**问题描述：把长L1，L2…Ln的n个程序放在磁带T1，T2上，并且按照最大检索时间最小值的方式存放。**

**问题分析：这个问题，类似于货船的最优装载问题，是一个特殊的0-1背包问题，通过遍历可能的结果，找出最优值。当然，如果遍历所有，那就那就是穷举法了，所以我们要辅以贪心思想和限界剪枝来减少时间复杂度。在这里，它能不能装下Ln取决于它的前一个状态，而如果我们每一个状态都选取它的最优的状态，就可以得到一个最优的状态集合，那么结果也会是最优的。那么，在这里，我们先界定“最优”的标准——最少检索时间。**

**则：设X1=l表示将Li存放在T1上，且且检索时间最短，那么：**

-<=0,<=,<=1/2

**T1的检索时间取最大值得：**

**Max**

<=1/2

**具体实现操作为：**

#include<iostream>

#include<stdlib.h>

#include<cstring>

using namespace std;

int max(int a,int b)

{

    return a > b? a:b;

}

int  min(int a,int b)

{

    return b > a? a:b;

}

void Knapsack(int \*w,int c,int n,int \* \* arrayC)

{

    int  Max=min(w[n]-1,c);

    for(int j=0;j<= Max;j++)

        arrayC[n][j]=0;

    for(int k=w[n];k<=c;k++)

        arrayC[n][k]=w[n];

    for(int i=n-1;i>1;i--)

    {

        Max=min(w[i]-1,c);

        for(int j=0;j<=Max;j++)

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

        for(int l=w[i];l<=c;l++)

            arrayC[i][l]=max(arrayC[i+1][l],arrayC[i+1][l-w[i]]+w[i]);

    }

   arrayC[1][c]=arrayC[2][c];

    if(c>=w[1])

        arrayC[1][c]=max(arrayC[1][c],arrayC[2][c-w[1]]+w[1]);

}

void Traceback(int \*\*arrayC, int \*w, int c, int n, int \*x)

{

    for(int i = 1; i < n; i++)

    {

        if(arrayC[i][c] == arrayC[i+1][c])

            x[i] = 0;

        else

        {

            x[i] = 1;

            c-=w[i];

        }

        x[n] = (arrayC[n][c])?1:0;

    }

}

void distribution(int \* Array, int \* ArrayA, int \* ArrayB,int \*\*arrayC,int sum,int\*x)

{

    Knapsack(Array,sum,Array[0],arrayC);

    Traceback(arrayC,Array,sum,Array[0],x);

    ArrayA[0] = 0;

    ArrayB[0] = 0;

    for(int i = 1; i <= Array[0]; i++)

    {

        if(x[i] == 1)

        {

            ArrayA[++ArrayA[0]] = Array[i];

        }

        else

        {

            ArrayB[++ArrayB[0]] = Array[i];

        }

    }

}

void main()

{

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    cout<<endl;

    int n;

    int \*array;

    int \*arrayA;

    int \*arrayB;

    int \*\*arrayC;

    int \*x;

    int sum;

    sum= 0;

    cout<<""<<endl;

    cin>>n;

    cout<<endl;

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

    arrayA = (int \*)malloc((n+1)\*sizeof(int));

    arrayB = (int \*)malloc((n+1)\*sizeof(int));

    x= (int \*)malloc((n+1)\*sizeof(int));

    arrayC=new int\*[n+1];

    array[0] = n;

    cout<<""<<endl;

    for(int i=1; i<=array[0]; i++)

    {

        cout<<""<<i<<":";

        cin>>array[i];

        sum +=array[i];

    }

    sum = sum/2;

    cout<<endl;

    for(int j=0;j<=n;j++)

    {

       arrayC[j]=new int[sum+1];

       memset(arrayC[j],0,(sum+1)\*sizeof(int));

    }

    distribution(array,arrayA,arrayB,arrayC,sum,x);

    cout<<"A分配的程序有 "<<endl;

    for(int k=1; k<=arrayA[0]; k++)

    {

        cout<<arrayA[k]<<"  ";

    }

    cout<<endl;

    cout<<"B分配的程序有 "<<endl;

    for(int l=1; l<=arrayB[0]; l++)

    {

        cout<<arrayB[l]<<"  ";

    }

    cout<<endl;

}