Combinations

Given two integers n and k, return all possible combinations of k numbers out of 1 ... n.

For example,
If n = 4 and k = 2, a solution is:

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

Solution 1

Basically, this solution follows the idea of the mathematical formula C(n,k)=C(n-1,k-1)+C(n-1,k).

Here C(n,k) is divided into two situations. Situation one, number n is selected, so we only need to select k-1 from n-1 next. Situation two, number n is not selected, and the rest job is selecting k from n-1.

```java
public class Solution {
    public List<List<Integer>> combine(int n, int k) {
        if (k == n || k == 0) {
            List<Integer> row = new LinkedList<>();
            for (int i = 1; i <= k; ++i) {
                row.add(i);
            }
            return new LinkedList<>(Arrays.asList(row));
        }
        List<List<Integer>> result = this.combine(n - 1, k - 1);
        result.forEach(e -> e.add(n));
        result.addAll(this.combine(n - 1, k));
        return result;
    }
}
```

written by kxcf original link here

Solution 2

```
    public static List<List<Integer>> combine(int n, int k) {
        List<List<Integer>> combs = new ArrayList<List<Integer>>();
        combine(combs, new ArrayList<Integer>(), 1, n, k);
        return combs;
    }
    public static void combine(List<List<Integer>> combs, List<Integer> comb, int 
start, int n, int k) {
        if(k==0) {
            combs.add(new ArrayList<Integer>(comb));
            return;
        }
        for(int i=start;i<=n;i++) {
            comb.add(i);
            combine(combs, comb, i+1, n, k-1);
            comb.remove(comb.size()-1);
        }
    }
```

written by fabrizio3 original link here

Solution 3

my idea is using backtracking ,every time I push a number into vector,then I push a bigger one into it; then i pop the latest one,and push a another bigger one... and if I has push k number into vector,I push this into result;

this solution take 24 ms.

```
class Solution {
public:
    vector<vector<int> > combine(int n, int k) {
        vector<vector<int> >res;
        if(n<k)return res;
        vector<int> temp(0,k);
        combine(res,temp,0,0,n,k);
        return res;
    }

    void combine(vector<vector<int> > &res,vector<int> &temp,int start,int num,in
t n ,int k){
        if(num==k){
            res.push_back(temp);
            return;
        }
        for(int i = start;i<n;i++){
            temp.push_back(i+1);
            combine(res,temp,i+1,num+1,n,k);
            temp.pop_back();
            }
        }
};
```

written by nangao original link here