1. 独立任务最优调度

#include <iostream>

using namespace std;

bool\*\*\* store;

int main()

{

int n,\*array\_a,\*array\_b;

cin>>n;

array\_a=(int\*)malloc(n\*sizeof(int));

if(!array\_a)

exit(-1);

array\_b=(int\*)malloc(n\*sizeof(int));

if(!array\_b)

exit(-1);

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

cin>>array\_a[i];

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

cin>>array\_b[i];//输入作业数量n和两台机器的处理时间

int sum\_a=0;

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

sum\_a=sum\_a+array\_a[i];

int sum\_b=0;

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

sum\_b=sum\_b+array\_b[i];

store=(bool\*\*\*)malloc((sum\_a+1)\*sizeof(bool\*\*));

if(!store)

exit(-1);

for(int i=0;i<=sum\_a;++i)

{

store[i]=(bool\*\*)malloc((sum\_b+1)\*sizeof(bool\*));

if(!store[i])

exit(-1);

}

for(int i=0;i<=sum\_a;++i)

{

for(int j=0;j<=sum\_b;++j)

{

store[i][j]=(bool\*)malloc(n\*sizeof(bool));

if(!store[i][j])

exit(-1);

}

}

for(int i=0;i<=sum\_a;++i)

{

for(int j=0;j<=sum\_b;++j)

{

if(i>=array\_a[0]||j>=array\_b[0])

store[i][j][0]=true;

else

store[i][j][0]=false;

}

}

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

{

for(int i=0;i<=sum\_a;++i)

{

for(int j=0;j<=sum\_b;++j)

{

if(i-array\_a[k-1]>=0)

store[i][j][k]=store[i-array\_a[k-1]][j][k-1];

if(j-array\_b[k-1]>=0)

store[i][j][k]=store[i][j][k]||store[i][j-array\_b[k-1]][k-1];

}

}

}

int min=1000000;

for(int i=0;i<=sum\_a;++i)

{

for(int j=0;j<=sum\_b;++j)

{

if(store[i][j][n-1]==true)

{

int max;

if(i>j)

max=i;

else

max=j;

if(min>max)

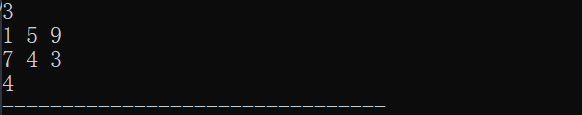
min=max;

}

}

}

cout<<min;



return 0;

}

1. 最大长方体问题

#include <iostream>

using namespace std;

int maxSum1D(int\* array1D,int p)//求一维数组的最大子段和

{

int sum=0,d=0;

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

{

if(d>0)

d=d+array1D[i];

else

d=array1D[i];

if(sum<d)

sum=d;

}

return sum;

}

int maxSum2D(int\*\* array2D,int n,int p)//求二维数组的最大子矩阵

{

int \*array1D,sum=0;

array1D=(int\*)malloc(p\*sizeof(int));

if(!array1D)

exit(-1);

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

{

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

{

for(int k=0;k<p;++k)

{

int S=0;

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

S=S+array2D[l][k];

array1D[k]=S;

}

int tem=maxSum1D(array1D,p);

if(sum<tem)

sum=tem;

}

}

return sum;

}

int maxSum3D(int\*\*\* array3D,int m,int n,int p)//求三维数组的最大子矩阵

{

int \*\*array2D,sum=0;

array2D=(int\*\*)malloc(n\*sizeof(int\*));

if(!array2D)

exit(-1);

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

{

array2D[i]=(int\*)malloc(p\*sizeof(int));

if(!array2D[i])

exit(-1);

}

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

{

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

{

for(int i1=0;i1<n;++i1)

{

for(int j1=0;j1<p;++j1)

{

int S=0;

for(int k1=i;k1<=j;++k1)

S=S+array3D[k1][i1][j1];

array2D[i1][j1]=S;

}

}

int tem=maxSum2D(array2D,n,p);

if(sum<tem)

sum=tem;

}

}

return sum;

}

int main()

{

int m,n,p,\*\*\*array3D;

cin>>m;

cin>>n;

cin>>p;

array3D=(int\*\*\*)malloc(m\*sizeof(int\*\*));

if(!array3D)

exit(-1);

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

{

array3D[i]=(int\*\*)malloc(n\*sizeof(int\*));

if(!array3D[i])

exit(-1);

}

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

{

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

{

array3D[i][j]=(int\*)malloc(p\*sizeof(int));

if(!array3D)

exit(-1);

}

}

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

{

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

{

for(int k=0;k<p;++k)

{

cin>>array3D[i][j][k];

}

}

}

cout<<maxSum3D(array3D,m,n,p);

return 0;

}

