Problems based on Recursion - 6

Assignment Solutions

PW SKILLS

Q1 - Count all the possible paths on an m x n grid from top left (grid[0][0]) to bottom right (grid[m-1][n-1]) having constraints that from each cell you can either move only to right or down. (Easy)

Input: m = 2, n = 3

Expected Output: 3

Explanation:

- The recursive function "NumberOfPaths" accepts 4 parameters as the destination coordinates(m,n) and the current coordinates(i,j).
- The terminating condition could be if the i pointer has exceeded the number of rows that is m value or j pointer has exceeded the number of columns that is n value that shows we are moving out of the grid and that should not be counted as a valid path so we returned 0 on the behalf of such kind of path.
- Another base case could be if the value of i = m-1 i.e the last row and j = n-1 i.e the last column i.e we have reached the destination successfully then we have to return 1 because this shows presence of one valid path to reach from source to destination.
- If the values of i and j pointers are such that they lie inside the grid, then we have two alternatives either we go down i.e from coordinates i,j to i+1,j or second option could be to move from i,j to i,j+1 i.e i n right direction.
- Now we know that we could probably get answer if we consider only moving to right direction, also we can get some possible paths if we consider only moving to the down direction. But the paths are also possible if we move down and right mutually.
- That's why we have made two calls one for the right direction and another for the down direction and mutually summed them up that gives us answer for all the three possibilities to reach the destination. (only right , only down , both right and down)

Code:

https://pastebin.com/755JNBQm

```
/Library/Java/JavaVirtualMachines/
Enter the dimensions of the matrix
2 3
3

Process finished with exit code 0
```

Q2 - Given an array of integers, print a sum triangle from it such that the first level(the bottom level in triangular fashion) has all array elements. From then, at each level, the number of elements is one less than the previous level and elements at the level is the sum of consecutive two elements in the previous level. (Medium)

Input1 : n = 5
arr = {1, 2, 3, 4, 5}

Output1 : [48]
[20, 28]
[8, 12, 16]
[3, 5, 7, 9]
[1, 2, 3, 4, 5]

Explanation:

- Create a recursive function with 1 parameter as the input array.
- Whenever our array's length is less than 1 meaning array has no element, we return from the function.
- Create a new array with length 1 less than the current array's length.
- Traverse through the input array using a for loop.
- At each iteration add a new element which is the sum of consecutive elements in the array passed as a parameter.
- After the for loop, make a recursive call and pass the newly created array in the previous step.
- While backtracking print the array (for printing in reverse order).

Code:

https://pastebin.com/yxiL3x4Y

```
/Library/Java/JavaVirtualMachines/jdk-
Enter the length of array:
5
Enter the elements of array:
1 2 3 4 5
[48]
[20, 28]
[8, 12, 16]
[3, 5, 7, 9]
[1, 2, 3, 4, 5]

Process finished with exit code 0
```

Q3 - Given an array of size n, generate and print all possible combinations of r elements in array. (Hard)

Input:

n = 4

{1, 2, 3, 4}

r = 2

Expected Output:

{1, 2}

{1, 3}

{1, 4}

{2, 3}

{2, 4}

{3, 4}

Explanation:

- We create 2 functions.
- printCombination creates a temporary array of length r to store all combinations one by one and then calls the recursive function combination.
- The recursive function has parameters as input array, n, r, index for temporary array, temporary array, i for input array.
- If index == r, it means the current combination is ready to be printed, so we print it.
- If i is greater than or equal to n, then no more elements are there to put in the temporary array, return.
- Next, We one by one consider every element of input array, and recur for two cases:
    i. The element is included in current combination (We put the element in data[] and increment next available index in data[])
    ii. The element is excluded in current combination (We do not put the element and do not change index)
- When the number of elements in data[] become equal to r (size of a combination), we print it.

Code:

https://pastebin.com/qAmbjHV3

```
/Library/Java/JavaVirtualMachines/jdk
Enter the length of array:
4
Enter the elements of array:
1 2 3 4
Enter r:
2
1 2
1 3
1 4
2 3
2 4
3 4

Process finished with exit code 0
```

Q4 - Given two sorted arrays A and B of length m and n respectively, generate all possible arrays such that the first element is taken from A then from B then from A, and so on in increasing order till the arrays are exhausted. The generated arrays should end with an element from B. (Hard)

Input:

m = 3

n = 4

A = {10, 15, 25}

B = {1, 5, 20, 30}

Expected Output:

10 20

10 20 25 30

10 30

15 20

15 20 25 30

15 30

25 30

Explanation:

- We create a recursive function with parameters as both the arrays, the third array which will store all the combinations, two pointers, i and j to keep track of indices of array A and B respectively, length of all three arrays, and a flag variable.
- Flag variable is of boolean type, it indicates whether current element in output should be taken from 'A' or 'B'.
- In the function, first check whether the flag is true or false.
- If the flag is true, first check if the length of the output array is greater than 0 which indicates that we have one of the combinations, and we print it using a printArr function we create.
- Next, while the flag is still true, we continue the combination in the same array by traversing over elements of the first array from its current index.
- If the length of the output array is 0, it means we are adding the first element, so directly add the current element from the first array to the output array at its current index.
- Give a recursive call with change in parameters: index of first array as current index +1 and reverse flag.
- If length of output array is not 0, we need to check if element at current index of output array is less than element at current index of first array, only then we add it to output array, and give a recursive call with change in parameters: index of first array as current index +1, index of output array as current index +1 and reverse flag.
- If the flag is false, we need to add an element from the second array.
- Traverse through the second array, if element at current index of output array is less than element at current index of second array, only then we add it to output array, and give a recursive call with change in parameters: index of second array as current index +1, index of output array as current index +1 and reverse flag.

Code:

https://pastebin.com/LTZruwc3

```
/Library/Java/JavaVirtualMachines/jdk-19
Enter the length of both the arrays:
3 4
Enter the elements of first array:
10 15 25
Enter the elements of second array:
1 5 20 30
10 20
10 20 25 30
10 30
15 20
15 20 25 30
15 30
25 30

Process finished with exit code 0
```