4-1会场安排问题

#include<iostream>

using namespace std;

#define M 50 //最大活动数

struct Active

{

int s;//开始时间

int e;//结束时间

int no;//预安排会场号

}a[M];

//两元素交换位置

void swap(Active &a,Active &b)

{

Active t;

t=a;

a=b;

b=t;

}

void main()

{

int k;

int i,j;

cout<<"输入待安排活动数:"<<endl;

cin>>k;

cout<<"输入待安排活动的开始时间和结束时间:"<<endl;

//输入活动时间

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

{

cin>>a[i].s>>a[i].e;

a[i].no=0;

}

//活动时间排序

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

{

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

{

if(a[i].s>a[j].s)

swap(a[i],a[j]);

if(a[i].s==a[j].s)

{

if(a[i].e>a[j].e)

swap(a[i],a[j]);

}

}

}

int sum=1;//使用的会场数初始化

int n;

a[1].no=sum;

for(i=2;i<=k;i++)

{

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

{

if(a[n].no!=0&&a[n].e<=a[i].s)

{

a[i].no=a[n].no;

a[n].no=0;//已经安排过的活动就不再比较

break;

}

}

if(n==i)

{

sum+=1;

a[i].no=sum;

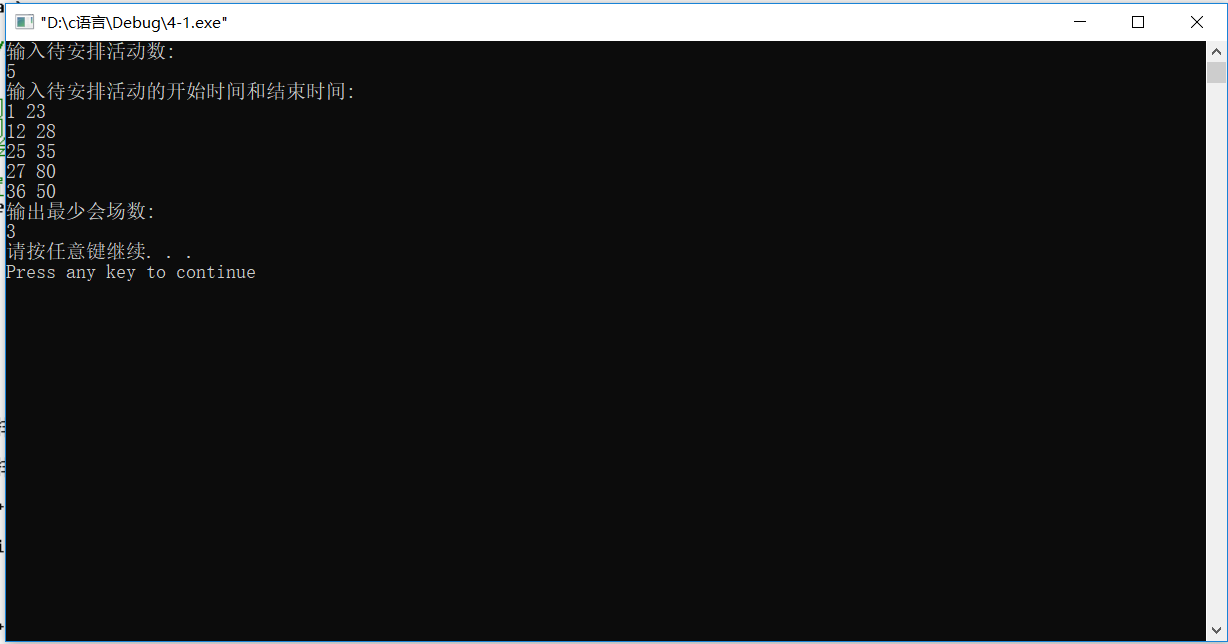
}

}

cout<<"输出最少会场数:\n"<<sum<<endl;

system("pause");

}



4-4磁盘最优存储问题

#include <iostream>

#include <algorithm>

#include <cstdio>

using namespace std;

int n;

struct node{

int a,b;

}p[1005];

bool cmp(node a,node b){

return a.a\*a.b<b.a\*b.b;

}

int main()

{

cin>>n;

int sum=0;

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

cin>>p[i].a>>p[i].b;

sum+=p[i].b;

}

sort(p,p+n,cmp);

double t=0,tmp=0;

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

tmp=0;

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

tmp+=p[j].a\*(p[j].b\*1.0/sum);

}

t+=tmp;

}

printf("%.4lf\n",t);

return 0;

}

4-7多处服务最优次序问题

#include<stdio.h>

#include<stdlib.h>

main()

{

int \*\*window,\*timewindow,\*array,num,serve,i,j,k,temp;

double min;

printf("请输入等待服务人数\n");

scanf("%d",&num);

printf("请输入服务窗口数\n");

scanf("%d",&serve);

array = (int \*)malloc((num + 1) \* sizeof(int));

timewindow = (int \*)malloc((serve + 1) \* sizeof(int));

window = (int \*\*)malloc((serve + 1) \* sizeof(int \*));

for(i = 0; i <= serve;i++)

window[i] = (int \*)malloc((num + 1) \* sizeof(int \*));

printf("请依次输入服务等待时间\n");

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

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

for(i = 0; i <= serve;i++)

{

timewindow[i] = 0;

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

window[i][j] = 0;

}

for(i = 1; i <= num; i++)//排序

{

for(k = i,j = i + 1; j < num; j++)

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

k = j;

temp = array[k];

array[k] = array[i];

array[i] = temp;

}

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

{

for(k = 1,j = 2; j <= serve;j++)

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

k = j;

timewindow[k] +=array[i];

window[k][++window[k][0]] = array[i];

}

for(min = 0.0,i = 1; i <= serve;i++)

for(j = 1; j <= window[i][0];j++)

min += window[i][j] \* (window[i][0] - j + 1);

min /= num;

printf("\n此方案最优服务次序为%f\n",min);

getch();

}

5-2最小长度电路板排列问题

#include<stdio.h>

#define n 8

#define m 5

int arr[n][m]={

{1,1,1,1,1},

{0,1,0,1,0},

{0,1,1,1,0},

{1,0,1,1,0},

{1,0,1,0,0},

{1,1,0,1,0},

{0,0,0,0,1},

{0,1,0,0,1}

};

int a[n],opt[n];

int mincount=99999;

int tempcount;

void traceback(int t)

{

int i,j;

int left,right;

int temp;

if(t==n)

{

tempcount=0;

for(i=0;i<m;i++)

{

for(j=0;j<n;j++) //最左边的

{

if(arr[a[j]][i]==1)

{

left=j;

break;

}

}

for(j=n-1;j>=0;j--) //最右边的

{

if(arr[a[j]][i]==1)

{

right=j;

break;

}

}

if(tempcount<right-left) //最大长度

{

tempcount=right-left;

}

}

if(tempcount<mincount) //最大长度的最小值

{

mincount=tempcount;

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

opt[i]=a[i];

}

return;

}

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

{

temp=a[i];

a[i]=a[t];

a[t]=temp;

traceback(t+1);

temp=a[i];

a[i]=a[t];

a[t]=temp;

}

}

int main()

{

int i;

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

a[i]=i;

traceback(0);

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

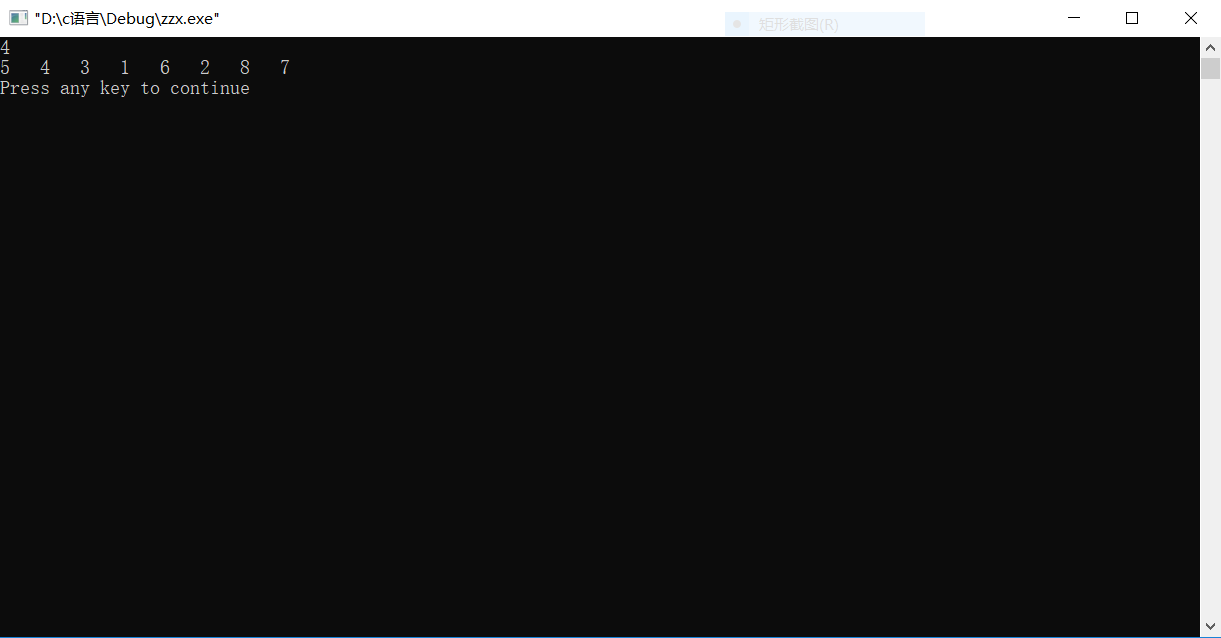
for(i=0;i<n;i++)

printf("%d ",opt[i]+1);

printf("\n");

return 0;

}



5-3最小重量机器设计问题

#include<stdio.h>

#define max 100

int cost[max][max],weight[max][max];

int n,m,d;

int current\_weight,current\_cost;

int best\_cost,best\_weight;

int array[max],best\_array[max];

void print()

{

printf("%d---%d\n",best\_weight,best\_cost);

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

printf("%d ",best\_array[i]);

printf("\n");

}

void BackTrack(int level)

{

if(current\_cost>d)

return;

if(level == n+1)

{

if(current\_cost<=d&¤t\_weight<best\_weight)

{

best\_cost = current\_cost;

best\_weight = current\_weight;

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

best\_array[i] = array[i];

}

}

else

{

int i,j,k,l;

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

{

current\_weight = current\_weight+weight[level][i];

current\_cost = current\_cost+cost[level][i];

array[level] = i;

BackTrack(level+1);

current\_weight = current\_weight-weight[level][i];

current\_cost = current\_cost-cost[level][i];

}

}

}

int main()

{

int i,j,k,l;

while(scanf("%d%d%d",&n,&m,&d)!=EOF)

{

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

{

for(j=1;j<=m;j++)

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

}

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

{

for(j=1;j<=m;j++)

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

}

best\_weight = 99999;//max\_value;

BackTrack(1);

print();

}

return 0;

}

5-8整数变换问题

#include <iostream>

using namespace std;

int k = 1;

int c = 0;

char a[100] = {'\0'};

int SelectFun(const int n, const int m, int s) //选择函数

{

if(s == 0){

return 3 \* n;

}

else{

return n / 2;

}

}

bool DeptSearch(int Dept, const int n, const int m)//深搜

{

int num;

if(Dept > k) return false;

num = n;

for(int i = 0; i < 2; i++)

{

num = SelectFun(n, m, i);

if(num == m || DeptSearch(Dept + 1,num,m)){

if(i == 0){

a[c] = 'f';

}

else{

a[c] = 'g';

}

c ++;

return true;

}

}

return false;

}

int main()

{

int m, n, Dept = 1;

cin >> m >> n;

k = 1;

while( !DeptSearch(1, m, n) )

{

k ++;

}

cout << k << endl;

int i = 0;

for(i = 0; i < c; i ++){

cout << a[i];

}

return 0;

}

