GATE

MADE EASY Workbook 2026

Detailed Explanations of Try Yourself *Questions*

Computer Science & IT

Algorithms

MADE EASY PUBLICATIONS

1 Fundamental of Algorithms

Detailed Explanation of Try Yourself Questions

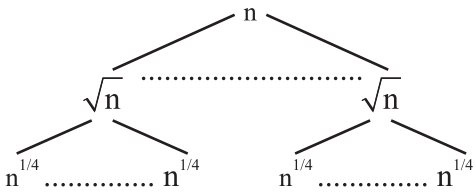
T1 : Solution

(d)

The given function is recursive so the equivalent recursion equation is

$$T(n) = 1; \quad n \le 2$$

$$T(n) = \left\lfloor \sqrt{n} \right\rfloor + n; \; n > 2$$



All the level sums are equal to n. The problem size at level k of the recursion tree is n^{2-k} and we stop recursing when this value is a constant. Setting $n^{2-k} = 2$ and solving for k gives us

$$2^{-k} \log_2 n = 1$$

$$\Rightarrow \quad 2^k = \log_2 n$$

$$\Rightarrow \quad k = \log_2 \log_2 n$$

So $$T(n) = \Theta(\log_2 \log_2 n)$$

T2 : Solution

(c)

The given code is

```
1.      Counter = 0;
2.  for (i = 1; i < = n; i ++)
3.  {   if (A[i] = = 1) counter ++;
4.      else {f(counter); counter = 0;}
5.  }
```

The time complexity of the program fragment depends on the frequency (Number of steps) of line 3 and 4. In line 4 the frequency depends on the variable counter and there is no increment in the counter variable which is initialize to 0, so $f(0)$ then counter = 0 it means there is no cell in an array which having a bit 0, so all cells in the array contains 1. Consider the line 3 if (A[i] = = 1) counter ++; the value of i will be increases upto n so the value of counter will be n. Since n is the frequency of the line 3 and the frequency of line 4 is 0. So the time complexity of line 3 is $O(n)$ on average n and $f(0) = O(1)$ is the time complexity of line 4. So the time complexity of the program fragment is maximum of line 3 and 4 which is $O(n)$ on average.

T3 : Solution

(a, d)

Function 1

```
while n > 1 do
    for i = 1 to n do
        x = x + 1;
    end for
    n = ⌊n/2⌋;
end while
```

$f_1(n)$: Number of times $x = x + 1$ executes

$$f_1(n) = n + \frac{n}{2} + \frac{n}{2^2} + ... + \frac{n}{2^k}$$

where $\frac{n}{2^k} > 1$

$k \approx \log_2 n$

$$f_1(n) = n\left[1 + \frac{1}{2} + \frac{1}{2^2} + ... + \frac{1}{2^k}\right] = \theta(n)$$

Function 2

```
for i = 1 to 100 * n do
    x = x + 1;
end for
```

$f_2(n)$: Number of times $x = x + 1$ executes

$f_2(n) = 100 * n = \theta(n)$

$\therefore$ $f_1(n) = \theta(f_2(n))$

$f_1(n) = O(n)$ are correct

T4 : Solution

(b)

According to Standard Master Theorem, only option (b) is correct.

T5 : Solution

(b)

$$T(n) = \begin{cases} T(n^{1/a}) + 1, & \text{when } n \neq b \\ 1, & \text{when } n = b \end{cases}$$

$$T(n) = T(n^{1/a}) + 1$$
$$= T(n^{1/a^2}) + 1 + 1 \quad [\because T(n^{1/a}) = T(n^{1/a^2}) + 1]$$
$$= T(n^{1/a^3}) + 1 + 1 + 1 \quad [\because T(n^{1/a^2}) = T(n^{1/a^3}) + 1]$$

After k interactions

$$= T(n^{1/a^k}) + k$$

when, $n^{1/a^k} = b$, i.e. $\frac{1}{a^k}\log n = \log b$

$$\Rightarrow \quad a^k = \frac{\log n}{\log b} = \log_b n$$

$$\Rightarrow \quad k = \log_a \log_b n$$

[$\because$ a and b are $w(1)$, so a and b are some function of n and not constant. So a and b can't be replaced with 2]

Now,

$$T(n) = T(n^{1/a^k}) + k$$
$$= T(b) + \log_a \log_b n$$
$$= 1 + \log_a \log_b n$$
$$= \theta(\log_a \log_b n)$$

Hence, option (b) is the answer.

■■■■

2 Divide and Conquer

Detailed Explanation of Try Yourself Questions

T1 : Solution

(b)

n unsorted arrays is given where n is odd.
Each array contain n elements so finding median will take $O(n)$.
Total n array are given so time take

$$n \times O(n) = O(n^2)$$

Medians of array $A_1, A_2 \ldots\ldots, A_n$ has n element M_i where $M_i = \sum_{i=1}^{n} M_i$, to find median of M_i time taken is $O(n)$.

Worst case time complexity of computing the median of the medians

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

T2 : Solution

(a)

Using divide and conquer min-max algo, WC number of comparisons = $\frac{3n}{2} - 2$.

Hence $\frac{3n}{2} - 2$ will lie in the range of n to $3\left\lceil \frac{n}{2} \right\rceil$.

Hence option (a) is the correct answer.

T3 : Solution

(0.08)

Worst cases in quick sort when array in divided into 1 : $n-1$ ratio.
Total 25 element in the array and worst case when pivot element is maximum or minimum.
So total 2 cases for worst case

$$\text{Probability} = \frac{\text{Favourable cases}}{\text{Total cases}} = \frac{2}{25} = 0.08$$

T4 : Solution

(b)

For $n = 64$

$$c_1.n \log n = 30 \text{ sec}$$
$$c_1.64.\log 64 = 30 \text{ sec}$$
$$c_1 = \frac{5}{64} \quad \ldots(1)$$

Now, $c_1.n \log n = 6 \times 60 \text{ sec}$

$$n \log n = \frac{6 \times 60 \text{ sec}}{c_1 \text{ sec}}$$
$$n \log n = \frac{6 \times 60}{5} \times 64$$
$$n \log n = 4608$$
$$n \log n = 512 \times 9 \Rightarrow n = 512$$

T5 : Solution

(b)

Given program is implementation of binary search. x is used as key, K is in middle position of an array/ subarray, i and j positions may change based on availability of key with comparison to listA[K] element.

■■■■

3 Tree, Graphs, Binary Heap

Detailed Explanation of Try Yourself Questions

T1 : Solution

O(E + V)

In adjacency list representation of directed graph to find the out degree of each vertex will take O(E+V) time in worst case i.e. for an element we have to search *n* time.

T2 : Solution

O(V + E)

In adjacency matrix representation of directed graph to find universal sink will take O(V + E) time i.e. for every entry in adjacency matrix we have to check *n* time.

T3 : Solution

BST and O(*n*)

Sorting the array using **binary search tree** will take O(*n*) time i.e. inorder sequence.

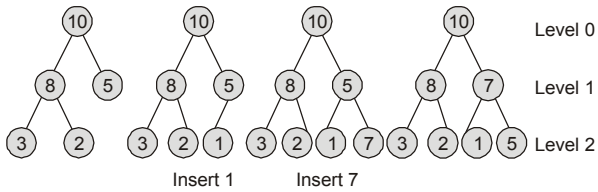
Sorting the array using **min heap tree** will take O(*n* log *n*) time i.e. O(*n*) time to build and log *n* time to get every minimum element. So O(*n*) + O(*n* log *n*) = O(*n* log *n*).

In the giving question **binary search tree is better than min heap tree by O(*n*) time**.

T4 : Solution

(d)

Initially the sequence of Max Heap is



So, the sequence of level order traversal is 10, 8, 7, 3, 2, 1, 5.

T5 : Solution

(b)

Build heap method to make heap on 'n' elements will take $O(n)$ time. Since less than this is not possible.

■■■■

4 Greedy Method and Dynamic Programming

Detailed Explanation of Try Yourself Questions

T1 : Solution

(b)

The activities are A, B, C, D, E, F, G and H. The starting and ending times are "$a_s\, b_s\, c_s\, a_e\, d_s\, c_e\, e_s\, f_s\, b_e\, d_e\, g_s\, e_s\, f_e\, h_s\, g_e\, h_e$".

The gap between starting and ending time of B is greater than all other activities. Consider the order $b_s c_s a_e d_s c_e e_s f_s b_e$ be the gap between b_s to b_e is 6 in which two activity a_e and c_e ended so minimum number of required rooms is 4.

T2 : Solution

(c)

Undirected graph G contains n nodes.

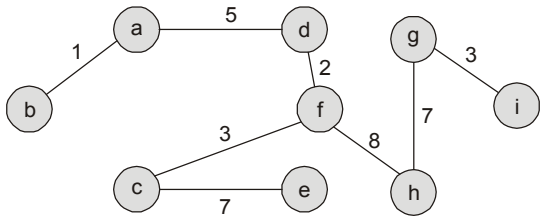
	1	2	3	⋯	n
1	0	1	1	⋯	1
2	1	0	1	⋯	1
3	1	1	0	⋯	1
⋮					
n	1	1	1	⋯	0

In undirected graph G, the diagonal elements are 0 means there is no self loop of any vertex. Where each vertex in G is connected through $n-1$ vertices in the graph G so G is a complete graph. In a complete graph apply the Cayley's theorem for the total number of minimum spanning trees in a complete graph G with n vertices is n^{n-2} so G has multiple distinct MSTs. The cost of a spanning tree is sum of all edges. If G contains n vertices then we will must stop the algorithm after adding the $n-1$ edges so each spanning tree have a cost $n-1$.

T3 : Solution

(c)

Prim's algorithm having special property which is, while finding minimum cost spanning tree, graph always be connected.



Minimum cost spanning tree using Prim's

Now for correct options, we need to check while entry for any edge except for 1st edge, only one vertex match from all left side (i.e. visited edges) edges.

(a) (a,b), (d,f) since (d,f) does not have any common vertex with (a,b) so not possible.

(b) (c,e), (c,f), (f,d), (d,a), (a,b), (g,h) since (g,h) edge does not have any common vertex from all left hand side edges, so not possible.

(c) (d,f), (f,c), (d,a), (a,b), (c,e), (f,h), (g,h), (g,i) is true since does not violate any rule.

T4 : Solution

(b)

We observe a pattern in weight of MST being formed:

For $n = 3 \Rightarrow (1 + 2 + 3) + (1)$

For $n = 4 \Rightarrow (1 + 2 + 3 + 4) + (1 + 2)$

For $n = 5 \Rightarrow (1 + 2 + 3 + 4 + 5) + (1 + 2 + 3)$

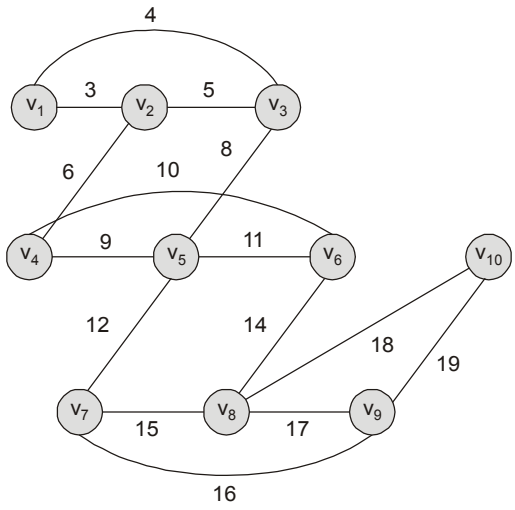
For $n = 6 \Rightarrow (1 + 2 + 3 + 4 + 5 + 6) + (1 + 2 + 3 + 4)$

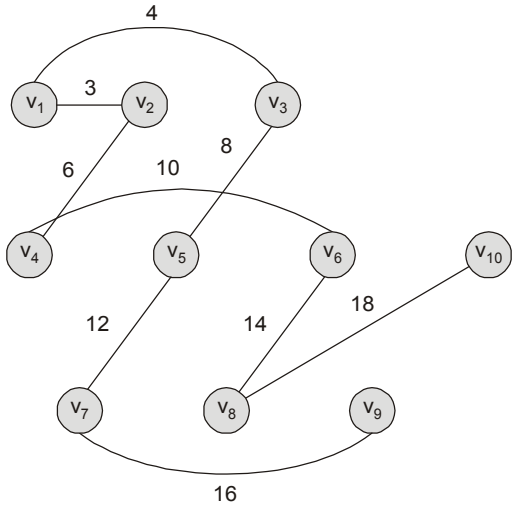
In general, total weight of MST for n :

$$= \sum_{i=1}^{n} i + \sum_{i=1}^{n-2} i = n^2 - n + 1$$

T5 : Solution

(c)





$v_5 \rightarrow v_6: (v_5 - v_3 - v_1 - v_2 - v_4 - v_6)$

$\Downarrow$

31

$v_5 \rightarrow v_6 = 31$

■■■■

5 Graph Traversal and Basic Sorting Algorithm's

Detailed Explanation of Try Yourself Questions

T1 : Solution

(b)

The subset-sum problem is NP-complete problem.

For example given a set S = {–7, –3, –2, 5, 8} the algorithm determine whether the sum of the element of S′ is zero such that S′ ⊆ S. For example if S′ = {– 3, – 2, 5} then return yes.

In given dynamic algorithm x is a two dimensional Boolean array contains n rows and W + 1 columns where W is a positive weight or sum.

$X[i, j]$ $1 \le i \le n$, $0 \le j \le W$

The i^{th} row determines the element of subset of S, and j^{th} column determines the corresponding weight.

If $2 \le i \le n$ and $a \le j \le W$

Let subset $S' = \{a_1, a_2, ..., a_i\}$ the weight as $a_1, ..., a_2$ is W.

So $X[i, j] = X[i - 1, j] \vee X[i - 1, j - a_i]$

The row starts from 2nd position so $i = i - 1$. The column position starts from j but we will add upto a_i^{th} position so $j = j - a_i$.

T2 : Solution

(c)

The entry X[n, W] is true, because we find the weight W of subset of S which contains n elements.

T3 : Solution

(d)

If an algorithm Q solves subset-sum problem in $O(nW)$ time where w is an integer which is constant so Q solves the subset-sum problem in $O(n)$ time because W is a constant and input must encoded in binary.

T4 : Solution

(34)

$$A = qpqrr$$
$$B = pqprqrp$$
$$\text{LCS}(A, B) = qpqr$$
$$(\text{or}) = qprr$$
$$(\text{or}) = pqrr$$
$$x = \text{Length of LCS}(A, B) = 4$$
$$y = \#\,\text{LCS}(A, B) = 3$$
$$\therefore \quad x + 10y = 4 + 10 \times 3 = 34$$

■■■■