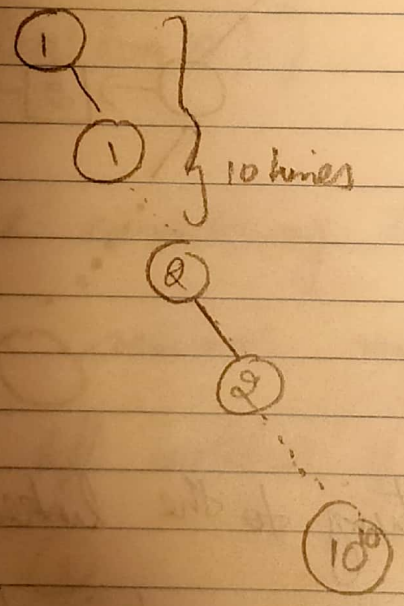
④ Therefore, overall it take $\Theta(V^2)$ to update D.

3.

a) If $n = 10^{10} \Rightarrow \log n = 10 \log 10 = 10$.

∴ 1, 1 .... ten times, 2, 2, ... ten times ..... so on till 10^{10}.

Keys of equal value will be inserted to the right subtree, to maintain uniformity.



∴ Height, h of the BST above is

$$h = 10 \times 10^{10} = 10^{11}$$

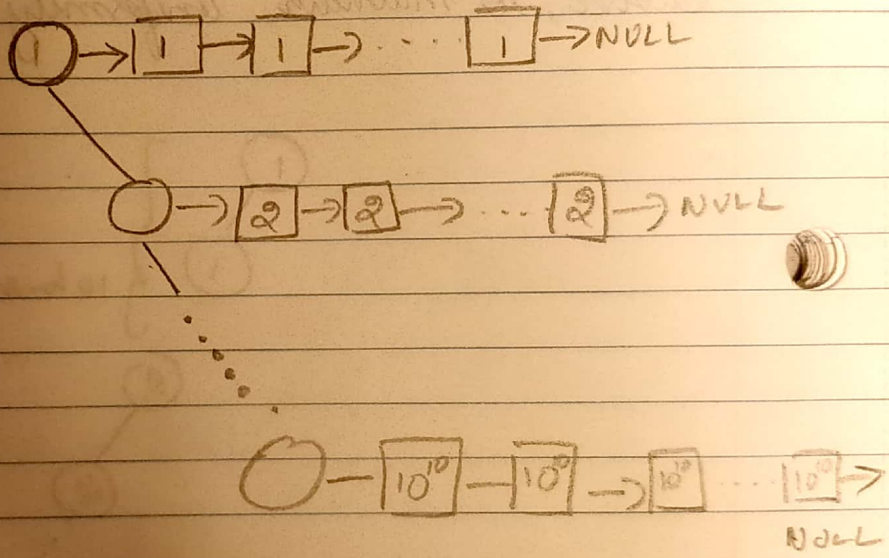
∴ Worst case time for insert will be

$$\Theta(h) = \Theta(10^{11})$$

In general for any n, it'll be $\Theta(n \log n)$

b) Assumption: Singly linked list is used & that the new node is inserted at the head.

Inserting to the head of the linked list is $\Theta(1)$.



Since inserting to the linked list happens in constant time, runtime for insert depends on the tree's height.

$$h = n = 10^{10}$$

$\Rightarrow$ Worst case time for insert

is $\Theta(h) = \Theta(n) = \Theta(10^{10})$

In general, $\Theta(n)$

2nd assumption: If we were to insert to the tail of the linked list, then it's time complexity would be

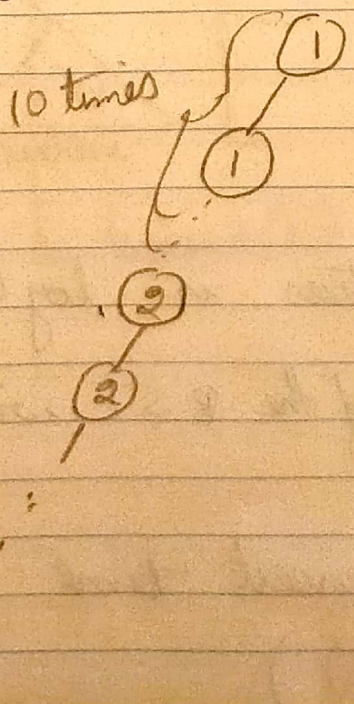
$\Theta(\log(\log n))$ ← size of the linked list

height of the tree is n

∴ Worst case insert time will be

$\Theta(n + \log(\log n))$

c. Since we are considering the worst-case scenario, let's assume my algorithm always randomly picks left subtree of insertion.

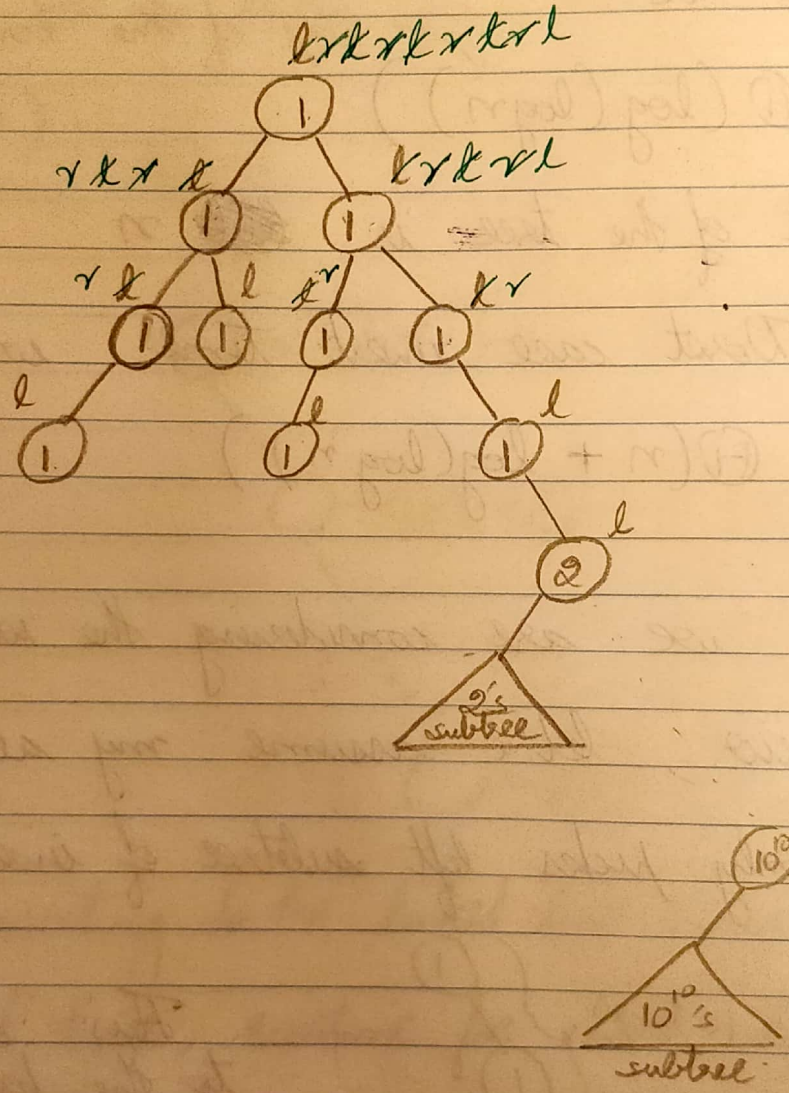


This case is similar to the first case

∈ worst-case time would be $\Theta(n \log n)$

d. $l \rightarrow$ "left" boolean flag

$r \rightarrow$ "right" boolean flag



Height of duplicates subtrees is $\log(\log n)$

Height of unique keys of the BST will be n

Therefore, worst case insert time would be $\Theta(n \log(\log n))$