Backtracking

Assignment Solutions

PW SKILLS

Q1. Given an integer array arr and an integer k, return true if it is possible to divide the vector into k non-empty subsets with equal sum.

Input: arr = [1,3,2,2] k = 2
Output: true
Explanation : 1 + 3 and 2+2 are two subsets with equal sum.
Code : ASS_Code1.java

Output :

```
yes it is possible to partition the array.
```

Approach:

The most basic thing is in order to divide the arr into k subsets with equal sum, the sum of arr should be divisible by k. If the sum is not divisible by k then we can return false, else we will call a recursive function helper which basically divides arr into all possible k subsets and for each of them it checks if the sum of all of these subsets is equal.

- The function helper() takes in vector arr, vector vis which basically tells if the current index si is already used by any other subset or not, an integer curr_sum which stores the curr_sum of each subset and a target variable which is basically the desired sum that each subset should have inorder to have k subsets with equal sum and thus the target will be sum of arr / k;
- Our base case will be:
- If k==1 i.e. if we need just 1 subset with equal sum, in that case the answer will always be true.
- If (curr_sum>target), this means that the current subset we are making has the sum greater than our desired sum so we will return false for this subset.
- If (curr_sum==target) this means that we have found a subset which has the sum equal to our desired sum and so we will now look for other subsets and we need to find k-1 subsets now, since we have found 1 subset.
- We run a for loop which basically starts with our index i.e. the index of the element at which we currently are. We check if we have already considered that index in any previous subset for this recursive iteration, if vis[i]=-1 this means that we haven't considered it for any other subset in previous iteration and so we can use it in this iteration and we then mark it visited by vis[i]=1 for the current recursive iteration and thus call the recursive function with curr_sum=curr_sum+arr[i] and the starting index for the next iteration changes to i+1 since we will be considering the next elements after i.
- In this way we keep running the recursive function.
- Once a recursive call returns we then backtrack and make vis[i] =-1 so that we can use arr[i] in making other subsets.

Q2. Given an integer array arr, print all the possible permutations of the given array.

Note : The array will only contain non repeating elements.
Input 1 : arr = [1, 2, 3]

Output1 : [[1,2,3] , [1,3,2] , [2,1,3] , [2,3,1] , [3,1,2] , [3,2,1]]

Q1. Given an integer array arr and an integer k, return true if it is possible to divide the vector into k non-empty subsets with equal sum.

Solution :

Code : ASS_Code2.java

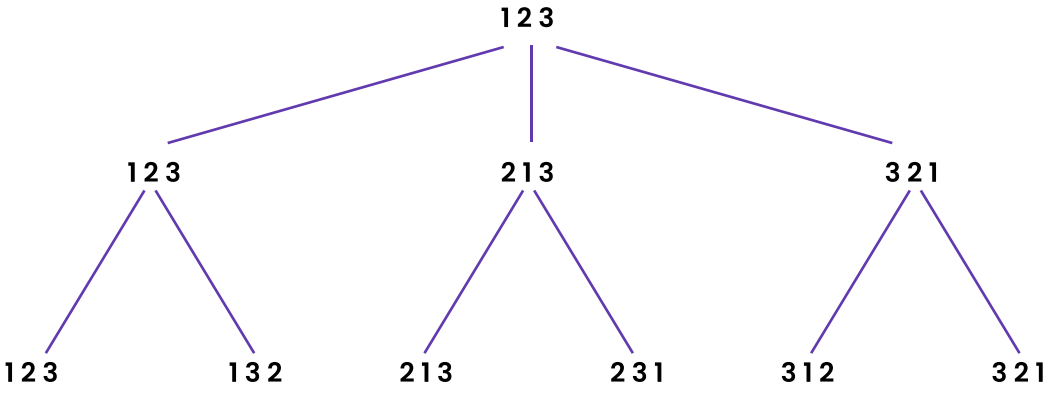
Output :

```
The possible permutations are :
1  4  2  3
1  4  3  2
1  2  4  3
1  2  3  4
1  3  2  4
1  3  4  2
4  1  2  3
4  1  3  2
4  2  1  3
4  2  3  1
4  3  2  1
4  3  1  2
2  4  1  3
2  4  3  1
2  1  4  3
2  1  3  4
2  3  1  4
2  3  4  1
3  4  2  1
3  4  1  2
3  2  4  1
3  2  1  4
3  1  2  4
3  1  4  2
```

Approach :

The approach is much similar to what we have used in the previous question.

- We can get all the required permutations by swapping the number with all the other numbers ahead one by one



- We will make a helper function called permutationsHelper() that will iterate through the given vector nums and will keep on swapping the nums[start] with the rest of the remaining element.

Q3. Given a collection of numbers, nums, that might contain duplicates, return all possible unique permutations in any order.

Example 1:
Input: nums = [1,1,2]
Output:
[[1,1,2], [1,2,1], [2,1,1]]

Example 2:
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]

Solution :
Code : ASS_Code3.java

Output :

```
The possible permutations are :
1  3  4  4
1  4  3  4
1  4  4  3
3  1  4  4
3  4  1  4
3  4  4  1
4  1  3  4
4  1  4  3
4  3  1  4
4  3  4  1
4  4  1  3
4  4  3  1
```

Approach :

- Use an extra boolean array " boolean[] used" to indicate whether the value is added to the list.
- Sort the array "int[] nums" to make sure we can skip the same value.
- when a number has the same value as its previous, we can use this number only if its previous is not used.

Q4. Check if the product of some subset of an array is equal to the target value.

Input : n = 5 , target = 16
Array = [2 3 2 5 4]
Here the target will be equal to 2x2x4 = 16
Output : YES

Solution :
Code : ASS_Code4.java

Output :

```
Enter the number of elements you want and the value of target respectively :
7 12
Enter the array elements
2 3 6 5 4 3 9
YES
```

Approach :

- This is a generic subset problem in which we consider 2 cases.
- Either we are including the current element in the product or we are excluding it.
- Just make a recursive function and take two cases. In the first case we consider taking the element for product and multiply it with product
- In the second case, we do not take the element and leave that element.
- This way we reach the point where i is the end of the array, simply check whether the product is equal to the target.

Q5. The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other.Given an integer n, return the number of distinct solutions to the n-queens puzzle.

Input: n = 4
Output: 2
Explanation: There are two distinct solutions to the 4-queens puzzle as shown.
Input: n = 1
Output: 1

Solution :

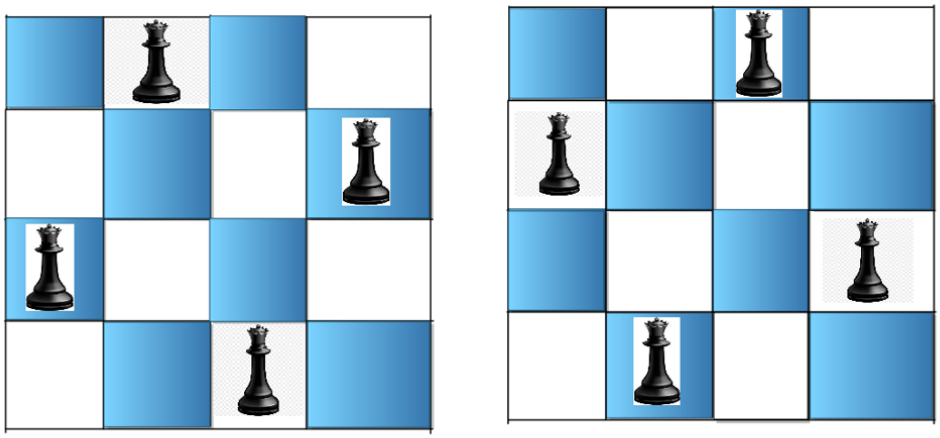
Code : ASS_Code5.java

Output :

```
The desired answer is : 2
```

Approach :

- This Problem is Almost Similar to N-Queens that we have discussed in our lecture, what extra we have to do over here is, Instead of printing the position we have to count the number of ways in which Queens can be arranged on N-By-N chess.



Two arrangements possible for 4 queens