1. 代码题

回溯法做全排列

class Solution {

public:

vector<vector<int>> result;

vector<vector<int>> permute(vector<int>& nums) {

vector<int> track;

backtrack(nums, track);

return result;

}

void backtrack(vector<int> nums, vector<int> track) {

if(track.size() == nums.size()) {

result.push\_back(track);

return;

}

int nums\_size = nums.size();

for(int i = 0; i < nums\_size; ++ i) {

//如果nums[i]没有出现在track中，即可供选择

if(find(track.begin(), track.end(), nums[i]) == track.end()) {

track.push\_back(nums[i]);

backtrack(nums, track);

track.pop\_back();

}

}

}

};

回溯法做全排列不能重复

class Solution {

public:

vector<vector<int>> res;

void backtrack(vector<int> &candidates, vector<int> &track, int sum, int target, int start) {

if(target==sum) {

res.push\_back(track);

return;

}

for(int i =start; i < candidates.size(); ++ i) {

if(target<sum||i>start&&candidates[i]==candidates[i-1]) {

continue;

track.push\_back(candidates[i]);

backtrack(candidates, track,sum+candidates[i],target,i+1);

track.pop\_back();

}

}

}

vector<vector<int>> combinationsum2(vector<int>& candidates, int target) {

vector<int> track;

sort(candidates.begin(),candidates.end());

backtrack(candidates, track,0,target,0);

return res;

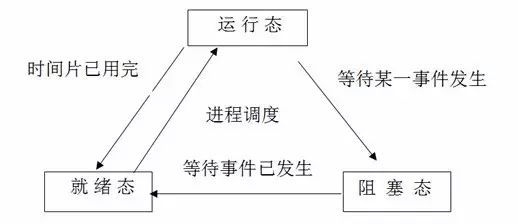
}

};

2. 知识点

**进程有哪几种状态？**

* 就绪状态：进程已获得除处理机以外的所需资源，等待分配处理机资源；
* 运行状态：占用处理机资源运行，处于此状态的进程数小于等于CPU数；
* 阻塞状态： 进程等待某种条件，在条件满足之前无法执行；



**线程有几种状态？**

1. 新建(NEW)：新创建了一个线程对象。

2. 可运行(RUNNABLE)：线程对象创建后，其他线程(比如main线程）调用了该对象的start()方法。该状态的线程位于可运行线程池中，等待被线程调度选中，获取cpu 的使用权 。

3. 运行(RUNNING)：可运行状态(runnable)的线程获得了cpu 时间片（timeslice） ，执行程序代码。

4. 阻塞(BLOCKED)：阻塞状态是指线程因为某种原因放弃了cpu 使用权，也即让出了cpu timeslice，暂时停止运行。直到线程进入可运行(runnable)状态，才有机会再次获得cpu timeslice 转到运行(running)状态。阻塞的情况分三种：

5. 死亡(DEAD)：线程run()、main() 方法执行结束，或者因异常退出了run()方法，则该线程结束生命周期。死亡的线程不可再次复生。

在给定的时间点上，一个线程只能处于一种状态。

3. 开源项目

将自己最近做的作业上传到了项目文件夹中