7-1 修理牧场 （25 分）

农夫要修理牧场的一段栅栏，他测量了栅栏，发现需要N块木头，每块木头长度为整数L_i个长度单位，于是他购买了一条很长的、能锯成N块的木头，即该木头的长度是L_i的总和。

但是农夫自己没有锯子，请人锯木的酬金跟这段木头的长度成正比。为简单起见，不妨就设酬金等于所锯木头的长度。例如，要将长度为20的木头锯成长度为8、7和5的三段，第一次锯木头花费20，将木头锯成12和8；第二次锯木头花费12，将长度为12的木头锯成7和5，总花费为32。如果第一次将木头锯成15和5，则第二次锯木头花费15，总花费为35（大于32）。

请编写程序帮助农夫计算将木头锯成N块的最少花费。

输入格式:

输入首先给出正整数N（$\leq 10^4$），表示要将木头锯成N块。第二行给出N个正整数（≤ 50），表示每段木块的长度。

输出格式:

输出一个整数，即将木头锯成N块的最少花费。

输入样例:

```
8
4 5 1 2 1 3 1 1
```

输出样例:

```
49
```

```
#include<stdio.h>

#define MAX

typedef struct Node{
     int data;
     int left,parent,right;
}Node;

Node* create(int num);
int select(Node* tree,int num,int* s1,int* s2);

int main(){
   int num;
   scanf("%d",&num);
   Node* tree=create(num);
   int count=0;
   for(int i=num+1;i<num*2;i++){
     count+=tree[i].data;
   }
   printf("%d",count);
}
```

```
Node* create(int num){
    Node *tree;
    int nums=num*2-1;
    tree=(Node*)malloc(sizeof(Node)*(nums+1));
    for(int i=0;i<nums;i++){
        tree[i].data=0;
        tree[i].left=0;
        tree[i].right=0;
        tree[i].parent=0;
    }


    for(int i=1;i<num+1;i++){
        scanf("%d",&tree[i].data);
    }

    for(int i=num+1;i<=nums;i++){
        int s1,s2;
        select(tree,i,&s1,&s2);
        tree[i].data=tree[s1].data+tree[s2].data;
        tree[i].left=s1;
        tree[i].right=s2;
        tree[s1].parent=i;
        tree[s2].parent=i;
    }
    return tree;
}

int select(Node* tree,int num,int* s1,int* s2){
    int i=1;
    int min1,min2;

    while(tree[i].parent!=0&&i<num){
        i++;
    }
    *s1=i;
    min1=tree[i].data;
    i++;

    while(tree[i].parent!=0&&i<num){
       i++;
    }
    *s2=i;
    min2=tree[i].data;
    i++;
```

```
    if(min1>min2){
        int temp=min1;
        min1=min2;
        min2=temp;
        int t=*s1;
        *s1=*s2;
        *s2=t;
    }

    for(i;i<num;i++){
        if(tree[i].parent!=0){
            continue;
        }
        if(tree[i].data>=min1&&tree[i].data<min2){
            *s2=i;
            min2=tree[i].data;
        }else if(tree[i].data<min1){
            min2=min1;
            min1=tree[i].data;
            *s2=*s1;
            *s1=i;
        }
    }

}
```