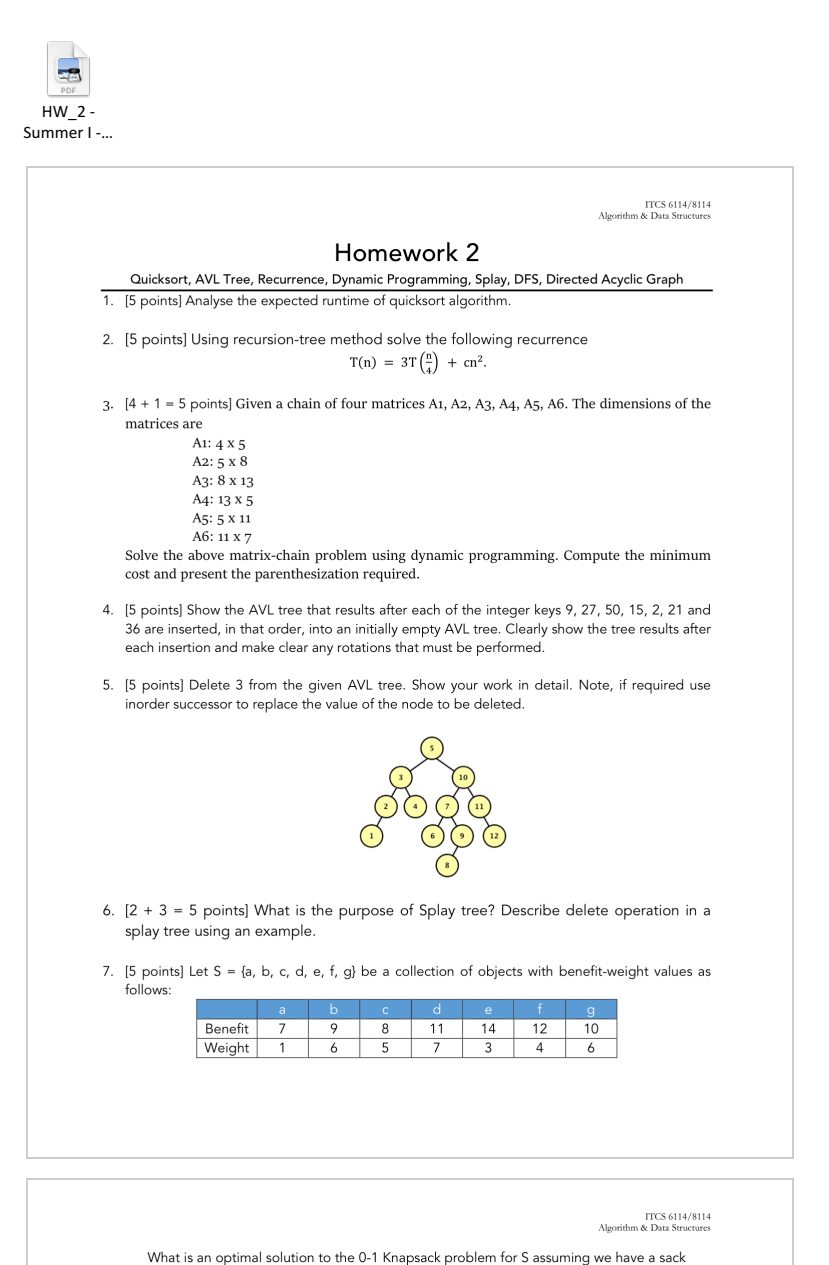
HW2

Thursday, 29 May 2025 2:10 PM

HW_2 - Summer I -...

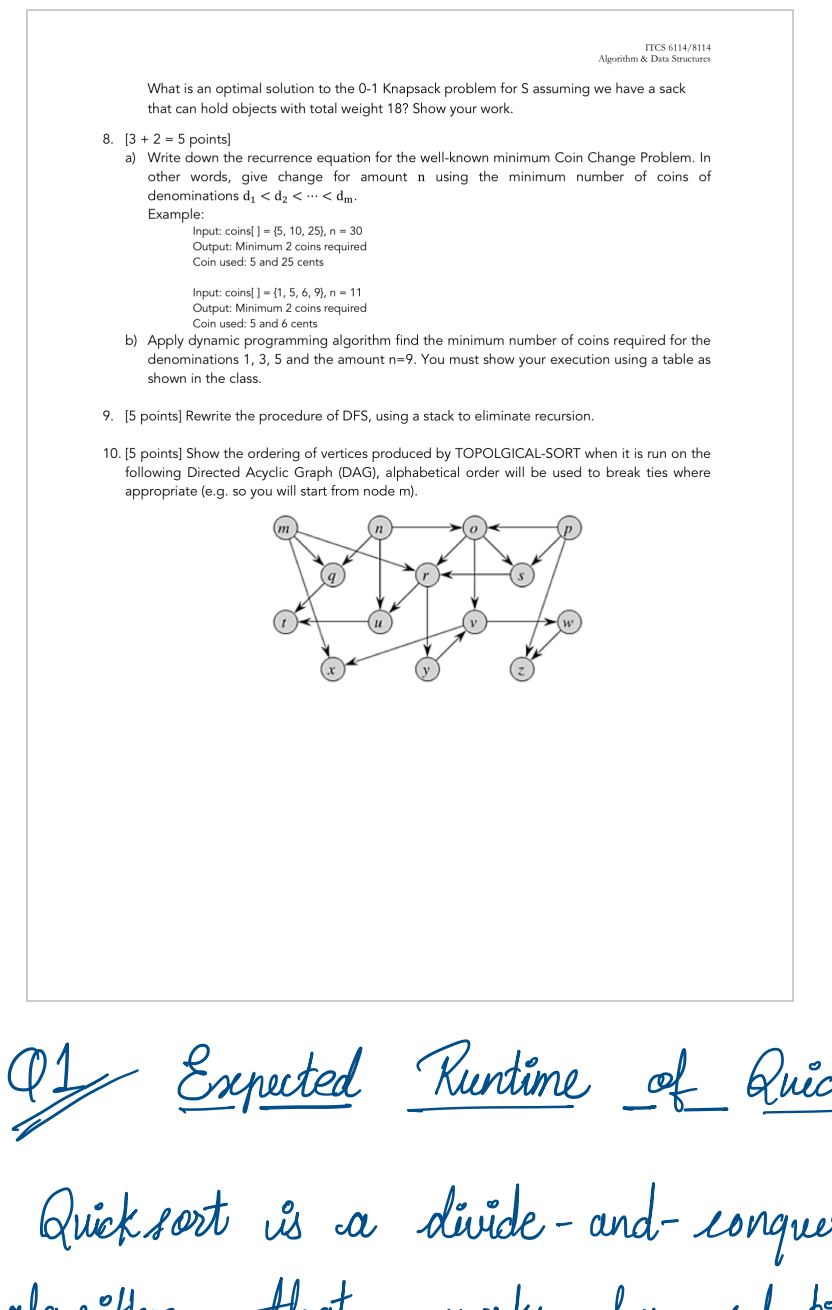

ITCS 6114/8114
Algorithm & Data Structures

Homework 2

Quicksort, AVL Tree, Recurrence, Dynamic Programming, Splay, DFS, Directed Acyclic Graph

1. [5 points] Analyse the expected runtime of quicksort algorithm.
2. [5 points] Using recursion-tree method solve the following recurrence $T(n) = 3T\left(\frac{n}{4}\right) + cn^2$.
3. [4 + 1 = 5 points] Given a chain of four matrices A1, A2, A3, A4, A5, A6. The dimensions of the matrices are
   A1: 4 x 5
   A2: 5 x 8
   A3: 8 x 13
   A4: 13 x 5
   A5: 5 x 11
   A6: 11 x 7
   Solve the above matrix-chain problem using dynamic programming. Compute the minimum cost and present the parenthesization required.
4. [5 points] Show the AVL tree that results after each of the integer keys 9, 27, 50, 15, 2, 21 and 36 are inserted, in that order, into an initially empty AVL tree. Clearly show the tree results after each insertion and make clear any rotations that must be performed.
5. [5 points] Delete 3 from the given AVL tree. Show your work in detail. Note, if required use inorder successor to replace the value of the node to be deleted.
6. [2 + 3 = 5 points] What is the purpose of Splay tree? Describe delete operation in a splay tree using an example.
7. [5 points] Let S = {a, b, c, d, e, f, g} be a collection of objects with benefit-weight values as follows:

	a	b	c	d	e	f	g
Benefit	7	9	8	11	14	12	10
Weight	1	6	5	7	3	4	6


ITCS 6114/8114
Algorithm & Data Structures

What is an optimal solution to the 0-1 Knapsack problem for S assuming we have a sack that can hold objects with total weight 18? Show your work.

8. [3 + 2 = 5 points]
   a) Write down the recurrence equation for the well-known minimum Coin Change Problem. In other words, give change for amount n using the minimum number of coins of denominations $d_1 < d_2 < \cdots < d_m$.
   Example:
   Input: coins[ ] = {5, 10, 25}, n = 30
   Output: Minimum 2 coins required
   Coin used: 5 and 25 cents

   Input: coins[ ] = {1, 5, 6, 9}, n = 11
   Output: Minimum 2 coins required
   Coin used: 5 and 6 cents
   b) Apply dynamic programming algorithm find the minimum number of coins required for the denominations 1, 3, 5 and the amount n=9. You must show your execution using a table as shown in the class.
9. [5 points] Rewrite the procedure of DFS, using a stack to eliminate recursion.
10. [5 points] Show the ordering of vertices produced by TOPOLGICAL-SORT when it is run on the following Directed Acyclic Graph (DAG), alphabetical order will be used to break ties where appropriate (e.g. so you will start from node m).

Q1 Expected Runtime of Quicksort

Quicksort is a divide-and-conquer sorting algorithm that works by selecting a 'pivot' element from the array and partitioning the other elements into two sub-arrays: those less than the pivot and those greater than the pivot. It then recursively applies the same operation to the sub-arrays.

1) Base Case: Occurs when the pivot splits the array into two equal halves every time. In this recurrence relation is: $T(n) = 2T(n/2) + O(n)$ Solving this gives $T(n) = O(n \log n)$.

2) Worst Case: Occurs when the pivot is always the smallest or largest element (e.g., the array is already sorted and the first or last element is always chosen as pivot). This leads to: $T(n) = T(n-1) + O(n)$ which solves to $T(n) = O(n^2)$

3) Average Case: Assuming that the pivot divides the array in a reasonably balanced way (i.e., not worst-case skewed), the expected time complexity is: $T(n) = T(kn) + T((1-k)n) + O(n)$ for some constant $0 < k < 1$; and it can be shown through probablistic analysis or recurrence tree method that this leads to:

$T(n) = O(n \log n)$

Thus, Quicksort has an expected runtime of $O(n \log n)$, which makes it one of the fastest sorting algorithms in practice due to low overhead and good cache performance, despite the worst-case scenario.

Q2

Using the recursion-tree method to expand the recurrence:

Level 0:

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

Level 1:

$3 \cdot T(n/4) = 3 \cdot (3T(n/16) + c(n/4)^2)$
$= 9T(n/16) + 3c(n^2/16)$

Level 2:

$9 \cdot T(n/16) = 9 \cdot (3T(n/64) + c(n/16)^2)$
$= 27T(n/64) + 9c(n^2/256)$

Level i:

$3^i T(n/4^i) + cn^2(3^i/16^i) = 3^i T(n/4^i) + cn^2(3/16)^i$

Eventually, when $n/4^i = 1 \Rightarrow i = \log_4 n$, the depth of the tree is $\log_4 n$.

Now, sum the cost at each level:

$T(n) = cn^2\left[1 + \frac{3}{16} + \left(\frac{3}{16}\right)^2 + \dots + \left(\frac{3}{16}\right)^{\log_4 n}\right]$

This is a geometric series with ratio $r = \frac{3}{16} < 1$ so the series converges to a constant:

$T(n) = O(n^2)$

The solution of this recurrence $T(n) = 3T(n/4) + cn^2$ is $O(n^2)$

Q3 Given matrices:

A_1 : 4 X 5
A_2 : 5 X 8
A_3 : 8 X 13
A_4 : 13 X 5
A_5 : 5 X 11
A_6 : 11 X 7

This gives the dimension vector

P = [4, 5, 8, 13, 5, 11, 7]

Let m[i][j] represent the minimum number of scalar multiplications required to compute the product $A_i A_{i+1} \cdots A_j$, and s[i][j] be the idex where the optimal split occurs.

Initialize m[i][j] = 0

Since multiplying one matrix requires zero scalar multiplications.

Calculating m[i][j] for increasing lengths of chains

ex. m[1][2] = m[1][1] + m[2][2] + P0 * P1 * P2
= 0 + 0 + 4×5×8 = 160

DP table m[i][j] (costs)

i/j	1	2	3	4	5	6
1	0	160	656	896	1324	1367
2		0	520	840	1240	1425
3			0	520	1034	1258
4				0	715	1080
5					0	385
6						0

Minimum Cost

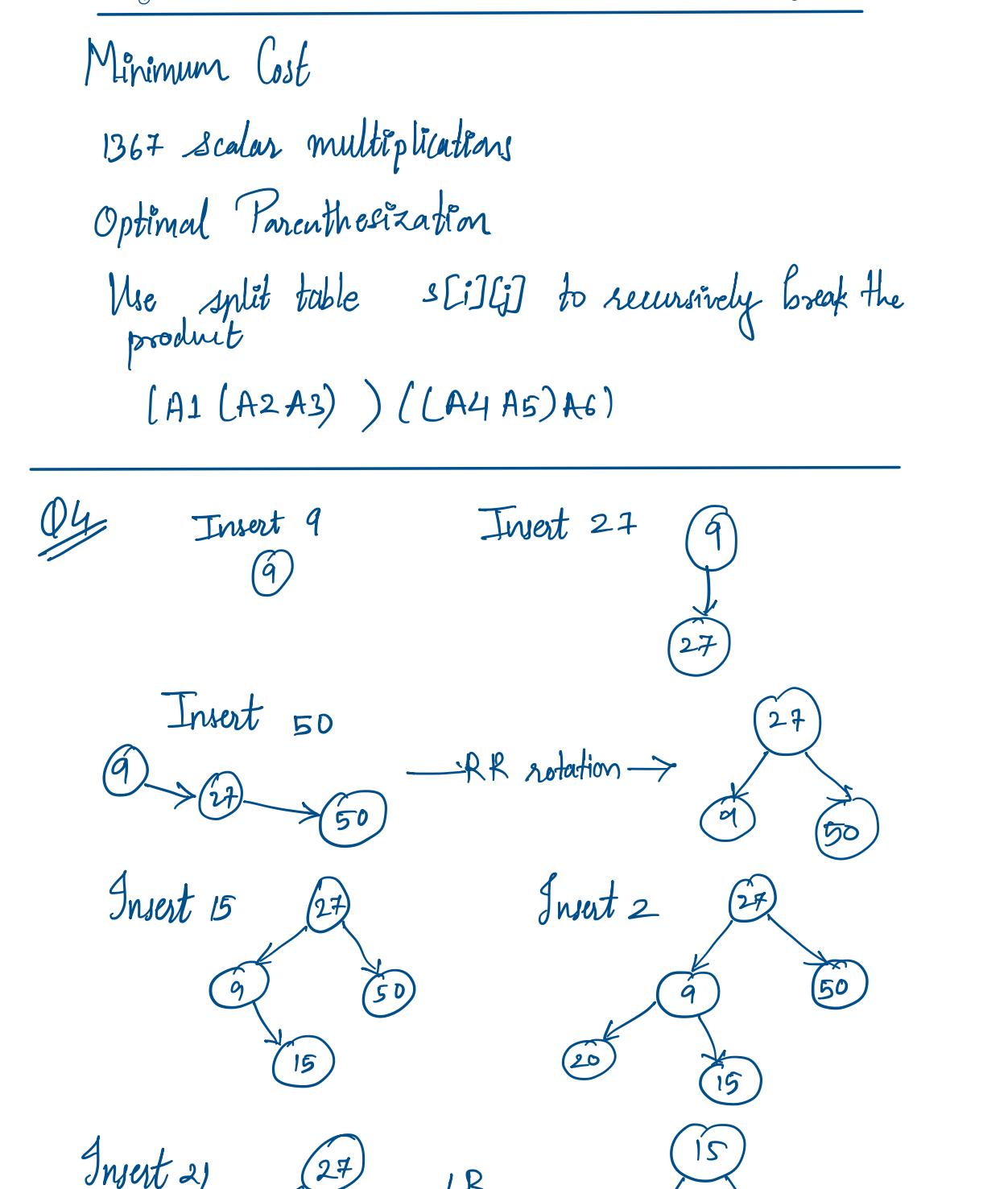
1367 scalar multiplications

Optimal Parenthesization

Use split table s[i][j] to recursively break the product

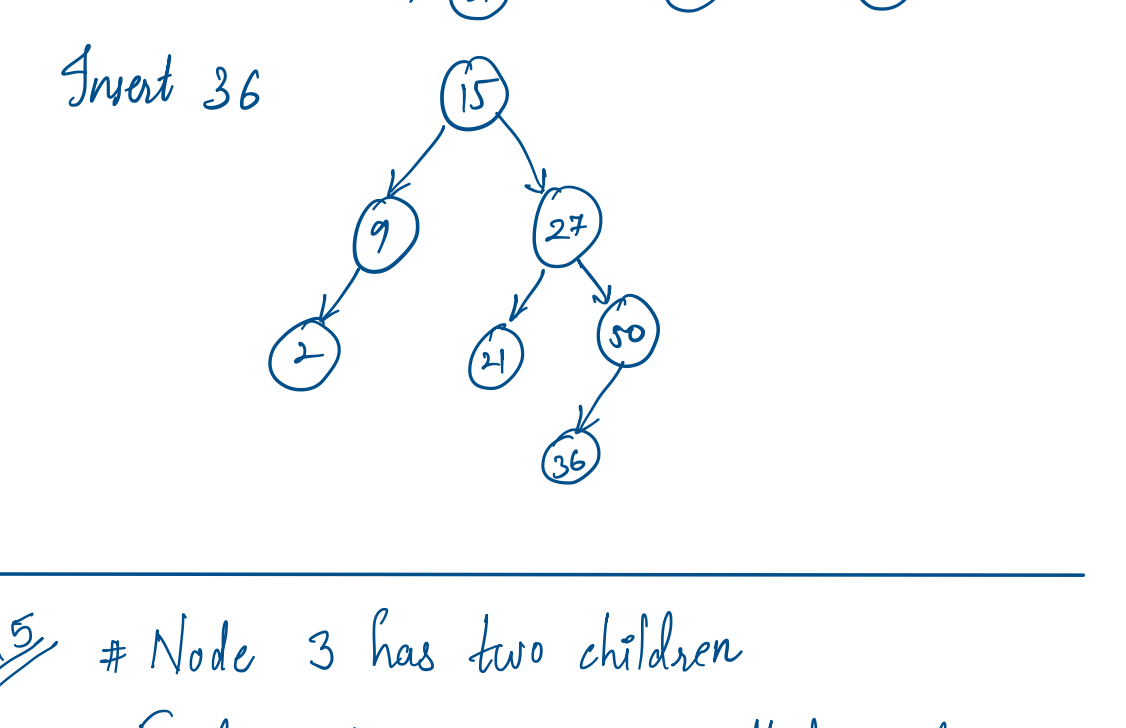
(A1 (A2 A3)) ((A4 A5) A6)

Q4

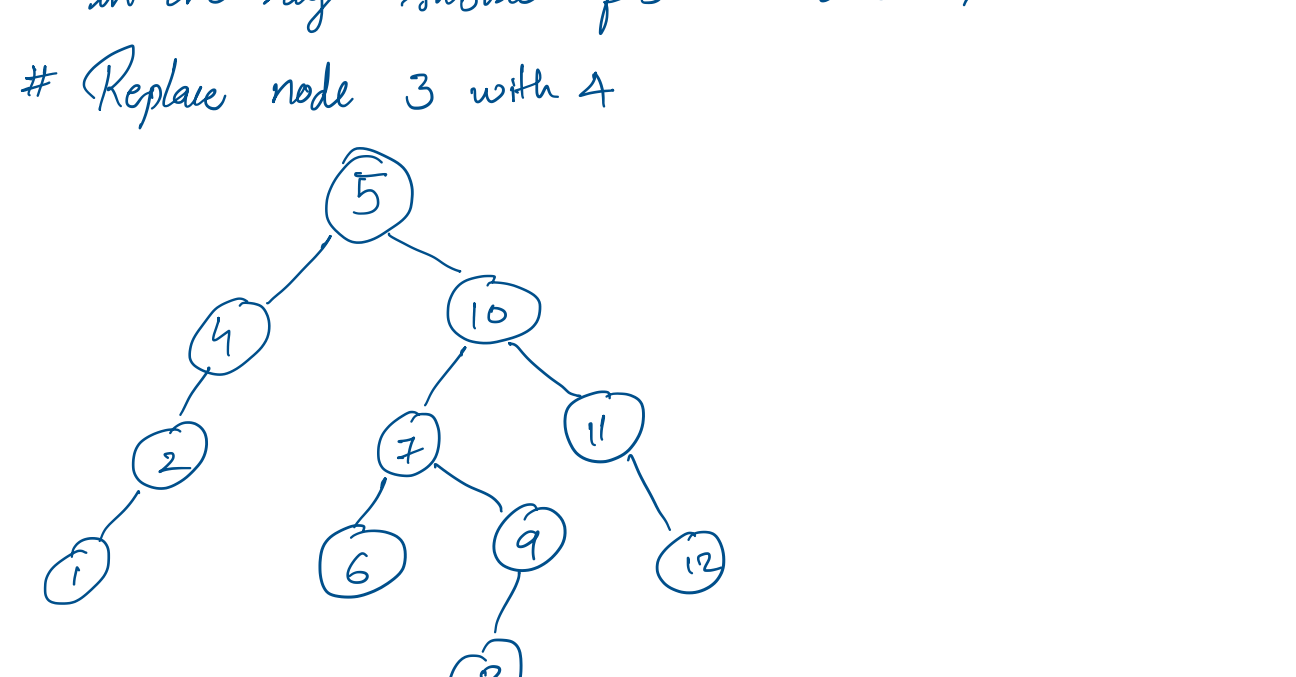


Q5

Node 3 has two children
Find inoder succssor: smallest node in the right subtree of 3 → it's node 4
Replace node 3 with 4



Check from deleted node upward
Subtree rooted at node 4:
Left subtree height = 2 (node 2 → 1)
Right = 0
Balance = 2 → unbalanced
Performing Right rotation at node 4



Q6 Purpose of Splay Tree

A Splay Tree is a type of self-adjusting BST. The main purpose of a splay tree is to improve acces time for recently accessed elements by moving them to the root using a process called splaying.

Advantages:

1) Frequently accessed elements are faster to reach.
2) No need to store balance factors or extra metadata.
3) Guarantee amortized $O(\log n)$ time complexity for operations.

Delete Operation in Splay Tree

To delete a node in a splay tree:

1) Splay the target node to the root.
2) Remove the root node.
3) If the root had two subtrees:
   - Splay the maximum node in the left subtree to make it the new root.
   - Attach the original right subtree as the right child of the new root.

Q7

V[i,w]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/a	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
2/b	0	7	7	7	7	7	9	16	16	16	16	16	16	16	16	16	16	16	16
3/c	0	7	7	7	7	8	15	16	16	16	16	17	24	24	24	24	24	24	24
4/d	0	7	7	7	7	8	15	16	18	18	18	18	24	26	27	27	27	27	28
5/e	0	7	7	14	21	21	21	21	22	29	30	32	32	32	32	38	40	41	41
6/f	0	7	7	14	21	21	21	26	33	33	33	33	41	41	42	44	44	44	44
7/g	0	7	7	14	21	21	21	26	33	33	33	33	41	41	43	44	44	44	44

Q8

a) Recurrence:

$F(n) = \min_{i : n \geq d_i} \{F(n - d_i)\} + 1$ for $n > 0$,

$F(0) = 0$

b) Coins = [1, 3, 5], Amount = 9 DP Table:

dp [0] = 0
dp [0] = 1
dp [2] = 2
dp [3] = 1
dp [4] = 2
dp [5] = 1
dp [6] = 2
dp [7] = 3
dp [8] = 2
dp [9] = 3

Answer: Minimum Coins = 3 (5+3+1)

Q9 DFS Using Stack

```
DFS (graph, start):
    stack = [start]
    visited = set()
  while stack:
      vertex = stack.pop()
      if vertex not in visited:
          visited.add(vertex)
          for neighbor in reversed(graph[vertex]):
              if neighbor not in visited:
                  stack.append(neighbor)
```

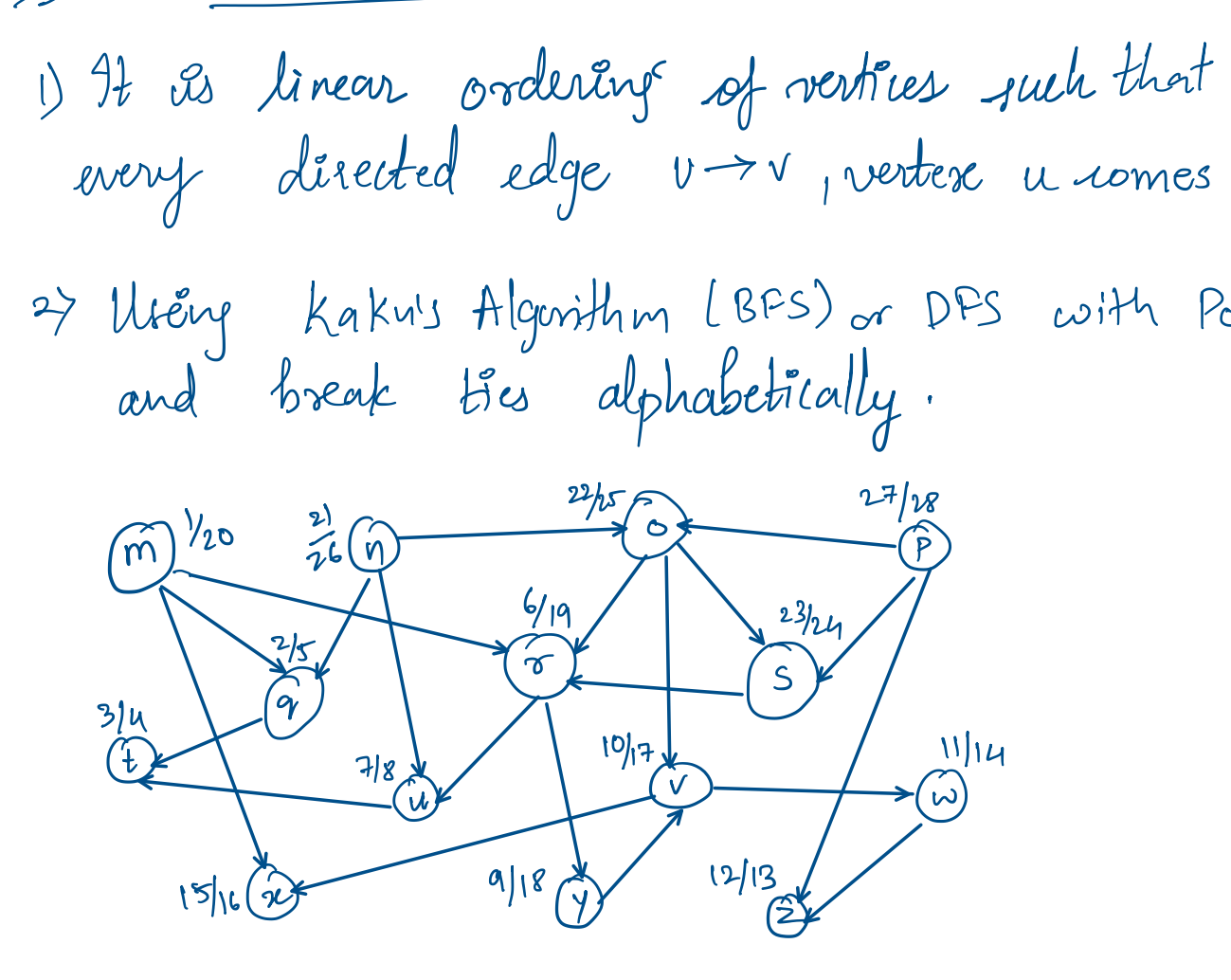
1) Initialize an empty stack and push the starting node onto it.
2) While the stack is not empty, perform following:
   i) Pop the top node from the stack & process it.
   ii) For each neighbor of popped node that has not been visited, push the neighbor onto the stack and mark it as visited.

By using the stack to manage the nodes processed, you effectively eliminate the need for recursion in DFS procedure.

Q10 TOPOLOGICAL-SORT on DAG

1) It is linear ordering of vertices such that for every directed edge u → v, vertex u comes before v

2) Using Kahn's Algorithm (BFS) or DFS with Post order and break ties alphabetically.



Ordering of vertices: