Day4

1.你爬一个n格的楼梯，每一步可以走上1格或2格。问有多少种步法爬上去？

记忆化搜索 （线性动态规划）

代码实现：public int climbStairs(int n)

{

if (n==1) return 1;

if (n==2) return 2;

return climbStairs(n-1)+climbStairs(n-2);

}

把求解过的子问题的解记录下来，再次递归到相同规模的子问题，直接返回对应结果。

递推

代码实现：

public int climbStairs(int n)

{ static int sol[100]; //定义仅为这个函数所用的静态变量，用于递推

sol[1]=1; sol[2]=2;

for (int i=3;i<=n;i++) sol[i]=sol[i-1]+sol[i-2];

return sol[n];

}

从边界情况(n=1,n=2)开始，依次求解n=3,4,…的问题，每一个问题可以通过直接使用前面的答案求解。

状态转移方程：sol[n]:=sol[n-1]+sol[n-2] (n>=3) sol[n]表示爬n格楼梯的方案数。

2. 给定一个未排序的序列，求其中最长上升子序列的长度。

搜索

* 1. 枚举所求序列的第1个元素，然后枚举它后面大于它的元素作为序列的下一个元素，以此类推。以此枚举所有的上升子序列。
  2. 枚举得到的所有序列中，最长的一个，其长度即为答案。

代码实现1：

int ans;

void dfs(const vector<int>& nums, int a,int n,int cnt) {

//在枚举的序列，是第a个元素处于子序列第cnt位的情况

if (cnt>ans) ans=cnt; //若以第a个元素结尾，更新答案

for (int i=a+1;i<n;i++) if (nums[i]>nums[a]) dfs(nums,i,n,cnt+1); //枚举下一个元素

}

int lengthOfLIS(vector<int>& nums) { //nums为读入的序列

int n=nums.size(); ans=0;

for (int i=0;i<n;i++) dfs(nums,i,n,1); //枚举第1个元素

return ans;}

代码实现2：

int dfs(const vector<int>& nums, int a,int n,int cnt) {

int ans=cnt;

for (int i=a+1;i<n;i++) if (nums[i]>nums[a]) ans=max(ans,dfs(nums,i,n,cnt+1));

return ans; //此处返回的是这个分支找到的所有子序列长度的最大值

}

int lengthOfLIS(vector<int>& nums) {

int n=nums.size(), ans=0;

for (int i=0;i<n;i++) ans=max(ans,dfs(nums,i,n,1));

return ans;

}