(91)

DAA Assignment - I

Harith. Y
CS23I1027
B2 (CSE-DD)

Q1) Let $T(n)$ be the Recursive relation defined as follows:

$T(0) = 1$

$T(1) = 2$ and

$T(n) = 5T(n-1) - 6T(n-2)$ for $n \geq 2$

Which one of the following statements is true ?

(a) $T(n) = \theta(2^n)$

(b) $T(n) = \theta(n.2^n)$

(c) $T(n) = \theta(3^n)$

(d) $T(n) = \theta(n.3^n)$

[GATE CSE 2024 SET 2]

Solution :

(a) $T(n) = \theta(2^n)$

Given:

$T(0) = 1$
$T(1) = 2$
$T(n) = 5T(n-1) - 6T(n-2) \quad \forall n \geq 2$

Assume $T(n) = r^n$

$\Rightarrow r^n = 5r^{n-1} - 6r^{n-2}$

$\Rightarrow r^2 = 5r - 6$

$\Rightarrow r^2 - 5r + 6 = 0$

$\Rightarrow \boxed{r=3}$ or $\boxed{r=2}$

$\therefore T(n) = A \cdot 3^n + B \cdot 2^n$

Now, to find A and B, we use initial conditions.

$T(0) = 1 = A + B \quad -(1)$

$T(1) = 2 = 3A + 2B \quad -(2)$

from (1) and (2):

$\boxed{A=0}$ & $\boxed{B=1} \implies T(n) = 2^n$

$\therefore T(n) = \theta(2^n)$ (a)

Q2) Let f and g be functions of natural numbers given by

$f(n) = n$

$g(n) = n^2$

Which of the following statements is/are TRUE?

(a) $f \in O(g)$

(b) $f \in \Omega(g)$

(c) $f \in o(g)$

(d) $f \in \Theta(g)$

[GATE CSE 2013]

Solution:

(a) $f \in O(g)$ and (c) $f \in o(g)$

For option (a):

$f \in O(g)$

$\Rightarrow$ i.e. $f(n) \leq C \cdot g(n)$ $\forall n \geq n_0$, where $C > 0$ & $n_0 \geq 1$

$\Rightarrow n \leq C \cdot n^2$

Say $C = 1$ and any $n_0 \geq 1$

$\Rightarrow n \leq n^2$

$\therefore$ Equality holds true $\forall n \geq 1$.

$\therefore$ (a) $\longrightarrow$ True

For option (b):

$f \in \Omega(g)$

$\Rightarrow$ i.e. $f(n) \geq C \cdot g(n)$ $\forall n \geq n_0$, where $C > 0$ & $n_0 \geq 1$

$\Rightarrow n \geq C \cdot n^2$

for any $C > 0$, this ~~cond~~ inequality won't hold true.

$\therefore$ (b) $\rightarrow$ false

For option (c):

$f \in o(g)$

$\Rightarrow$ i.e. $f(n) < C \cdot g(n)$ $\forall n \geq n_0$, where $\epsilon > 0$ & $n_0 \geq 1$

$\Rightarrow n < \epsilon \cdot n^2$

$\exists C, C > 0$, we can find $n_0 > \frac{1}{C}$,

Harith.Y
CS23I1027
B2 (CSE-DD)

$\therefore$ Inequality holds true

$\therefore$ (c) $\rightarrow$ True

For option (d):

$f(n) \in \theta(g)$

$\Rightarrow$ i.e. $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$, where $c_1, c_2 > 0$ and $n_0 \geq 1$ $[\forall n \geq n_0]$

$\Rightarrow c_1 \cdot n^2 \leq n \leq c_2 \cdot n^2$

As n^2 grows faster than n, This inequality can't hold.

$\therefore$ (d) $\rightarrow$ false

$\therefore$ (a) & (c) $\rightarrow$ True

(Q3) Consider the following Recurrance relation:

$$T(n) = \begin{cases} \sqrt{n} \cdot T(\sqrt{n}) + n, & \forall n \geq 1 \\ 1, & n = 1 \end{cases}$$

Which of the following options is correct ?

(a) $T(n) = \theta(n \cdot \log(\log(n)))$

(b) $T(n) = \theta(n \log n)$

(c) $T(n) = \theta(n^2 \log n)$

(d) $T(n) = \theta(n^2 \log(\log n))$

[GATE CSE 2024 SET-1]

Solution:

(a) $T(n) = \theta(n \cdot \log(\log n))$

As

$$T(n) = \begin{cases} \sqrt{n} \cdot T(\sqrt{n}) + n, & \forall n \geq 1 \\ 1, & n = 1 \end{cases}$$

Let $n = 2^m$

$\Rightarrow \sqrt{n} = 2^{m/2}$

Substitute this in $T(n)$ for $n \geq 1$:

$T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$

$\Rightarrow T(2^m) = \sqrt{2^m} \cdot T(2^{m/2}) + 2^m$

$= 2^{m/2} \cdot T(2^{m/2}) + 2^m$

Let $S(m) = T(2^m)$

$\Rightarrow S(m) = 2^{m/2} S(m/2) + 2^m$

We can assume $S(m) = 2^m \cdot f(m)$

$\Rightarrow 2^m f(m) = 2^{m/2} \cdot 2^{m/2} \cdot f(m/2) + 2^m$

$\Rightarrow 2^m f(m) = 2^m f(m/2) + 2^m$

$\Rightarrow f(m) = f(m/2) + 1$

Now,

$f(m/2) = f(m/4) + 1$

$f(m/4) = f(m/8) + 1$

$\Rightarrow f(m) = f(1) + \log(m)$

↳ constant

$\Rightarrow f(m) = C + \log(m)$

$\Rightarrow S(m) = 2^m (C + \log m)$

$\Rightarrow T(2^m) = 2^m (C + \log m)$

$\Rightarrow T(m) = m\{C + \log(\log m)\}$

Asymptotically,

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

$\therefore$ (a) $\longrightarrow$ True

Q4) Consider the following recurrence:

$f(1) = 1$;

$f(2n) = 2f(n) - 1$, $\forall n \geq 1$;

$f(2n+1) = 2f(n) + 1$, $\forall n \geq 1$;

Then, which of the following statements is/are True?

Harith. Y
CS23I1027
B2 [CSE-DD]

(a) $f(2^n-1) = 2^n-1$

(b) $f(2^n) = 1$

(c) $f(5.2^n) = 2^{n+1}+1$

(d) $f(2^n+1) = 2^n+1$ [GATE CSE 2022]

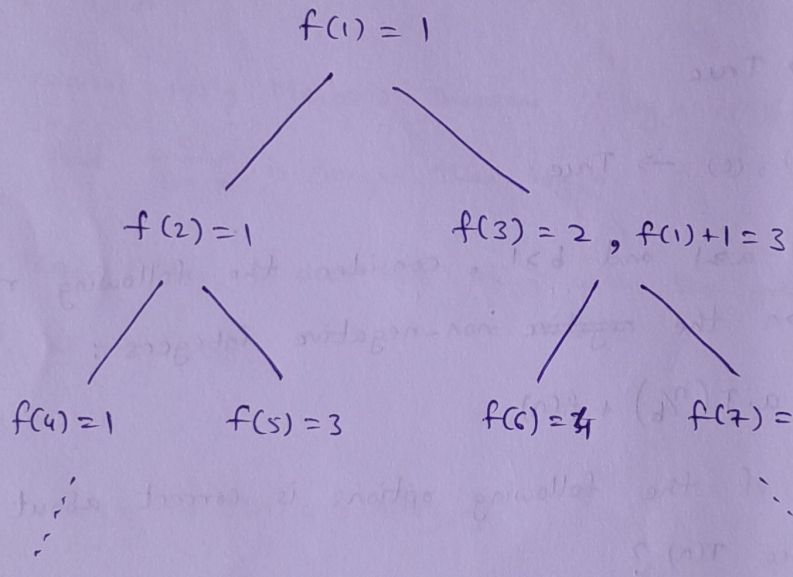
Solution:

(a), (b) & (c) are True.

Given:

When input to function f is even, $f(2n) = 2f(n)-1 \quad \forall n \geq 1$

When input to function f is odd, $f(2n+1) = 2f(n)+1 \quad \forall n \geq 1$



$f(2^n) = 1$ ∴ option (b) → True

$f(2^n-1) = 2^n-1$

∴ Option (a) → True

For option (d):

∵ 2^n+1 is definetily odd,

$f(2^n+1) = 2f(2^{n-1}) + 1$

$= 2+1 \quad [\because f(2^{n-1}) = 1]$

$= 3$

∵ (d) → false

For option (c):

∵ 5.2^n is definitely an even number $\forall n \geq 1$

$f(5.2^n) = 2f(5.2^{n-1}) - 1$

$\Rightarrow 2(2f(5\cdot 2^{n-2})-1)-1$

$\Rightarrow 2^2 \cdot f(5\cdot 2^{n-2}) - 2 - 1$

$\Rightarrow 2^2(2\cdot f(5\cdot 2^{n-3})-1) - 2 - 1$

$\Rightarrow 2^3 \cdot f(5\cdot 2^{n-3}) - 2^2 - 2 - 1$

$\vdots$

$\Rightarrow 2^n f(5\cdot 2^0) - (2^{n-1}) - (2^{n-2}) - (2^{n-3}) \ldots\ldots - 2 - 1$

$\Rightarrow 2^n f(5) - (1+2+4+\ldots\ldots 2^{n-2} + 2^{n-1})$

$\Rightarrow 2^n \cdot (3) - (2^n - 1)$

$\Rightarrow 2^{n+1} + 1$

$\therefore$ (c) $\rightarrow$ True

$\therefore$ (a), (b), (c) $\rightarrow$ True

Q5) For constants $a \geq 1$ and $b > 1$, consider the following recurrence defined on the ~~negative~~ non-negative integers:

$$T(n) = a \cdot T(n/b) + f(n)$$

Which one of the following options is correct about the recurrence $T(n)$?

(a) If $f(n)$ is $\frac{n}{\log_2(n)}$, Then $T(n) = \theta(\log_2 n)$.

(b) If $f(n)$ is $n\log_2(n)$, Then $T(n) = \theta(n\log_2(n))$.

(c) If $f(n)$ is $O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, Then $T(n) = \theta(n^{\log_b a})$

(d) If $f(n)$ is $\theta(n^{\log_b a})$, Then $T(n) = \theta(n^{\log_b a})$

[GATE CSE 2021 SET-2]

Solution:

(c) $\exists \varepsilon > 0$ S.T. $f(n) = O(n^{\log_b a - \varepsilon})$, Then $T(n) = \theta(n^{\log_b a})$

Master Theorem overview:

There are 3 cases, they are:

Harith. Y
CS23I1027
B2 [CSE-DD]

Case - I : If $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$,

Then $T(n) = \Theta(n^{\log_b a})$

Case - II : If $f(n) = \Theta(n^{\log_b a} \cdot \log^k n)$ for some $k \geq 0$,

Then $T(n) = \Theta(n^{\log_b a} \cdot \log^{k+1} n)$

Case - III : If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and

$af(n/b) \leq c \cdot f(n)$ for some $c < 1$ for sufficiently large n,

Then $T(n) = \Theta(f(n))$

For option (a) :

$$f(n) = \frac{n}{\log_2 n} \Rightarrow T(n) = \Theta(\log_2 n)$$

We cannot apply Master's Theorem $\because \log_2 n$ is smaller than any n^{ε}

And $\frac{n}{\log_2 n}$ is larger than $\log_2 n$

$\therefore T(n) = \Theta(\log_2 n)$ does not align with the growth rate of $f(n)$

$\therefore$ (a) $\rightarrow$ False.

For option (b) :

$f(n) = n \log_2 n \Rightarrow T(n) = \Theta(n \cdot \log_2 n)$

Here, we do not have sufficient data, i.e. we need values of a and b to calculate $\log_b a$ and compare $f(n)$ with $n^{\log_b a}$.

$\therefore$ (b) $\rightarrow$ False

For option (c) :

Case - I of Master's Theorem.

Hence, (c) $\rightarrow$ True.

For option (d) :

Directly applies to Master's Theorem, Case - II

When $k = 0$, ~~True~~

~~True~~

i.e. $f(n) = \Theta(n^{\log_b a} \cdot \log^0 n)$

$\Rightarrow T(n) = \Theta(n^{\log_b a} \cdot \log^{0+1} n)$

$= \Theta(n^{\log_b a} \cdot \log n)$

$\therefore$ (d) $\rightarrow$ ~~False~~ True

Both (c) and (d) are True,

But (c) precisely matches Master's Theorem, Case-2 without any ambiguity.

$\therefore$ (c) $\rightarrow$ True Answer

Q6) Consider the following array :

23 32 45 69 72 73 89 97

Which algorithm of the following options uses the least number of comparisons (among the following array elements) to sort above array in ascending order ?

(a) Insertion Sort
(b) Selection Sort
(c) Quick sort using last element as pivot
(d) Merge Sort

[GATE CSE 2021 SET-1]

Solution:

(a) Insertion Sort

As the given array is already sorted in ascending order,

(a) Insertion sort :

- Best for nearly sorted data, i.e. Performs minimum comparisons in the best case.
- Total Comparisons $= n-1 = 8-1 = 7$

(b) Selection sort :

- Always performs the same number of comparisons irrespective of the initial order.
- Total Comparisons $= \frac{n(n-1)}{2} = \frac{8 \times 7}{2} = 28$

option (c) : Quick Sort

- For an already sorted array with the last element as pivot, it results in the worst-case scenario.

- Total comparisons = $7+6+5+4+3+2+1 = 28$

(d) : Merge Sort

- Does not take advantage of pre-sortedness
- Total comparisons = $4+6+7 = 17$

$\therefore$ (a) $\rightarrow$ best

Q7) Let A be an array containing integer values. The distance of A is defined as the minimum number of elements in A that must be replaced with another integer so that the resulting array is sorted in non-decreasing order. The distance of the array $[2,5,3,1,4,2,6]$ is

Solution : 3

[GATE CSE 2024 SET-2]

Given, $A = [2,5,3,1,4,2,6]$

Finding Longest Increasing Subsequence,

Start with first element, LIS = [2]

Next element is 5, $5>2 \Rightarrow$ LIS = [2,5]

Next element is 3, $3<5 \Rightarrow$ LIS = [2,3]

Next element is 1, $1<3 \Rightarrow$ LIS = [2,3]

Next element is 4, $4>3 \Rightarrow$ LIS = [2,3,4]

Next element is 2, $2<4 \Rightarrow$ LIS = [2,3,4]

Next element is 6, $6>4 \Rightarrow$ LIS = [2,3,4,6]

Now, Distance = Length of Array − Length of LIS

$= 7 - 4$

$= 3$

$\therefore$ Distance of Array A = 3

i.e. 3 elements need to be replaced to turn the array into a non-decreasing order.

Q8) Let H be a binary min-heap consisting of n elements implemented as an array. What is the worst case time complexity of an optimal algorithm to find the maximum element in H?

(a) $\theta(\log n)$

(b) $\theta(n)$

(c) $\theta(1)$

(d) $\theta(n \log n)$

[GATE CSE 2021 SET-2]

Solution:

(b) $\theta(n)$

Finding max. element:

Start index of leaves = $\lfloor \frac{n}{2} \rfloor + 1$

$\because$ no. of leaf nodes = $\lceil \frac{n}{2} \rceil$

End index of leaves = n

$\therefore$ Iterate over leaf nodes from index $\lfloor \frac{n}{2} \rfloor + 1$ to n and update max value if larger value found.

Scanning leaf nodes $\rightarrow$ $O(n/2)$

$\therefore$ Time complexity = $\theta(n)$

Worst case scenario $\rightarrow$ scan all leaf nodes, i.e. $\theta(n)$

Q9) What is the worst-case number of arithmetic operations performed by recursive binary search on a sorted array of size n

(a) $\theta(n)$

(b) $\theta(\sqrt{n})$

(c) $\theta(\log_2 n)$

(d) $\theta(n^2)$

[GATE CSE 2021 SET-2]

Solution:

(c) $\theta(\log_2 n)$

Compare low and high – 1 comparison

Harith. Y
CS23I1027
B2 (CSE-DD)

Compute midpoint – 2 Arithmetic operations

Comparison with target – 1 Comparison

∴ Total operations per Recursive call : c

Depth of Recursion :

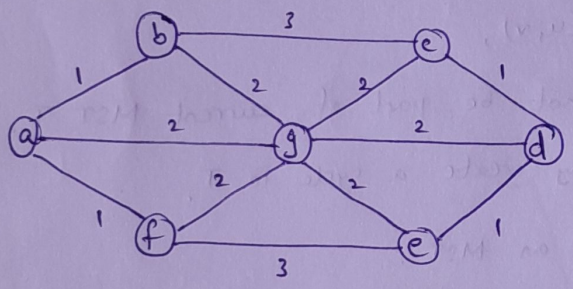
The max no. of times we can halve n elements is $\log_2 n$.

∴ Worst case depth = $\lceil \log_2 n \rceil$

& Total operations = $c \times \lceil \log_2 n \rceil$

∴ Order of Growth = $\theta(\log_2 n)$ (c)

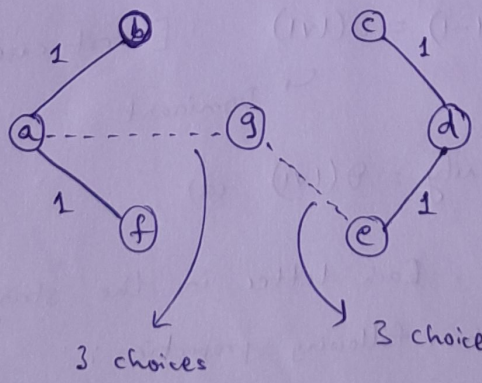
Q10) The number of distinct minimum-weight spanning trees of the following graph is



[GATE CSE 2024 SET-2]

Solution :

9 distinct minimum-weight spanning tree



∴ No. of MST = ${}^3C_1 \times {}^3C_1 = 3 \times 3 = 9$

∴ 9 distinct MSTs

Q11) Let $G = (V, E)$ be a weighted undirected graph and let T be a Minimum Spanning Tree (MST) of G maintained using adjacency lists. Suppose a new weighted edge $(u, v) \in V \times V$ is added to G. The worst case time complexity of determining if T is still an MST of the resultant graph is

(a) $\Theta(|E| + |V|)$

(b) $\Theta(|E| \cdot |V|)$

(c) $\Theta(|E| \cdot \log|V|)$

(d) $\Theta(|V|)$ [GATE CSE 2020]

Solution:

(d) $\Theta(|V|)$

When adding new edge (u, v),

- It may or may not be part of current MST T.
- It could potentially create a cycle in T.

Determining if T is still an MST:

- Find a path from u to v in T.
- Keep track of edge with max weight
- Compare Edge weights

  i.e. weight $(u, v) <$ weight$(e_{max}) \Rightarrow$ T is not an MST anymore

Time complexity = $O(|V|-1) = O(|V|)$ [worst case]

↳ Dominant

∴ Overall Time complexity = $\Theta(|V|)$ (d)

Q12) Consider the string abbccddeee. Each letter in the string must be assigned a binary code satisfying the following properties:

(i) For any two letters, the code assigned to one letter must not be a prefix of the code assigned to the other letter.

(ii) For any two letters of the same frequency, the letter which occurs

earlier in the dictionary order is assigned a code whose length is at most the length of the code assigned to the other letter.

Among the set of all binary code assignments which satisfy the above two properties, what is the minimal length of the encoded string?

(a) 23

(b) 21

(c) 25

(d) 30

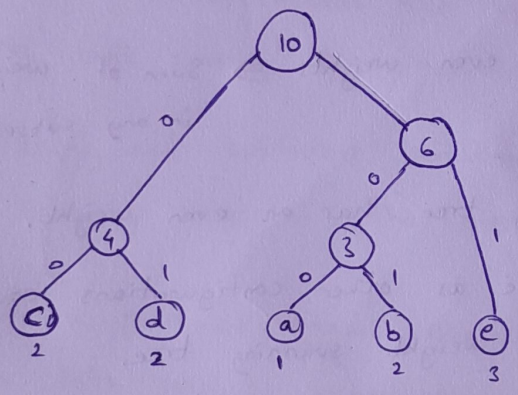
[GATE CSE 2021 SET-2]

Solution:

(a) 23

Given string : abbccddeee

Using Huffman coding:



Character	Frequency	Encoding
a	1	100
b	2	101
c	2	00
d	2	01
e	3	11

∴ Hence, Length of encoded string

$= (1\times3) + (2\times3) + (2\times2) + (2\times2) + (3\times2)$

$= 3 + 6 + 4 + 4 + 6$

$= 23$ (a)

Q13) Let G be an undirected connected graph in which every edge has a positive integer weight. Suppose that every spanning tree in G has even weight. Which of the following statements is/are TRUE for every such graph, G?

(a) All edges in G have even weights.

(b) All edges in G have even weights ~~or~~ OR all edges in G have odd weights.

(c) In each cycle C in G, all edges in C have even weights.

(d) In each cycle C in G, either all edges in C have even weights OR all edges in C have odd weights.

Solution

[GATE CSE 2024 SET-2]

(d)

Option (a):

All edges in G have even weights $\Rightarrow$ Sum of weights of the edges in any subset will also be even.

i.e. every spanning tree has an even weight.

This is not true as other configurations could also result in an even weight spanning tree.

option (b):

All Edges in G have even weights $\Rightarrow$ ~~Sum~~ Every spanning tree will have an even weight

All Edges in G have odd weights $\Rightarrow$ Each spanning tree, having odd no. of edges, the sum of the weights will be a sum of odd numbers

$\Downarrow$

Weight of spanning trees remain odd which contradicts given condition.

$\therefore$ Not true for every graph G

option (c) :

This option implies that within every cycle, every edge should have an even weight.

This strong condition is not necessary to ensure every spanning tree has an even weight.

∴ Not true necessarily for every such graph G.

option (d) :

All Edges in a cycle C have even weights ⟹ Contributes an even sum

All Edges in a cycle C have odd weights ⟹ Sum of weights will be even if there are even no. of edges.

This condition ensures that the sum of weights in any redundant paths isn't causing any inconsistencies regarding tree weights.

∴ Necessary and sufficient condition to ensure that every spanning tree in G has even weights.

Q14) Define R_n to be maximum amount earned by cutting a rod of length 'n' meters into one or more pieces of integer length and selling them. For $i>0$, let $p[i]$ denote the selling price of a rod whose length is 'i' meters. Consider the array of ~~pieces~~ prices :

$p[1] = 1$, $p[2] = 5$, $p[3] = 8$, $p[4] = 9$, $p[5] = 10$, $p[6] = 17$, $p[7] = 18$

Which of the following statements is/are correct about R_7 ?

(a) $R_7 = 18$

(b) $R_7 = 19$

(c) R_7 is achieved by 3 different solutions

(d) R_7 cannot be achieved by a solution consisting of three pieces

[GATE CSE 2021 SET-2]

Solution

(a), (c)

This question is similar to 0/1 Knapsack.

(i) Include 7 length rod ⇒ Profit = 18 — (1)

Now, Traverse through the path where we don't include 7.

(ii) Include 6

Only one case, (6,1) again leads to a profit of 18. — (2)

Now, traverse the path where we don't include 6.

(iii) Include 5

Two cases:

(5,2) ⇒ Profit = 15

(5,1,1) ⇒ Profit = 12

Now, traverse through the path where we don't include 5.

(iv) Include 4

Three cases:

(4,3) ⇒ Profit = 17

(4,2,1) ⇒ Profit = 15

(4,1,1,1) ⇒ Profit = 12

Now, traverse through the path where we don't include 4.

(v) Include 3

Three cases:

(3,2,2) ⇒ Profit = 18 — (3)

(3,2,1,1) ⇒ Profit = 15

(3,1,1,1,1) ⇒ Profit = 12

Now, Traverse through the path where we don't include 3.

(vi) Include 2

Three cases:

(2,2,2,1) ⇒ Profit = 16

(2,2,1,1,1) ⇒ Profit = 13

(2,1,1,1,1,1) ⇒ Profit = 10

finally, Traversing through the path where we don't include 2,

(vii) Include 1:

$(1,1,1,1,1,1,1) \Rightarrow$ Profit $= 7$

Hence, we get $R_7 = 18$ from 3 cases, $\therefore$ (a), (c) are True.

Q15) Consider the weights and values of items listed below. Note that there is only one unit of each item

Item Number	Weight (in Kgs)	Value (in rupees)
1	10	60
2	7	28
3	4	20
4	2	24

The task is to pick a subset of these items such that their total weight is no more than 11 kgs and their total value is maximum. Moreover, no item may be split. The total value of items picked by an optimal algorithm is denoted by V_{opt}. A greedy algorithm sorts the items by their value-to-weight ratios in descending order and packs them greedily, starting from the first item in the ordered list. The total value of items picked by the greedy Algorithm is denoted by V_{greedy}.

The value of $V_{opt} - V_{greedy}$ is ?

[GATE CSE 2018]

Solution:

$V_{opt} - V_{greedy} = 16$

As we can see, Either by Normal Intuition / Brute force / Dynamic Programming Approach, $V_{opt} = 60$

Now, solving for V_{greedy}:

Item Name	Weight (in kgs)	Value (in rupees)	Value/Weight
1	10	60	6
2	7	28	4
3	4	20	5
4	2	24	12

Sort them in descending order of Value/weight as per the question.

Item Name	Weight (in kgs)	Value (in rupees)	Value/weight
4	2	24	12
1	10	60	6
3	4	20	5
2	7	28	4

Now, Packing items, Given max. weight = 11:

Item 4 is picked ⇒ Remaining weight = 11 − 2 = 9

Item 1 cannot be picked, As combined weight exceeds 11.

∴ Item 3 is picked ⇒ Remaining Weight = 9 − 4 = 5

Now, Item 2 cannot be picked as 5 − 7 < 0.

∴ Items 4 and 3 are picked

i.e. $V_{greedy} = W_4 + W_3$

$= 24 + 20$

$= 44$

$\therefore V_{optimal} - V_{greedy} = 60 - 44$

$= \underline{\underline{16}}$

For obtaining $V_{optimal}$, we can use the recurrence:

$R_n = \max_{1 \le i \le n} (p[i] + R_{n-i})$ with base case $R_0 = 0$

Harith. Y
CS23I1027
B2 [CSE-DD]

Q16) An array of 25 distinct elements is to be sorted using QuickSort. Assume that the pivot element is chosen uniformly at random. The probability that the pivot element gets placed in the worst possible location in the first round of partitioning (rounded off to 2 decimal places) is ?

[GATE CSE 2019]

Solution:

0.08

The worst case scenario for QuickSort Algorithm is when the pivot element is either the smallest or the largest element of the array when chosen randomly. In such a case, the pivot does not split the array into reasonably balanced parts, leading to a skewed partition.

Given → Array with 25 distinct elements, the probability of selecting the smallest element as pivot is $1/25$ and the probability of selecting the largest element as pivot is $1/25$.

Since, these 2 events are mutually exclusive, the total probability is the sum of the above probabilities.,

i.e. $P(\text{worst case}) = P(\text{smallest}) + P(\text{largest})$

$= 1/25 + 1/25$

$= 2/25 = 8/100$

$= 0.08$

Q17) The Floyd-Warshall Algorithm for all-pair shortest paths computation is based on ?

(a) Greedy Paradigm
(b) Divide-and-Conquer Paradigm
(c) Dynamic Programming Paradigm
(d) None of the above

[GATE CSE 2016 SET-2]

Solution:

(c) Dynamic Programming Paradigm

Floyd Warshall Algorithm is a dynamic programming based Algorithm. It finds all-pairs shortest paths using following recursive nature of problems.

For every pair (i, j) of source and destination vertices respectively, there are 2 possible cases:

(i) k is not an intermediate vertex in shortest path from i to j. We keep the value of dist[i][j] as it is.

(ii) k is an intermediate vertex in shortest path from i to j. We update the value of dist[i][j] as dist[i][k] + dist[k][j].

This gives us a time complexity of $O(n^3)$. Whereas in case of Dense graphs, we get $O(v^3 \log v)$ by using Djikstra's Algorithm and $O(v^4)$ by using Bellman Ford's Algorithm.

Q18) Let A_1, A_2, A_3 & A_4 be 4 matrices of dimensions 10×5, 5×20, 20×10 and 10×5 respectively. The minimum no. of scalar multiplications required to find the product $A_1A_2A_3A_4$ using the basic matrix multiplication method is ?

[GATE CSE 2016 SET-2]

Solution:

1500

$A_{10\times5} \mid A_{5\times20} \mid A_{20\times10} \mid A_{10\times5}$

$P_0 \times P_1 \mid P_1 \times P_2 \mid P_2 \times P_3 \mid P_3 \times P_4$

Making the DP table:

(1,1) = 0	(2,2) = 0	(3,3) = 0	(4,4) = 0
(1,2) = 1000	(2,3) = 1000	(3,4) = 1000	×
(1,3) = 1500	(2,4) = 1250	×	×
(1,4) = 1500	×	×	×

Harith.Y
CS23I1027
B2 [CSE-DD]

Now, filling the table :

$$A_{13} = \min \begin{cases} A_{12} + A_{33} + 10\times20\times10 (\langle P_0, P_2, P_3 \rangle) = 3000 \\ A_{11} + A_{23} + 10\times5\times10 (\langle P_0, P_1, P_3 \rangle) = 1500 \end{cases}$$

$$A_{24} = \min \begin{cases} A_{23} + A_{44} + 5\times10\times5 (\langle P_1, P_3, P_4 \rangle) = 1250 \\ A_{22} + A_{34} + 5\times20\times5 (\langle P_1, P_2, P_4 \rangle) = 1500 \end{cases}$$

$$A_{14} = \min \begin{cases} A_{11} + A_{24} + 10\times5\times5 (\langle P_0, P_1, P_4 \rangle) = 1500 \\ A_{12} + A_{34} + 10\times20\times5 (\langle P_0, P_2, P_4 \rangle) = 3000 \\ A_{13} + A_{44} + 10\times10\times5 (\langle P_0, P_3, P_4 \rangle) = 2000 \end{cases}$$

Matrix Paranthesizing : $A_1((A_2A_3)A_4)$

$\therefore A_{14} = 1500$, i.e. min no. of scalar multiplications $= 1500$.

Q19) Consider product of 3 matrices M_1, M_2 and M_3 having w rows and x columns, x rows and y columns, and y rows and z columns respectively. Under what condition will it take less time to compute the product as $(M_1M_2)M_3$ than to compute $M_1(M_2M_3)$?

(a) Always take the same time

(b) $(1/x + 1/z) < (1/w + 1/y)$

(c) $x > y$

(d) $(w+x) > (y+z)$ [GATE CSE 2020]

Solution :

(b) $\left(\frac{1}{x} + \frac{1}{z}\right) < \left(\frac{1}{w} + \frac{1}{y}\right)$

$(M_1)_{w\times x}$, $(M_2)_{x\times y}$, $(M_3)_{y\times z}$

Cost of $(M_1M_2)M_3 = wxy + wyz$

Cost of $M_1(M_2M_3) = xyz + wxz$

Now, the time taken by $(M_1M_2)M_3$ will be less than $M_1(M_2M_3)$

when, $\omega xy + \omega yz < xyz + \omega xz$

$$\Rightarrow \frac{\omega xy}{\omega xyz} + \frac{\omega yz}{\omega xyz} < \frac{xyz}{\omega xyz} + \frac{\omega xz}{\omega xyz}$$

$$\Rightarrow \frac{1}{z} + \frac{1}{x} < \frac{1}{y} + \frac{1}{\omega}$$

$\therefore$ (b) $\rightarrow$ correct answer

Q20) Assume that multiplying a matrix G_1 of dimension $p \times q$ with another matrix G_2 of dimensions $q \times r$ requires pqr scalar multiplications. Computing the multiplications of n matrices $G_1G_2G_3 \ldots\ldots G_n$ can be done by paranthesizing in different ways. Define G_iG_{i+1} as an explicitly computed pair for a given paranthesization if they are directly multiplied. For example, in the matrix multiplication chain $G_1G_2G_3G_4G_5G_6$ using Paranthesization $(G_1(G_2G_3))(G_4(G_5G_6))$, G_2G_3 and G_5G_6 are the only explicitly computed pairs.

Consider a matrix multiplication chain $F_1F_2F_3F_4F_5$, where matrices F_1, F_2, F_3, F_4 and F_5 are of dimensions 2×25, 25×3, 3×16, 16×1, 1×1000 respectively. In the paranthesization of $F_1F_2F_3F_4F_5$ that minimizes the total number of scalar multiplications, the explicitly computed pairs ~~are~~ is/are

(a) F_1F_2 and F_3F_4 only

(b) F_2F_3 only

(c) F_3F_4 only

(d) F_1F_2 and F_4F_5 only

[GATE CSE 2018]

Solution

(c) F_3F_4 only

Harith - Y
CS23I1027
B2[CSE - DD]

$$m[i,j] = \begin{cases} 0 & , \text{ if } i=j \\ \min_{i \le k < j} \{m[i,k] + m[k+1,j] + P_{i-1} P_k P_j & , \text{ if } i<j \end{cases}$$

I^st part:

$m[1,2] = m[1,1] + m[2,2] + P_0 P_1 P_2$

$= 0 + 0 + 2 \times 25 \times 3$

$= 150$

$m[2,3] = P_1 P_2 P_3$

$= 25 \times 3 \times 16$

$= 1200$

$m[3,4] = P_2 P_3 P_4$

$= 3 \times 16 \times 1$

$= 48 \iff$ min

$m[4,5] = P_3 P_4 P_5$

$= 16 \times 1 \times 1000$

$= 16000$

II^nd Part:

In $m[1,3]$, $m[1,2]$ was minimum, so

$m[1,3] = m[1,2] + m[3,3] + P_0 P_2 P_3$

$= 150 + 2 \times 3 \times 16$

$= 246$

So minimum is $m[3,4]$, Hence

$m[2,4] = m[2,2] + m[3,4] + P_1 P_2 P_4$

$= 48 + 25 \times 3 \times 1$

$= 123 \iff$ min

In $m[3,5]$, $m[3,4]$ is already minimum,

$\therefore m[3,5] = m[3,4] + m[5,5] + P_2 P_4 P_5$

$= 48 + 3 \times 1 \times 1000$

$= 3048$

III^rd Part:

In Previous part, $m[2,4]$ is minimum

$m[1,4] = m[1,1] + m[2,4] + P_0 P_1 P_4$

$= 123 + 2 \times 25 \times 1$

$= 173 \iff$ min

$m[2,4]$ is minimum, so

$m[2,5] = m[2,4] + m[5,5] + P_1 P_4 P_5$

$= 123 + 25 \times 1 \times 1000$

$= 25123$

IV^th Part:

$m[1,4]$ is minimum in previous part, so

$m[1,5] = m[1,4] + m[5,5] + P_0 P_4 P_5$

$= 173 + 2 \times 1 \times 1000$

$= 2173$

Now, we can see that:

$(F_1 F_2 P_3 F_4 F_5) \Rightarrow ((F_1 F_2 F_3 F_4) F_5) \Rightarrow ((F_1 (F_2 F_3 F_4)) F_5)$

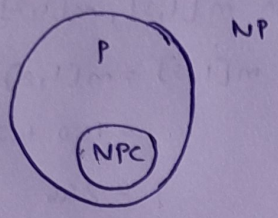
$\Downarrow$

$((F_1 (F_2 (F_3 F_4))) F_5)$

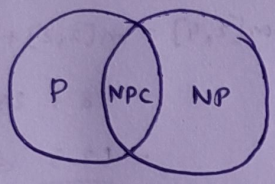
$\therefore F_3 F_4 \rightarrow$ Explicitly computed Pair, (c) $\rightarrow$ True

Q21) Suppose a polynomial time Algorithm is discovered that correctly computes the largest clique in given graph. In this scenario, which one of the following represents the correct Venn Diagram of the complexity classes P, NP and NP complete (NPC)

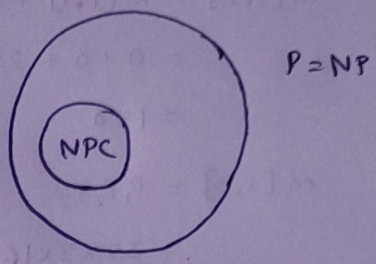
(a)



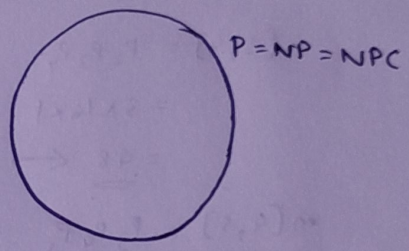
(b)



(c)



(d)



[GATE CSE 2014 SET-I]

Solution:

(C)

P is not a subset of NPC & this is unconditionally true. Since, if P=NP, every NP-C problem is solvable in polynomial time, hence, NPC is a subset of P.

For P=NPC, we need to prove that P is a subset of NPC, i.e. an NP-C problem can be reduced to every deterministic polynomial time problem, i.e.

$$A \leq B, \forall B \in P, A \in NP\text{-}C$$

Proving the above statement would mean that every problem in P is atleast as hard as every other problem in P and NP ($\because P=NP$). Hence, by definition, every problem in P is an NP-C problem.

For the sake of proof, let's consider an example,

Harith.Y
CS23I1027
B2 [CSE-DD]

Pick A to be Hamiltonian Path Problem. For a given graph, I solve the problem in Polynomial time

i.e. 'Yes' output for A ⇒ 'Yes' output to a 'P' problem,

(Ex. an even input for the P Problem "Is the Input no. even?")

and 'No' output for A ⇒ 'No' output to a 'P' problem.

∴ This effectively means that a reduction that reduces an NP-C problem to every other P problem exists, except for only 2 languages in P given below:

(i) Language that accepts everything (has no "no" answer)

(ii) Language that accepts nothing (has no "yes" answer)

This tells, $P - \{\Sigma^*, \emptyset\}$ is a subset of NPC

∴ P = NPC cannot be followed.

The conclusion is,

$$P - \{\Sigma^*, \emptyset\} = NPC \quad (c)$$

(d) is correct iff we exclude Σ^* and $\emptyset$ from P.

Q22) Consider two decision problems Q_1, Q_2 such that Q_1 reduces in polynomial time to 3-SAT and 3-SAT reduces in polynomial time to Q_2. Then which of the following is consistent with the above statement?

(a) Q_1 is NP, Q_2 is NP-Hard.

(b) Q_2 is NP, Q_1 is NP-Hard.

(c) Both Q_1 and Q_2 are NP.

(d) Both Q_1 and Q_2 are NP-Hard.

[GATE CSE 2015 SET-2]

Solution:

(a) $Q_1 \rightarrow$ NP, $Q_2 \rightarrow$ NP-Hard

3-SAT is NP-Complete

∴ 3-SAT is in NP as well as NP-Hard.

Now, any less or equally hard problem can be reduced, in polynomial time, to 3-SAT.

⇒ Q_1 reducing to 3-SAT means Q_1 is less harder than 3-SAT

⇒ Q_1 can be P or NP.

∵ $P \subseteq NP \Rightarrow Q_1$ is NP, need not be NP-Hard.

Now, 3-SAT reducing, in polynomial time, to Q_2 means that Q_2 is harder or as hard as 3-SAT meaning Q_2 is also NP-Hard.

⇒ Q_2 need not be in NP.

∴ (a) → Correct.

Q23) Language L_1 is polynomial time reducable to Language L_2. Language L_3 is polynomial time reducible to language L_2, which in turn is polynomial time reducible to Language L_4. Which of the following is/are true?

I. if $L_4 \in P$, then $L_2 \in P$

II. if $L_1 \in P$ or $L_3 \in P$, then $L_2 \in P$

III. $L_1 \in P$, if and only if $L_3 \in P$

IV. If $L_4 \in P$, then $L_3 \in P$

(a) II only

(b) III only

(c) I and IV only

(d) I only

[GATE CSE 2015 SET-3]

Solution:

(c) I and IV only

(i) L_4 is polynomial time reducible to L_2.

$\Rightarrow$ L_2 is atleast as hard as L_4

(ii) L_3 is polynomial time reducible to L_2

$\Rightarrow$ L_2 is atleast as hard as L_3.

(iii) L_2 is polynomial time reducible to L_4.

$\Rightarrow$ L_4 is atleast as hard as L_2.

If L_4 is in P $\Rightarrow$ L_3, L_2 and L_4 must also be in P.

Hence I and IV are true.

We can have L_4 in P and L_2 not in P and none of the given conditions are violated. $\therefore$ II is false.

Assume L_3 not in P. Since L_2 must be as hard as L_3, it must also not be in P. But L_1 is less harder than L_2 as per condition 1, and it can be in P without violating any given conditions. $\therefore$ III is false.

$\therefore$ (C) $\rightarrow$ True.

Q24) Consider the following decision problem 2CNFSAT defined as follows:

$\{\phi \mid \phi$ is a satisfiable propositional formula in CNF with atmost two literals per clause $\}$

For example, $\phi = (x1 \vee x2) \wedge (x1 \vee \overline{x3}) \wedge (x2 \vee x4)$ is a boolean formula and it is [redacted], in 2CNFSAT.

The decision problem 2CNFSAT is

(a) NP- Complete

(b) Solvable in Polynomial time by reduction to directed graph ~~reachability~~ reachability.

(c) Solvable in constant time since any input instance is satisfiable.

(d) NP-Hard but not NP-complete

Solution: [GATE CSE 2014 SET-3]

(b) Solvable in Polytime by reduction

As we know, 2CNF is P.

∴ (a) and (d) are ruled out.

Satisfiability means that you can have an assignment of values which make the expression true which you can clearly do for the given ~~expres~~ expression.

(c) → False, because you could have an expression like $(A \wedge \bar{A})$ which always evaluates to false and is thus not satisfiable.

∴ option (b) → True.

Q25) Huffman tree is constructed for the following data : {A, B, C, D, E} with frequency {0.17, 0.11, 0.24, 0.33, 0.15} respectively.

100 00 010 101 is decoded as

(a) BACE

(b) CADE

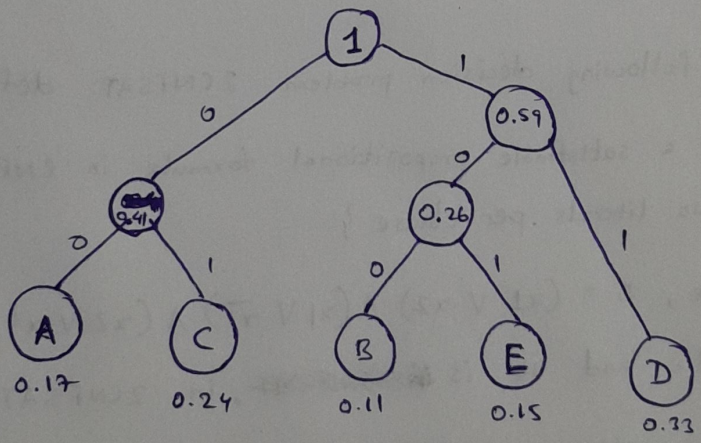
(c) BAD

(d) CADD

[GATE CSE 2020]

Solution:

(a) BACE



∴ A : 00

B : 100

C : 01

D : 11

E : 101

100 00 01 101

B A C E ⟹ BACE (a)

Harith. Y
cs23I1027
132 [CSE-DD]

Q26) A message is made up entirely of characters from the set $X = \{P, \varphi, R, S, T\}$. The table of probabilities for each of the characters is shown below:

Character	Probability
P	0.22
φ	0.34
R	0.17
S	0.19
T	0.08
Total	1.00

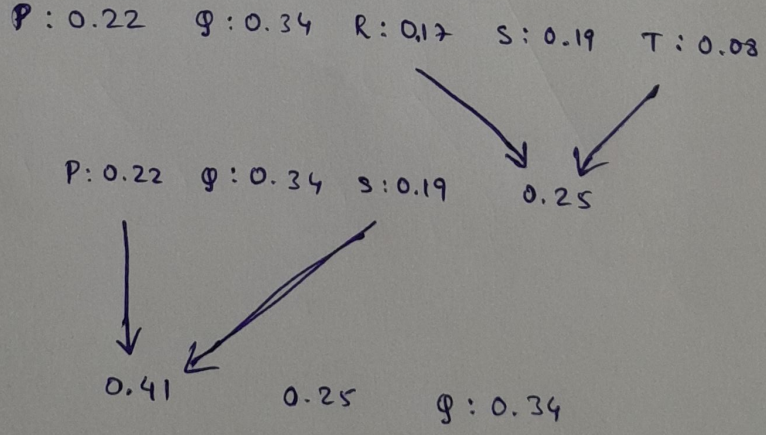
If a message of 100 characters over X is encoded using Huffman coding, then the expected length of the encoded message in bits is ?

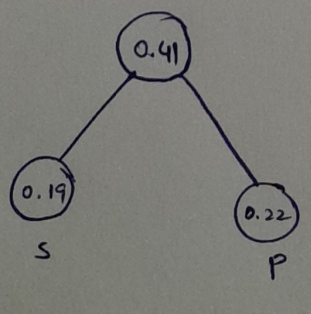
[GATE CSE 2017 SET-2]

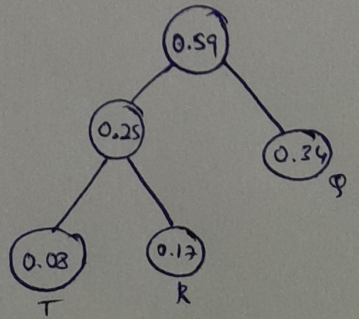
Solution

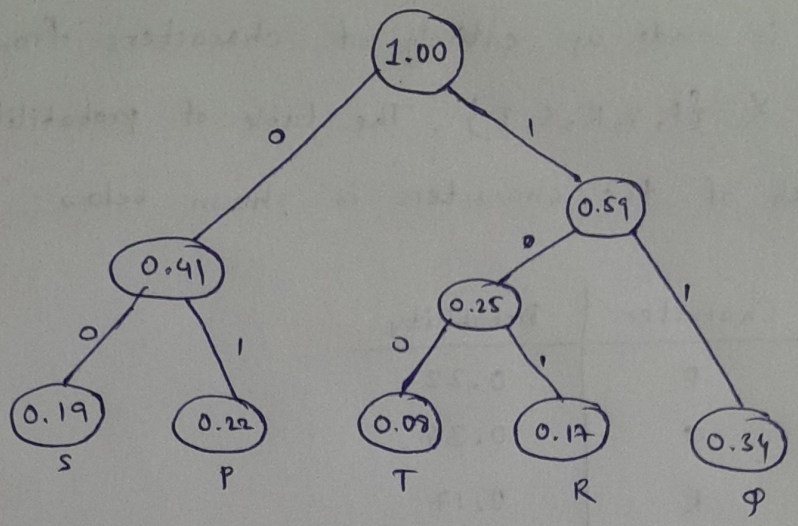
225

$X = \{P, \varphi, R, S, T\}$









∴ P : 01
φ : 11
R : 101
S : 00
T : 100

Expected length of an encoded character

$= (0.22 \times 2) + (0.34 \times 2) + (0.17 \times 3) + (0.19 \times 2) + (0.08 \times 3)$ bits

$= 0.44 + 0.68 + 0.51 + 0.38 + 0.24$ bits

$= 2.25$ bits = Sum of Internal nodes $(1 + 0.41 + 0.59 + 0.25)$
(Total Probability)

∴ Expected length of an encoded message of 100 characters in bits $= 2.25 \times 100 = 225$

Q27) The Master's Theorem

(a) assumes the subproblems are unequal size

(b) can be used if the subproblems are of equal size

(c) Cannot be used for Divide and conquer algorithms

(d) Cannot be used for Asymptotic complexity Analysis

Solution

(b) [GATE CSE 2020]

It divides the subproblem into each of size '$\frac{n}{b}$'.

∵ According to Master's Theorem, The recurrance relation is

$$T(n) = a \times T\left(\frac{n}{b}\right) + n^k \cdot \log^p n$$

Q28) The running time of an algorithm is represented by the following recurrence relation:

$$T(n) = \begin{cases} n & , n \leq 3 \\ T(n/3) + cn & \text{otherwise} \end{cases}$$

Which one of the following represents the time complexity of the algorithm?

(a) $\theta(n)$

(b) $\theta(n \log n)$

(c) $\theta(n^2)$

(d) $\theta(n^2 \log n)$

[GATE CSE 2009]

Solution:

(a) $\theta(n)$

From Master's Theorem,

$$T(n) \xleftarrow{\text{Monotonically increasing}} \begin{cases} \theta(n^d) & , \text{if } a < b^d \\ \theta(n^d \log n) & , \text{if } a = b^d \\ \theta(n^{\log_b a}) & , \text{if } a > b^d \end{cases}$$

where $a \geq 1$, $b \geq 2$, $c > 0$, $f(n)$ where $d \geq 0$

↳ $T(n) = aT(n/b) + f(n)$ & $T(1) = c$

Following case 1 of Master's Theorem:

↳ $f(n) = \theta(n^d)$

$a = 1$, $b = 3$, $d = 1$

$\because a < b^d \quad [1 < 3]$

$\Rightarrow T(n) = \theta(n^d) = \theta(n^1) = \theta(n)$

$\therefore T(n) = \theta(n)$ (a)

Q29) Let $F(n)$ denote the maximum number of comparisons made while searching for an entry in a sorted array of size 'n' using binary search. Which one of the following options is true?

(a) $F(n) = F(\lfloor n/2 \rfloor) + 1$

(b) $F(n) = F(\lfloor n/2 \rfloor) + F(\lceil n/2 \rceil)$

(c) $F(n) = F(\lfloor n/2 \rfloor)$

(d) $F(n) = F(n-1) + 1$ [GATE DS & AI 2024]

Solution:

(a) $F(n) = F(\lfloor n/2 \rfloor) + 1$

In binary search, at each step, you compare the target element with the middle element of the array. Based on the comparison result, you recursively search either the left or right half of the array.

Let's denote $F(n)$ as the maximum number of comparisons made while searching for an entry in a sorted array of size 'n' using binary search.

For an array of size 'n',

- At each step, 1 comparison is made to check if the middle element is equal to the target element.
- Then, you either search the left half or the right half of the array

∴ Max. no. of comparisons is $F(n) = 1 + F(\lfloor n/2 \rfloor)$ (a)

Q30) What is the worst-case number of arithmetic operations performed by recursive binary search on a sorted array of size 'n'?

(a) $\theta(\sqrt{n})$

(b) $\theta(\log_2 n)$

(c) $\theta(n^2)$

(d) $\theta(n)$ [GATE CSE 2021 SET-2]

Solution:

(b) $\theta(\log_2 n)$

Worst-case number of arithmetic operations performed by recursive binary search on a sorted array is given by the following recurrence relation:

$T(n) = T(n/2) + \theta(1)$

For $T(n) = aT(n/b) + f(n)$, $a \geq 1$, $b > 1$

Case 2 of Master's Theorem says that

if $f(n) = \theta(n^{\log_b a})$, Then $T(n) = \theta(n^{\log_b a} \cdot \log n)$

$\Rightarrow$ Here, $n^{\log_b a} = n^{\log_2 1} = n^0 = 1 \Rightarrow f(n) = \theta(n^{\log_b a})$

Hence, we can say that $T(n) = \theta(1 \cdot \log_2 n) = \theta(\log_2 n)$

$\therefore$ (b) $\longrightarrow$ True

Q31) We are given 9 tasks $T_1, T_2, \ldots T_9$. The execution of each task requires one unit of time. We can execute one unit at a time. Each task has a Profit P_i and a deadline d_i. Profit P_i is earned if the task is completed before the end of the d_i^{th} unit of time.

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Profit	15	20	30	18	18	10	23	16	25
Deadline	7	2	5	3	4	5	2	7	3

Are all tasks completed in the schedule that gives maximum profit ?

(a) All tasks are completed

(b) T_1 and T_6 are left out

(c) T_1 and T_8 are left out

(d) T_4 and T_6 are left out

[GATE CSE 2005]

Solution :

(d) T_4 and T_6 are left out

As Given, "Each task requires one unit of time", this shows that we can greedily choose the better task and that should give us the optimal solution. The best task would be the one

with maximum profit. Hence, we can sort the tasks based on deadlines and then profit as follows:

Task	T_7	T_2	T_9	T_4	T_5	T_3	T_6	T_8	T_1
Deadline	2	2	3	3	4	5	5	7	7

$0 \xrightarrow{T_7} 1 \xrightarrow{T_2} 2 \xrightarrow{T_9} 3 \xrightarrow{T_5} 4 \xrightarrow{T_3} 5 \xrightarrow{T_8} 6 \xrightarrow{T_1} 7$

$\therefore T_4$ and T_6 are left out (d)

And the maximum profit will not include those of T_4 and T_6, and will be $15+20+30+18+16+23+25 = \underline{147}$

Q32) Let G be an undirected graph. Consider a depth-first traversal of G, let T be the resulting depth-first search tree. Let u be a vertex in G and let v be the first new (unvisited) vertex visited after visiting u in the traversal. Which of the following statement is always true?

(a) $\{u, v\}$ must be an edge in G, and u is a descedent of v in T.

(b) $\{u, v\}$ must be an edge in G, and v is a descendant of u in T.

(c) If $\{u, v\}$ is not an edge in G, then u is a leaf in T.

(d) If $\{u, v\}$ is not an edge in G, then u and v must have the same parent in T.

[GATE CSE 2000]

Solution:

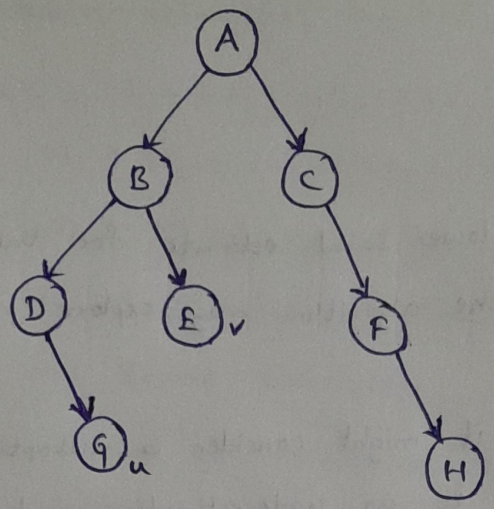
(c) $\{u, v\}$ is an edge in $G \Rightarrow u$ is a leaf in T.

Let us make a DFS order of a tree using vertices A, B, C, D, E, F, G and H and

$u = G$

$v = E$

Harith • Y
CS23I1027
B2 [CSE-DD]



Let this be the DFS order of the tree.

We can conclude that

(i) It is not necessary that there is an edge between them.

∴ (a) and (b) → False

(ii) If there is no edge, then u must be a leaf, i.e. G is a leaf node here.

∴ (c) → True

(iii) It is not always possible that u and v have same parent. But ~~they~~ they have same ancestor

∴ (d) → False

Only when we hit a dead-end, we backtrack to another unvisited vertex, Hence (c) → True.

Q33) Which of the following statements is true for Branch-and-Bound Search?

(a) Underestimates of remaining distance may cause deviation from optimal path.

(b) Overestimates cannot cause right path to be overlooked.

(c) Dynamic Programming principle can be used to discard redundant partial paths.

(d) ~~Above~~ All of the above

[UGC NET CSE 2015 PART-3]

Solution :

(d) All of the above

Option (a) :

In BnB, if the lower bound estimate for the cost to the goal is too low, the algorithm may explore paths that are not optional.

This is because it might consider a suboptimal path to be promising due to an underestimating cost.

Hence, it may not follow the true optimal path initially, leading to inefficiencies.

∴ (a) ⟶ True

Option (b) :

BnB maintains a global bound (best solution found so far), overestimations may delay exploring the optimal path, but they won't cause it to be permenantly overlooked.

This is different from greedy search, where an overestimate could prevent exploration.

∴ (b) ⟶ True

option (c) :

If the same subproblem (partial path) is encountered multiple times, BnB can use the principle of Dynamic Programming to avoid re-solving the same subproblem.

By storing previously computed costs for subproblems, redundant paths can be discarded.

This avoids re-exploring ~~patts~~ parts of the search space and speeds up the process.

∴ (c) ⟶ True

∴ All Statements, (a), (b) & (c) are True :

∴ (d) → Answer

Harith. Y
CS23I1027
B2 [CSE-DD]

Q34) Consider the following instance of the 0-1 Knapsack problem:

- maximise $6X_1 + 11X_2 + 16X_3 + 21X_4 + 26X_5$
- Subject to $4X_1 + 8X_2 + 12X_3 + 16X_4 + 20X_5 \leq 32$
- and $X_i = 0$ (or) $1 \quad \forall i \in \{1,2,3,4,5\}$

It is required to find all the optimal solutions to this instance using Branch and Bound technique.

(i) State what method you would use to compute bounds on the partial solutions.

(ii) Using a suitable branching technique, Generate the entire search tree for this instance of the problem and find all optimal solutions. Number the nodes in the tree in the order in which they are expanded; and for each node, show the bound on the partial solutions and the decision which leads to that node.

Solution: [GATE CSE 1990]

Items, $i = \{1, 2, 3, 4, 5\}$

Values $= \{6, 11, 16, 21, 26\}$

Weights $= \{4, 8, 12, 16, 20\}$

Capacity, $C = 32$

Calculating Upper Bound:

Value/weight Ratios $= \{1.5, 1.375, 1.333, 1.3125, 1.3\}$

Filling Knapsack greedily:

Taking items 1, 2 & 3 $\Rightarrow 4+8+12 = 24$

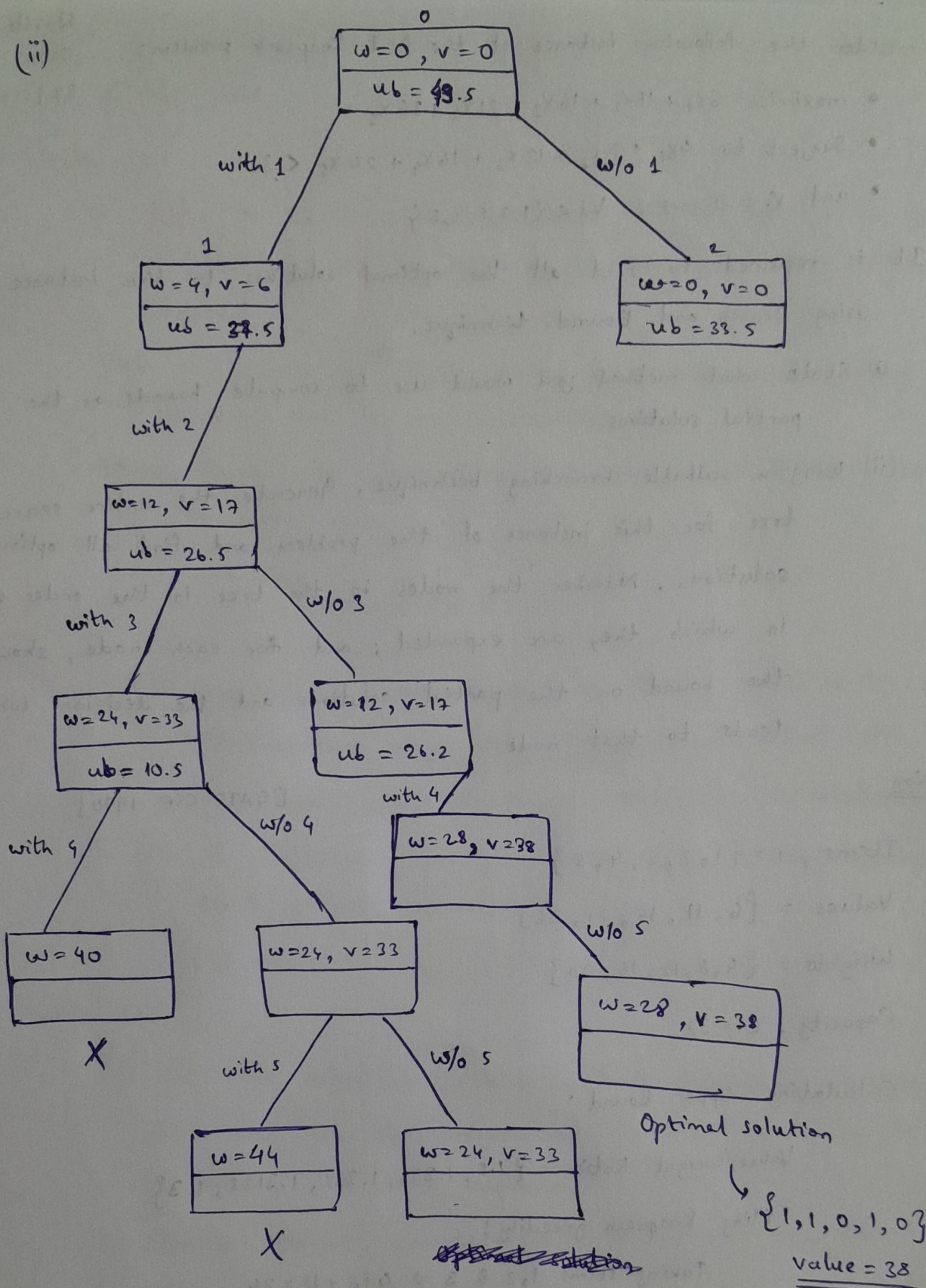
Take $\frac{8}{16} = 0.5$ of item 4 to reach capacity 32

$\therefore$ Upper bound $= 6 + 11 + 16 + (0.5 \times 21) \Rightarrow$

$= 43.5$

Let's construct the entire search tree.

(ii)



(i) Method used : Greedy Fractional Knapsack (Relaxed Knapsack)

(I) Relax the constraint $x_i \in \{0, 1\}$ to $x_i \in [0, 1]$

(II) Calculate Upper Bound by solving relaxed version of the problem.

(III) Upper Bound = Current value + max. potential Value

Harith. Y
CS23I1027
B2 [CSE-DD]

Q35) Which data structure is most suitable for implementing best first Branch and Bound strategy?

(a) Stack
(b) Queue
(c) Priority Queue
(d) Linked List

Solution:

(c) Priority Queue

Priority Queue is the data structure used for implementing best first Branch and Bound strategy. Djikstra's algorithm is an example of best first search algorithm.

Q36) Which of the following is not a Branch and Bound strategy to generate branches?

(a) LIFO Branch and Bound
(b) FIFO Branch and Bound
(c) Lowest Cost Branch and Bound
(d) Highest Cost Branch and Bound

Solution:

(d) Highest Cost BnB

LIFO, FIFO and Lowest Cost Branch and Bound are different strategies to generate branches.

Lowest Cost Branch and Bound helps us find the lowest cost path.

LIFO BnB uses a Last-In, First-Out stack to manage unexplored branches, being a DFS Approach.

FIFO BnB uses a first-In, first-Out queue for branch exploration, following a BFS approach.

Lowest Cost BnB expands the branch with the lowest cost first, often using a priority queue for efficient cost-based selection.

Q37) Given an undirected Graph, to determine whether the graph contains a Hamiltonian cycle or not, which approach would be best suitable?

(a) Backtracking
(b) Greedy ~~approach~~ method
(c) Dynamic Programming
(d) Divide and Conquer

[GATE CSE 2015 SET-1]

Solution:

(a) Backtracking

The Hamiltonian Circuit problem involves finding a path in a graph that visits every vertex exactly once and returns to the starting vertex. Using backtracking is an effective approach because it explores all possible solutions and backtracks whenever a partial solution cannot be completed.

Backtracking systematically builds the solution incrementally and abandons a path as soon as it determines that the current path cannot lead to a valid Hamiltonian Circuit. This allows for exploring all possibilities in an efficient manner without redundant checks.

Approach:

(i) Choice: Add a vertex to the current path
(ii) Constraint: Ensure the vertex is not already visited and is connected to the previously added vertex.
(iii) Goal: All vertices are visited, and the last vertex is connected to the starting vertex.

Q38) Which of the following problems cannot be solved by Backtracking?

(a) N-Queen Problem
(b) Subset Sum Problem
(c) Hamiltonian Circuit Problem
(d) Travelling Salesman Problem

Harith. Y
CS23I1027
B2 [CSE-DD]

Solution:

(d) TSP

N-Queen Problem: Can be solved using backtracking by placing queens one by one in valid positions and backtracking if conflict arises.

Subset Sum Problem: Can be solved by backtracking via exploring all subsets of numbers and checking if the sum matches the target.

Hamiltonion Circuit Problem: Can be solved by backtracking while ~~exploring~~ incrementally constructing a path and backtracking when a valid circuit cannot be completed.

Traveling Salesman Problem:

Although Backtracking can be used theoretically, it is highly inefficient for TSP because it involves exploring all permutations of cities.

For TSP, dynamic programing (e.g. using Held-Karp Algorithm) or approximation algorithms are more efficient.

Therefore, TSP is generally not solved using backtracking in practice

∴ (d) → TSP.

9) Backtracking algorithm is implemented by constructing a tree of choices called as ?

(a) State-space tree
(b) State-chart tree
(c) Node tree
(d) Backtracking tree

Solution:

(a) State-Space tree

In a Backtracking algorithm, the solution process is represented as a state-space tree, where

- Each node represents a partial solution
- Each branch represents a choice made to reach the next state.
- The algorithm explores the tree to find a valid solution, backtracking when a node leads to an invalid/non-promising state.

Other options like state-chart tree, node tree and backtracking tree are not standard terminologies associated with backtracking.

Q40) An algorithm to find the length of the longest monotonically increasing sequence of numbers in an array $A[0:n-1]$ is given below:

Let L_i denote the length of the longest monotonically increasing sequence starting at index i in the array.

Initialize $L_{n-1} = 1$

For all i such that $0 \leq i \leq n-2$,

$$L_i = \begin{cases} 1 + L_{i+1}, & \text{if } A[i] < A[i+1] \\ 1, & \text{otherwise} \end{cases}$$

Finally, the length of the longest monotonically increasing sequence is $\max(L_0, L_1, \ldots, L_{n-1})$.

Which of the following statements is TRUE?

(a) The algorithm uses dynamic programming paradigm

(b) The algorithm has a linear complexity and uses Branch and Bound paradigm

(c) The algorithm has a non-linear polynomial complexity and uses Branch and Bound Paradigm.

(d) The algorithm uses divide and conquer Paradigm.

[GATE CSE 2011]

Solution:

(a) Dynamic Programming ~~Approx~~ Paradigm

~~The~~

The algorithm is storing the optimal solutions to subproblems at each point (for each i), and then using it to derive the optimal solution of a bigger problem. This is dynamic

programming approach. And the program has linear time complexity.

Now, Branch and Bound comes when we explore all possible solutions (branch) and we backtrack as soon as we realise we won't get a result.

In backtracking, In each step, you check if this step satisfies all the conditions. If it does, you continue generating subsequent solutions. If not, you go one step backward to check for another path. ~~So backtracking~~

So backtracking gives all possible solutions while Branch and Bound will give only the optimal one.

The given algorithm here is neither Backtracking nor Branch and Bound, because we are not branching anywhere in the solution space.

And the algorithm is not divide and Conquer as we are not dividing the problem and then merging the solution as in case of merge sort.

→ Summary:

Asymptotic Notations - Q1, 2, 3, 4

Masters Theorem - Q5, 27, 28

Sorting - Q6, 7, 16

Searching - Q8, 9, 29, 30

Graph - Q10, 11, 13, 32

Huffman Coding - Q12, 25, 26

Greedy - Q14, 15, 31

Dynamic Programming - Q17, 18, 19, 20

P, NP, NP-C, NP-H - Q21, 22, 23, 24

Branch and Bound - Q33, 34, 35, 36

Backtracking - Q37, 38, 39, 40

Q2

(1) Graph Colouring Problem:

Given a Graph $G = (V, E)$ and an integer k, determine whether it is possible to assign k colours to the vertices such that no two adjacent vertices share the same colour

Analysis:

(i) Belongs to NP:

A proposed solution (colouring of the vertices) can be verified in polynomial time by checking all edges in the graph to ensure no two adjacent vertices share the same colour. Verification requires $O(|E|)$ time, where $|E|$ is the no. of edges in the graph.

(ii) NP-Complete for $k \geq 3$:

The graph colouring problem is known to be NP-Complete when $k \geq 3$. This can be proven by polynomial time reduction to 3-SAT problem, a well known NP-Complete problem for $k = 2$.

The problem reduces to checking if graph is bipartite, which is solvable in Polynomial time using BFS or DFS.

(iii) Conclusion:

For $k = 2$, the problem belongs to P.
For $k = 3$, the problem belongs to NP-Complete.

(2) 0-1 Integer Programming Problem:

Let A be any $p \times q$ matrix with integer coefficients and let $b \in Z^p$ be any vector with integer coefficients

The 0/1 Integer programming problem is to find whether a system of p linear equations with q variables with

Harith. Y
CS23I1027
B2 [CSE-DD]

$a_{ij}, b_i \in Z$ has any solution.

$x \in \{0,1\}^q$, that is with $x_i \in \{0,1\}$

i.e $a_{11}x_1 + a_{12}x_2 + \cdots\cdots + a_{1q}x_q = b_1$

$a_{21}x_1 + a_{22}x_2 + \cdots\cdots + a_{2q}x_q = b_2$

$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$

$a_{p1}x_1 + a_{p2}x_2 + \cdots\cdots + a_{pq}x_q = b_p$

In Matrix form, if we let

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots\cdots & a_{1q} \\ a_{21} & a_{22} & \cdots\cdots & a_{2q} \\ \vdots & \vdots & \vdots & \vdots \\ a_{p1} & a_{p2} & & a_{pq} \end{bmatrix},$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix} \quad \text{and} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{bmatrix}$$

Then, we write above system as

$$Ax = b$$

It is immediate that 0/1 integer programming problem is in NP.

To prove that it is NP-Complete, we reduce the bound tiling problem to it.

This means that we provide a method running in polynomial time that converts every instance of bound tiling problem to an instance of 0/1 integer programming problem

such that the first problem has a solution iff the converted problem has a solution.

—— X ——