7Vals Coding Test 2021

Question 9

Question text

Whats the Time and space complexity for the below code?

```
int sum(int n) {
  if (n <= 0) {
    return 0;
  }
  return n + sum(n-1);

}
```

Select one:

- a. O(N), O(1)
- b. O(N^2), O(N^2)
- c. O(N), O(N)
- d. O(1), O(N^2)

Question 10

Question text

What's the space complexity for the below code?

```
int pairSumSequence(int n) {
  int sum = 0;
  for (int i = 0; i < n; i++) {
    sum += pairSum(i, i + 1);
   }
  return sum;
}



int pairSum(int a, int b) {
  return a + b;
}
```

Select one:

- a. O(N^2)
- b. O(1)
- c. O(N)
- d. Cannot be determined

Question 11

Question text

SELECT name

FROM items

WHERE price != 100;

The above query selects:

Select one:

- a. Tuples with price equal to null or price not equal to 100.
- b. Tuples with price not null and price not equal to 100.
- c. Tuples with price not null.
- d. None of above.

Question 12

Question text

Consider the following data to be stored in a hash table:

1568, 3586, 1567, 3585, 1681, 1684, 3582, 1683

If the hashing function is "x mod 10", which of the following statements are true?

i. 3586, 3585, 3582 hash to the same value

ii. 1568, 1567 hash to the same value

iii. All elements hash to the same value

iv. Each element hashes to a different value

Select one:

- a. ii only
- b. iv only
- c. i only
- d. i and ii only

Question 13

Question text

What will be the height of a balanced full binary tree with 16 leaves? (Assume that the height of a tree with a single node starts from 1)

Select one:

- a. 8
- b. 6
- c. 4
- d. 5

Question 14

Question text

There's a discussion about the total number of PSL matches in the upcoming years as no one's sure about the actual number of teams because there may be an addition of teams or no additions. So, the count of the teams is not sure. Now based on the number of teams you have to tell the total number of matches that will be played based on the following **two rules.**

1) If the number of teams is even, each team gets paired with another team. A total of n / 2 matches are played, and n / 2 teams advance to the next round.

2) If the current number of teams is odd, one team randomly advances in the tournament, and the rest gets paired. A total of (n - 1) / 2 matches are played, and (n - 1) / 2 + 1 teams advance to the next round.

Given the above rules information you have to assist PSL management to calculate the number of matches played in the tournament until a winner is decided. Your job is to write a function **int**

totalLeageMatchesCount(int numberOfTeams) which takes a single integer argument as numberOfTeams and returns the total number of league matches until a winner is decided.

Example:
Input:

numberOfTeams = 11

Output: 10

Explanation: Details of the tournament:

- 1st Round: Teams =11, Matches =5, and 6 teams advance.

- 2nd Round: Teams = 6, Matches = 3, and 3 teams advance.

- 3rd Round: Teams = 3, Matches = 1, and 2 teams advance.

- 4th Round: Teams = 2, Matches = 1, and 1 team is declared the winner.

Total number of matches = 5 + 3 + 1 + 1 = 10.

Question 15

Question text

A hotel management team decided to launch an online room booking system. Users can visit their website and book rooms in advance. When booking, the user must specify the start and end date, as well as the number of rooms to book. If the requested rooms are available for reservation for the period specified by the user, the system will accept the reservation request otherwise the request will be rejected. If the system receives more than one reservation request, the requests should be processed in order of arrival time, that is, the request that arrived first should be processed first.

Implement a function ***processRequests(reservations, requests, totalRooms)*** that, given the approved reservations, reservation requests and total rooms, returns an array of same length as ***requests.*** For each reservation request, populate the resultant array with 1 if the reservation is accepted else insert 0. The details of the input arguments is given below:

reservations:

2d array, containing all approved reservations. The structure of the array is as follows:

[

[start, end, quantity],

[start, end, quantity]

....

]

requests:

2d array, containing all reservation requests. The structure of the array is as follows:

[

[start, end, quantity],

[start, end, quantity]

....

]

totalRooms: Total rooms in the hotel (integer)

start: Day of month, 1 to 30 (integer)

end: Day of month, 1 to 30 (integer)

quantity: Number of rooms requested (integer)

Assume that all reservations have reached within one month, and all rooms have the same capacity. The order of the reservation requests is also kept relative to the arrival time, so the zero index request is the oldest request in ***requests*** array.

Note: While processing a request, you must consider previously approved requests, check example 2 for detail.

Example 1:

reservations:

{

{04, 06, 02},

{15, 20, 10}

}

requests:

{

{03, 05, 05},

{15, 20, 06},

{25, 28, 06}

}

totalRooms: 15

Resultant Array *-> {1, 0, 1}*

Explanation:

In above example, total rooms available in the hotel are 15. Two reservations are already approved, i.e, initially two rooms are booked from 4 to 6 and 10 rooms are booked from 15 to 20.

request 1: {03, 05, 05}

A user requests 5 rooms from 3 to 5, based on approved bookings at current state, we have a maximum of 13 rooms available for the duration of 3 to 5, so the request will be approved.

request 2: {15, 20, 06}

Only 5 rooms are available for a period of 15 to 20, therefore the request will be rejected.

request 3: {25, 28, 06}

All 15 rooms are available for the duration of 25 to 28, so the request will be approved.

Example 2:

reservations:

{
{04, 06, 10},
{15, 20, 10}

}

requests:

{
{05, 06, 05},
{06, 09, 01}

}

totalRooms: 15

Resultant Array -> *{1, 0}*

Explanation:

Total rooms available in the hotel are 15. Two reservations are already approved, i.e, initially 10 rooms are booked from 4 to 6 and 10 rooms are booked from 15 to 20.

request 1: *{05, 06, 05}*

Based on approved bookings, we have a maximum of 5 rooms available for the duration of 5 to 6, so the request will be approved.

request 2: *{06, 09, 01}*

No rooms available on day 06, due to previously approved booking request, so the request will be rejected.

Question 16

Question text

Ahmed is playing a game in which he is given two integer arrays **numbers** and **multipliers** of sizes **n** and **m** respectively. He starts the game with a score of 0, and his goal is to maximize the score he can obtain given the following rules.

Ahmed will iterate through the **multipliers** array in order (from **multipliers[0]** to **multipliers[m-1]**), and for each integer **x** in **multipliers** he will perform **one** of the following operations:

- Choice 1: Multiply **x** with the first integer in **numbers**, and add the result to his score. The first integer in **numbers** will then be deleted.
- Choice 2: Multiply **x** with the last integer in **numbers**, and add the result to his score. The last integer in **numbers** will then be deleted.

Note that only **one** of the above operations will be performed at a time.

Also note that, in total, **m** operations will be performed (one for each integer in **multipliers**), and there may be integers left over in **numbers** at the end (as in Example 2).

Return the maximum score Ahmed can obtain. If you get "Testing was aborted due to error" result, this **may** be because your solution was too slow and needs to be optimized.

Input:

numbers - an integer array

multipliers - an integer array

n - size of numbers

m - size of multipliers

Output:

The maximum score Ahmed can obtain

Constraints:

1 <= **m** <= 1000

m <= **n** <= 100000

-100 <= **nums[i]**, **multipliers[i]** <= 100

Example 1:

Input:

numbers = [1, 2, 3]

multipliers = [3, 2, 1]

Output: **14**

Explanation:

For each integer in **multipliers** we must choose either the first or the last integer in **numbers** and multiply them, adding the result to the total score.

The following steps lead to an optimal solution:

- Select from the end: add 3 * 3 = **9** to the score. After this operation, **numbers** = [1, 2]
- Select from the end: add 2 * 2 = **4** to the score. After this operation, **numbers** = [1]
- Select the only remaining integer: adding 1 * 1 = **1** to the score. **numbers** = [ ]

The total score is 9 + 4 + 1 = **14**.

Example 2:

Input:

numbers = [-5, -3, -3, -2, 7, 1]

multipliers = [-10, -5, 3, 4, 6]

Output: **102**

Explanation:

The following steps lead to an optimal solution:

- Select from the start: add -5 * -10 = **50** to the score. **numbers** = [-3, -3, -2, 7, 1]
- Select from the start: add -3 * -5 = **15** to the score. **numbers** = [-3, -2, 7, 1]
- Select from the start: add -3 * 3 = **-9** to the score. **numbers** = [-2, 7, 1]
- Select from the end: add 1 * 4 = **4** to the score. **numbers** = [-2, 7]
- Select from the end: add 7 * 6 = **42** to the score. **numbers** = [-2]

The total score is 50 + 15 - 9 + 4 + 42 = **102**.