Combination Sum III

Find all possible combinations of ***k*** numbers that add up to a number ***n***, given that only numbers from 1 to 9 can be used and each combination should be a unique set of numbers.

Ensure that numbers within the set are sorted in ascending order.

Example 1:

Input: ***k*** = 3, ***n*** = 7

Output:

```
[[1,2,4]]
```

Example 2:

Input: ***k*** = 3, ***n*** = 9

Output:

```
[[1,2,6], [1,3,5], [2,3,4]]
```

Credits:
Special thanks to @mithmatt for adding this problem and creating all test cases.

Solution 1

```
class Solution {
public:
  void combination(vector<vector<int>>& result, vector<int> sol, int k, int n) {
    if (sol.size() == k && n == 0) { result.push_back(sol); return ; }
    if (sol.size() < k) {
      for (int i = sol.empty() ? 1 : sol.back() + 1; i <= 9; ++i) {
        if (n - i < 0) break;
        sol.push_back(i);
        combination(result, sol, k, n - i);
        sol.pop_back();
      }
    }
  }

  vector<vector<int>> combinationSum3(int k, int n) {
    vector<vector<int>> result;
    vector<int> sol;
    combination(result, sol, k, n);
    return result;
  }
};
```

written by yushu original link here

Solution 2

```
vector<vector<int>> combinationSum3(int k, int n) {
    vector<vector<int>> result;
    vector<int> path;
    backtrack(result, path, 1, k, n);
    return result;
}

void backtrack(vector<vector<int>> &result, vector<int> &path, int start, int k,
int target){
    if(target==0&&k==0){
        result.push_back(path);
        return;
    }
    for(int i=start;i<=10-k&&i<=target;i++){
        path.push_back(i);
        backtrack(result,path,i+1,k-1,target-i);
        path.pop_back();
    }
}
```

written by zephyr3 original link here

Solution 3

```
 public List<List<Integer>> combinationSum3(int k, int n) {
    List<List<Integer>> ans = new ArrayList<>();
    combination(ans, new ArrayList<Integer>(), k, 1, n);
    return ans;
}

private void combination(List<List<Integer>> ans, List<Integer> comb, int k,  int
start, int n) {
    if (comb.size() == k && n == 0) {
        List<Integer> li = new ArrayList<Integer>(comb);
        ans.add(li);
        return;
    }
    for (int i = start; i <= 9; i++) {
        comb.add(i);
        combination(ans, comb, k, i+1, n-i);
        comb.remove(comb.size() - 1);
    }
}
```

written by jinwu original link here