77. Combinations

Created	@July 9, 2021 12:18 AM
Difficulty	Medium
LC Url	https://leetcode.com/problems/combinations/
Importance	*****
Tag	Backtrack
Video	

Given two integers `n` and `k`, return *all possible combinations of* `k` *numbers out of the range* `[1, n]`.

You may return the answer in **any order**.

Example 1:

```
Input: n = 4, k = 2
Output:
[
  [2,4],
  [3,4],
  [2,3],
  [1,2],
  [1,3],
  [1,4],
]
```

Example 2:

```
Input: n = 1, k = 1
Output: [[1]]
```

Constraints:

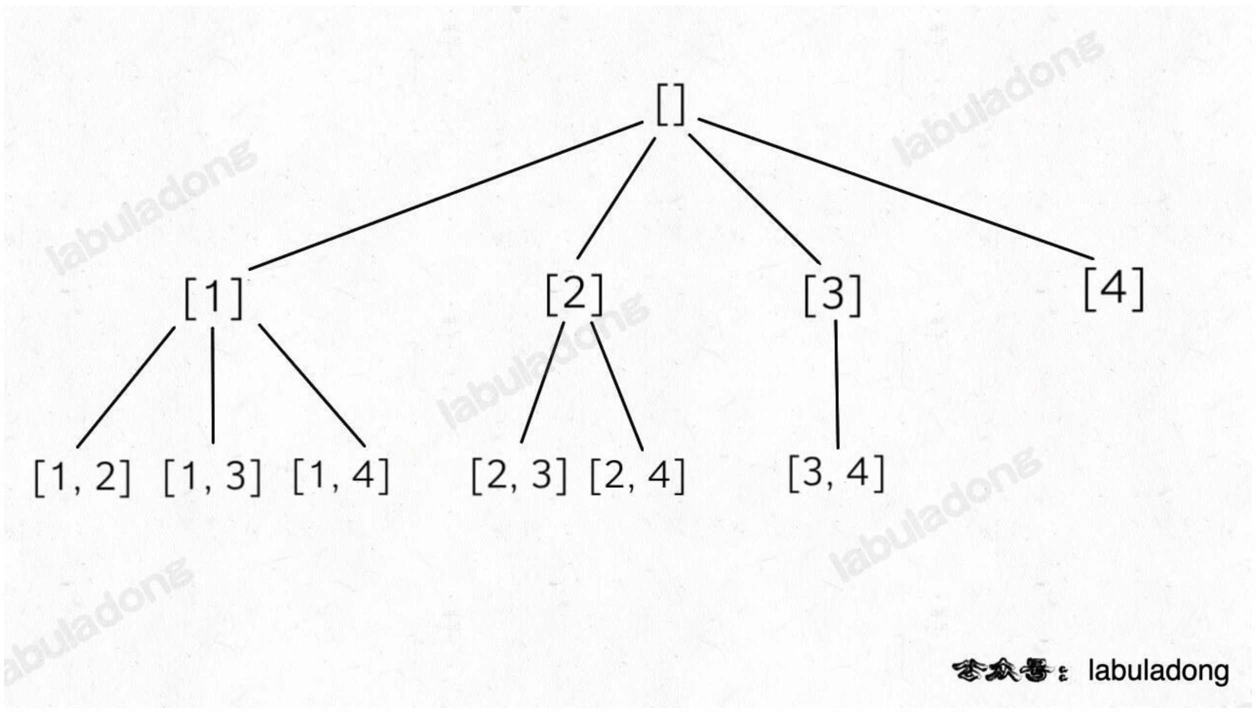
- `1 <= n <= 20`

- `1 <= k <= n`

Solution

PS：这道题在《算法小抄》的第 293 页。

这也是典型的回溯算法，`k` 限制了树的高度，`n` 限制了树的宽度，继续套我们以前讲过的 回溯算法模板框架 就行了：



```
class Solution:
    def combine(self, n: int, k: int) -> List[List[int]]:
        res = []
        self.backtrack(n, 1, k, [], res)
        return res

    def backtrack(self, n, start_index, k, subset, res):
        # base case
        if k == len(subset):
            res.append(list(subset))
            return

        # backtrack
        for i in range(start_index, n + 1):
            subset.append(i)
            self.backtrack(n, i + 1, k, subset, res)
            subset.pop()
```

```
// labuladong
List<List<Integer>> res = new LinkedList<>();
// 记录回溯算法的递归路径
LinkedList<Integer> track = new LinkedList<>();

// 主函数
public List<List<Integer>> combine(int n, int k) {
    backtrack(1, n, k);
    return res;
}

void backtrack(int start, int n, int k) {
    // base case
    if (k == track.size()) {
        // 遍历到了第 k 层，收集当前节点的值
        res.add(new LinkedList<>(track));
        return;
    }

    // 回溯算法标准框架
    for (int i = start; i <= n; i++) {
        // 选择
        track.addLast(i);
        // 通过 start 参数控制树枝的遍历，避免产生重复的子集
        backtrack(i + 1, n, k);
        // 撤销选择
        track.removeLast();
    }
}
```