**算法训练 拦截导弹**

时间限制：1.0s   内存限制：256.0MB

问题描述

　　某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。  
  
　　输入导弹依次飞来的高度（雷达给出的高度数据是不大于30000的正整数），计算这套系统最多能拦截多少导弹，如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

输入格式

　　一行，为导弹依次飞来的高度

输出格式

　　两行，分别是最多能拦截的导弹数与要拦截所有导弹最少要配备的系统数

样例输入

389 207 155 300 299 170 158 65

样例输出

6  
2

锦囊1

贪心和动态规划算法。

锦囊2

求一套系统拦截的导弹最大值为最长不下降子序列问题，用动态规划。 求最少多少套系统需要使用贪心法，每枚后面的炮弹使用能拦截它的最低的系统。

本题的C++参考代码如下：

#include <cstdio>

#include <cstdlib>

#include <cstring>

using namespace std;

int a[10001],high[10001];

int n = 1, top, maw;

void work(int p)

{

int l = 1, r = n, mid;

while(l != r)

{

mid = (l + r) / 2;

if(p < high[mid]) l = mid + 1;

else r = mid;

}

if(p > high[l]) high[l] = p;

if(l > maw) maw = l;

}

void solve(int p)

{

int l = 1, r = n, mid;

while(l != r)

{

mid = (l + r) / 2;

if(p > high[mid]) l = mid + 1;

else r = mid;

}

if(p < high[l]) high[l] = p;

if(l > top) top = l;

}

int main()

{

//freopen("missile.txt","r",stdin);

//freopen("missiles.txt","w",stdout);

while(scanf("%d", &a[n]) != EOF) ++n;

--n;

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

work(a[i]);

memset(high, 1, sizeof(high));

for(int i = 1; i <= n; ++i) solve(a[i]);

printf("%d\n%d\n", maw, top);

// system("pause");

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#define MAX\_N 10000

int dp[MAX\_N]={0};

int i=0;

int dao[MAX\_N];

int max(int a,int b)

{

return a>b?a:b;

}

void solve1()

{

int k,j;

int res = 0;

int n = i;

for(j=0; j<n; j++)

{

dp[j] = 1;

for(k=0; k<j; k++)

if(dao[k] > dao[j])

dp[j] = max(dp[j],dp[k]+1);

res = max(res , dp[j]);

}

printf("%d\n",res);

}

void solve2()

{

int k,j;

int res = 0;

int n = i;

for(j=0; j<n; j++)

{

dp[j] = 1;

for(k=0; k<j; k++)

if(dao[k] < dao[j])

dp[j] = max(dp[j],dp[k]+1);

res = max(res , dp[j]);

}

printf("%d\n",res);

}

int main()

{

char q;

int s=0;

while(q=getchar())

{

if(q>='0' && q<='9')

s = s\*10+q-'0';

else if(q == ' ')

{

dao[i] = s;

i++;

s = 0;

}

else

break;

}

dao[i] = s;

i++;

solve1();

solve2();

return 0;

}

本题的Java参考代码如下：

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String args[]) throws IOException{

BufferedReader bf=new BufferedReader(new InputStreamReader(System.in));

String s=bf.readLine();

String[] ss=s.split(" ");

int[] numa=new int[ss.length];

int[] numb=new int[ss.length];

int[] numc=new int[ss.length];

for(int i=0;i<ss.length;i++){

numa[i]=Integer.parseInt(ss[i]);

numb[i]=1;

numc[i]=1;

}

int a1=Integer.MIN\_VALUE;

int a2=Integer.MIN\_VALUE;

for(int i=0;i<numa.length;i++){

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

if(numa[i]<numa[j]&&numb[i]<numb[j]+1){

numb[i]=numb[j]+1;

}

a1=Math.max(a1, numb[i]);

}

}

for(int i=0;i<numa.length;i++){

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

if(numa[i]>numa[j]&&numc[i]<numc[j]+1){

numc[i]=numc[j]+1;

}

}

a2=Math.max(a2, numc[i]);

}

System.out.println(a1);

System.out.println(a2);

}

}