# HDU5542（树状数组优化DP）

题意：给你n（1e3）个数，每个数都在[1,1e9]范围。 然后让你保持数的顺序不变，选出长度恰好为m（1<=m<=n）的单调上升子序列。 问你有多少种选择方案

*分析：*

题目涉及到选择，我们不妨回归到一个很经典的DP模型——

用f[i][j]表示前i个数中选择了j个数的上升子序列的方案数。

这个状态的一个好处是，它把这个上升自序列的长度也包含了，就是j。

还有一个性质，我们如何查看一个数能否接在前面的序列后？其实只要知道之前上升子序列的最后一位就可以进行判定了。

鉴于这个，我们修改下f[i][j]的含义——

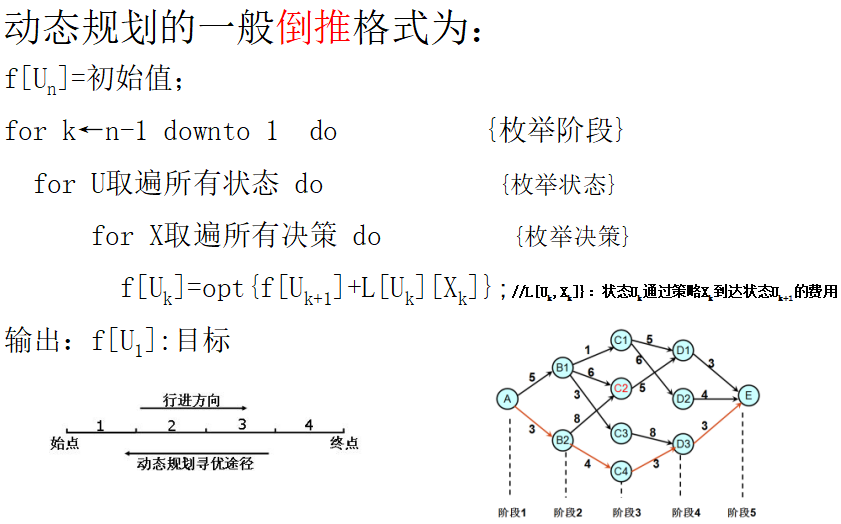
f[i][j]表示前i个数中选择了j个数，且最后一个数严格为a[i]时的上升子序列方案数。

这样就有状态转移f[i][j]=∑f[p][j-1],p∈[1,i-1]且a[p]<a[i]

因为我们可以严格保持i的升序，所以就只需要找到所有长度为j-1且尾节点a[p]<a[i]的子序列个数即可。

且因为for i for j已经使得复杂度变成O(n^2)，所以这个操作要在log(n)级别的时间内完成。

而且这个操作设计到动态操作，于是我们想到树状数组（线段树也可）。



#include <bits/stdc++.h>

using namespace std;

const int mod = 1000000000 + 7;

#define LL long long

#define N 1000 + 10

int n, m, a[N], b[N];

LL dp[N][N];

void update(int x, int y, LL val){

while(x < N) {

dp[x][y] += val;

dp[x][y] %= mod;

x += x & (-x);

}

}

LL get\_sum(int x, int y){

LL ret = 0;

while(x > 0){

ret += dp[x][y];

ret %= mod;

x -= x & (-x);

}

return ret;

}

int main(){

int \_T, kase = 0;

scanf("%d", &\_T);

while(\_T--){

scanf("%d%d", &n, &m);

for(int i = 1; i <= n; i++){

scanf("%d", a + i);

b[i] = a[i];

}

sort(b + 1, b + 1 + n);

for(int i = 1; i <= n; i++) a[i] = lower\_bound(b + 1, b + 1 + n, a[i]) - b;

memset(dp, 0, sizeof dp);

for(int i = 1; i <= n; i++)

for(int k = 1; k <= m; k++){

if(k == 1) update(a[i], 1, 1);

else{

LL tmp = get\_sum(a[i] - 1, k - 1);

update(a[i], k, tmp);

}

}

LL ans = get\_sum(n, m);

printf("Case #%d: %I64d\n", ++kase, ans);

}

return 0;

}

NOIP2013花匠

**题目描述**Description

花匠栋栋种了一排花，每株花都有自己的高度。花儿越长越大，也越来越挤。栋栋决定把这排中的一部分花移走，将剩下的留在原地，使得剩下的花能有空间长大，同时，栋栋希望剩下的花排列得比较别致。  
具体而言，栋栋的花的高度可以看成一列整数h\_1, h\_2, … , h\_n。设当一部分花被移走后，剩下的花的高度依次为g\_1, g\_2, … , g\_m，则栋栋希望下面两个条件中至少有一个满足：  
条件 A：对于所有的1<i<m/2，g\_2i > g\_2i-1，且g\_2i > g\_2i+1；   
条件 B：对于所有的1<i<m/2，g\_2i < g\_2i-1，且g\_2i < g\_2i+1。  
注意上面两个条件在m = 1时同时满足，当m > 1时最多有一个能满足。  
请问，栋栋最多能将多少株花留在原地。

**输入描述**Input Description

输入的第一行包含一个整数 n，表示开始时花的株数。  
第二行包含 n 个整数，依次为h\_1, h\_2,… , h\_n，表示每株花的高度。

**输出描述**Output Description

输出一行，包含一个整数 m，表示最多能留在原地的花的株数。

**样例输入**Sample Input

5   
5 3 2 1 2

**样例输出**Sample Output

3

**数据范围及提示**Data Size & Hint

对于 20%的数据，n ≤ 10；   
对于 30%的数据，n ≤ 25；   
对于 70%的数据，n ≤ 1000，0 ≤ h\_i ≤ 1000；   
对于 100%的数据，1 ≤ n ≤ 100,000，0 ≤ h\_i ≤ 1,000,000，所有的h\_i随机生成，所有随机数服从某区间内的均匀分布。

**解题报告**

这道题明显可以用类似最长上升子序列的动态规划求解，易得思路如下：

用 f(i,0)表示以 i 为结尾的且最后一段上升的子序列最大长度，f(i,1)表示表示 以 i 为 结 尾 的 且 最 后 一 段 下 降 的 子 序 列 最 大 长 度 ， 那 么 答 案 明 显 就 是 max{f(i,0),f(i,1)}

方程：

f(i,0)=max{f(j,1)}+1 0<=j<i 且 h[j]<h[i]

f(i,1)=max{f(j,0)}+1 0<=j<i 且 h[j]>h[i]

边界：f(0,0)=f(0,1)=0

如果直接 DP 毫无疑问复杂度是 O(n^2)，会 TLE，但是，考虑到我们每次取最值时候取得都是一个区间里的数，如 f(i,0)=max{f(j,1)}+1 0<=j<i 且 h[j]<h[i]取得 就是区间[0,h[i]-1]里的最值，所以可以使用线段树或者是 BIT（树状数组）来优 化，这样复杂度就是 O(nlogn)，可以过全部数据。

代码(DP+BIT)(Cpp):

#include <cstdio>

#include <algorithm>

#include <cstring>

using namespace std;

#define MAXN 100010

#define lowbit(x)(((~(x))+1)&x)

#define MAXH 1000010

#define For(i,x) for (int i=x;i;i-=lowbit(i))

#define rep(i,x) for (int i=x;i<=maxh;i+=lowbit(i))

int t0[MAXH],t1[MAXH]; int h[MAXN],n,maxh=0; int f[MAXN][2],ans=0;

void Add0(int x,int y) {

rep(i,x) t0[i]=max(t0[i],y);

}

void Add1(int x,int y) {

rep(i,x) t1[i]=max(t1[i],y);

}

int Max0(int x) {

int rec=0;

For(i,x) rec=max(rec,t0[i]);

return rec;

}

int Max1(int x) {

int rec=0;

For(i,x) rec=max(rec,t1[i]);

return rec;

}

int main() {

scanf("%d",&n);

for (int i=0;i++<n;) {

scanf("%d",&h[i]);

maxh=max(maxh,++h[i]);

f[i][0]=f[i][1]=1;

}

maxh++;

memset(t0,0,sizeof(t0)),memset(t1,0,sizeof(t1));

for (int i=0;i++<n;){

f[i][0]=max(Max0(h[i]-1)+1,f[i][0]);

f[i][1]=max(Max1(maxh-h[i]-1)+1,f[i][1]);

Add0(h[i],f[i][1]),Add1(maxh-h[i],f[i][0]);

ans=max(ans,max(f[i][0],f[i][1]));

}

printf("%d\n",ans); return 0;

}