算法设计与分析第三章作业

班级:计算机72班 姓名:徐子越 学号:2174410455

第一题:

O(n²)复杂度有如下两种算法:

1. 先对给定数组进行排序,可以在O(nlogn)下完成.再将已排序数组和原数组之间执行最长公共子序列算法即可得到最长递增子序列;
2. 对给定数组从后向前进行Max(p)算法:

If p is n then Max(p) = 1

Else Max(p) = max{Max(p+1~n)} +1

遍历完成之后将Max最大的值输出即得到最长递增子序列长度

再将Max从大到小间隔为1所对应的数组中的值输出即可得到具体的子序列

两种算法程序如下:

#算法1>python

a = [8, 3, 5, 6, 7, 9, 11, 1, 13, 10]  
n = len(a)  
b = a.copy()  
b.sort()  
r = [[0]\*n for i in range(n)]  
s = [[0]\*n for i in range(n)]  
for i in range(n):  
 r[i][0] = 0  
for i in range(n):  
 s[0][i] = 0  
for i in range(1,n):  
 for j in range(1,n):  
 if a[i] is b[j]:  
 r[i][j] = r[i-1][j-1] + 1  
 s[i][j] = 1  
 elif r[i-1][j]>=r[i][j-1]:  
 r[i][j] = r[i-1][j]  
 s[i][j] = 2  
 else:  
 r[i][j] = r[i][j-1]  
 s[i][j] = 3  
print(r[n-1][n-1])

//算法2>c++

#include <iostream>  
using namespace std;  
int a[] = {1,3,2,4,6,5,8,7};  
const int n = 8;  
int b[n] = {1};  
int max(int p){  
 if(p==n-1){  
 b[p] = 1;  
 }  
 else{  
 int temp = 0;  
 for (int i = p+1; i < n; i++) {  
 if ((a[p]<a[i])&&(temp<b[i]))  
 temp = b[i];  
 }  
 b[p] = temp+1;  
 }  
 return 0;  
}  
int main(){  
 for(int i = n-1;i>=0;i--){  
 max(i);  
 }  
 int temp = 0;  
 for(int j = 0;j<n;j++){  
 if(temp<b[j])  
 temp = b[j];  
 }  
 printf("max len:%d index as fellow:",temp);  
 int res = 0;  
 for(int i = 0;i<n;i++){  
 if((b[i]==temp)||((b[i]==(temp-1))&&(res<a[i]))){  
 cout<<a[i]<<" ";  
 res = a[i];  
 temp--;  
 }  
 }  
 return 0;  
}



第二题:

将n这数字按照一定的方法进行排序,即将n个数字从左到右依次添加进新建的临时数组中,最后检查那个临时数组最长即可,但是这样复杂度为n²;

进一步优化,我们将这最多可能有n个的临时数组全部和成为一个,在这个数组中,每一位数值代表可以达到该位所在长度的子序列,用二分法替换插入.那么这个临时数组最右边所在的值就是答案长度.复杂度为O(nlogn),在得到数字后用O(n)复杂度得到子序列

C++代码如下:

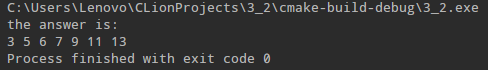
#include <iostream>  
#include <vector>

#include <cmath>  
using namespace std;  
vector<int> r;  
const int a[] = {8, 3, 5, 6, 7, 9, 11, 1, 13, 10};  
const int n = sizeof(a) / sizeof(int32\_t);  
int max(){  
 for(int i = 0; i < n; i++)  
 {  
 if(r.size() == 0)r.push\_back(a[i]);  
 else{  
 BS(r, a[i]);  
 }  
 }  
 return 0;  
}  
int main() {  
 max();  
 int rslt = r.size();  
 if(rslt==1&&r[0]==a[n-1])  
 cout<<rslt-1<<endl;  
 vector<int> rst;//储存最后最长递增子序列  
 rst.push\_back(r[rslt-1]);  
 //vector<int>::iterator it1;  
 cout<<"the answer is:";  
 //for(it1=r.begin();it1!=r.end();it1++)  
 //{  
 // cout<<\*it1<<" ";  
 //}  
 cout<<endl;  
 for(int i = n-1; 0<=i; i--){  
 if(rslt-2>=0 && a[i]>=r[rslt-2] && r[rslt-1]>a[i]){  
 int temp = r[rslt-2];  
 if (a[i]>=temp)  
 temp = a[i];  
 if(a[i] == r[rslt-2]){  
 rst.push\_back(temp);  
 rslt=rslt-1;  
 }  
 }  
 }  
 vector<int> reverse;  
 for(int i = rst.size()-1;i>=0; i--){  
 reverse.push\_back(rst[i]);  
 }  
 vector<int>::iterator it;  
 for(it=reverse.begin();it!=reverse.end();it++)  
 {  
 cout<<\*it<<" ";  
 }  
 return 0;  
}

int find(vector<int>&a, int start, int end, int f){  
 int middle = round((start+end)/2);  
 if(end-start>=1) {  
 if ((a[middle] > f) && (a[middle]-1 < f)) {  
 a[middle] = f;  
 return 0;  
 } else {  
 if (f < a[middle])find(a, start, middle - 1, f);  
 else if (a[middle] < f)find(a, middle + 1, end, f);  
 }  
 }  
 if(0<=end-start&&end-start<1){  
 if(a[middle]<f){  
 if(middle+1==a.size())  
 a.push\_back(f);  
 }  
 else if(f<a[middle])a[middle] = f;  
 return 0;  
 }  
}  
int BS(vector<int> &a, int f){  
 int end = a.size()-1, start = 0;  
 find(a, start, end, f);  
  
 return 0;  
}

程序输入为{8, 3, 5, 6, 7, 9, 11, 1, 13, 10}

输出如下:



第三题:

首先用单词去往第一行放，这一行会有一个空格数，然后再去计算剩下的单词的最少空格数，当前空格数+剩下单词的最少空格数就是最终的解，然而子问题也是最少空格数的求解。再进一步看，求最少空格数，我们需要安排前边的某几个单词放置于当前行，然后计算剩余的单词的最少空格数，所以我们需要求出到底在该行放置多少个单词，并且字符数量不超过行字符数量的限制的情况下，加上剩余单词的最少空格数量，取所有情况的一个最小值，那么这个问题就得到了一个最优值。子问题和父问题具有相同的性质，即最优性，不同于穷举方法。这里我们计算得到的值都是最优的，而且父问题的最优值不会影响子问题的最优值，子问题完全由子问题来确定，这就是没有后效性。 另外，由于每一层可以放不同数量的单词，我们需要分别计算，找一个最小的，在子问题中，大范围的子问题的子问题会包含小范围的子问题，就是有重叠的问题。

#include <iostream>

#include <cmath>

#define MAX 100

#define INF 10000

using namespace std;

int len(int s[],int i,int j)

{

int t=0;

for (int k=i;k<=j;k++)

{

t+=s[k];

}

return t;

}

int main()

{

int n,M;

cin>>n>>M;

int \*s=new int[n+1];

int \*c=new int[n+1];

int \*pos=new int[n+1];

int i,j;

s[0]=0;

c[0]=0;

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

{

scanf("%d",&s[a]);

c[a]=INF;

}

int

extra[MAX][MAX],lc[MAX][MAX];

int t=0;

int k=0;

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

{

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

{

extra[i][j]=M+i-j-len(s,i,j);

if (extra[i][j]<0)

lc[i][j]=INF;

else if (extra[i][j]>=0&&j==n)

lc[i][j]=0;

else lc[i][j]=pow(extra[i][j],3);

t=c[i-1]+lc[i][j];

if (t<c[j])

{

c[j]=t;

pos[k]=j;

}

}

k++;

}

cout<<c[n]<<endl;

for (i=0;i<k&&pos[i]>0;i++)

{

if (pos[i+1]!=pos[i])

{

cout<<pos[i]<<" ";

}

}

delete []s;

delete []c;

return 0;

}