1)

贪心策略，先计算所有站的 add[i] = gas[i] – cost[i],该值如果为正表示经过这站，到下站之后时还有结余的油量，该值为负表示只靠刚才站的油量不够到下一站。然后从队伍尾开始选取一点作为起始站，如果该站的左边的 add 值为正，则起始站往左推移，直到左边的站点的 add 值为负，然后从选定好的起始节点开始，定义一个 sum = 0 值，表示当前的油量，并记录该起点，每往后推一站则 sum += 当前站的 add 值，如果 sum >= 0 并且经过站的总数>=N 则表示，记录下来的起始站点为所求。若 sum < 0 则选取当前站点为起始值，并将 sum = 0。如果设定的起始值的位置已经绕最开始记录的起点值一周，则输出-1,并结束程序。

优化子结构，设 i 为起始站,则对于任何一个 j 有 sum(i – j) = add[i]+add[i+1]+add[i+2]..add[j-1]+add[j] >= 0,并且 add[i – 1] < 0;

贪心选择性，证明：假设当前 i 作为起点满足可行解，如果左边站点的 add 值 >= 0,则以左边的 i-1 作为起点同样满足可行解，所以以 i 作为起点的不是最优解，i-1 为起点的是更优解。

伪代码：

```
int biaoji_head;
int head = N – 1;
bool flg = false;
int curr = 0;
int sum = 0;
int total = 0;
while(add[head]  >= 0){
    head--;
}
head++;
head = head % N;
biaoji_head = head;
curr = (head + 1) % N;
do{
    sum  += add[ (curr – 1) % N];
    total++;
    if(sum >= 0 && total >= N){print(“%d”,head);break;}
    if(sum >= 0){
        curr++;
```

```
        }
        else{
            sum = 0;
            head  = curr;
            total = 0;
            curr++;
            flg = true;
        }
        if(flg && head == biaoji_head){print("-1");break;}
    }while(curr != head);
```

时间复杂度分析:o(n);

2)

贪心策略：

以最左和最右两个板子为界限，从中间的所有板子中选取最高的板子，如果最高的板子高于界限两板的最小板但是又低于最高板，设 ai < aj < ak 则该区间的最大水量是 H[I] * (j – I) +H[j] * (k – j),如果中间最高的板子小于边界两个板子的任意一个，则 aj < ai < ak,区间最大水量是 H[i] * (k – I),如果中间板子的最高板高于两个边界板，aj > ai,ak 则将区间(i,k)分成两个子区间 H[(i,j)]+H[(j,k)]继续迭代下去。

优化子结构，max(i,k) = {

H[j]为(i,k)中间最大值。

H[i]*(j-i)+H[j]*(k-j),ai < aj < ak

H[i] * (k – I),aj<ai,ak

H[(i,j)]+H[(j,k)],aj > ai,ak

}

贪心选择性：

证明，对于区间(i,k)选取 j1 作为中间隔板，如果存在 j2 板，高于 j1 板。则左右边界到 j2 板的储备水量一定大于等于左右边界到 j1 板的储备水量。所以选取 j1 板不是最优解。若是 j 板为最高板，则选取 j 板作为中间板一定是最优解。

伪代码：

```
int cal(int I ,int j){
    int max = 0;
    int loc = i;
```

```
        for(int k = I + 1; k < j; k++)
        {
                if(H[k] >= max){
                max = H[k]l
                loc = k;
                }
        }
        if(max <= H[i] && max <= H[j])return (H[i] < H[j] ? H[i] : H[j])*(j – I);
        else if(max > (H[i] < H[j] ? H[i] : H[j]) && max <= (H[i] > H[j] ? H[i] : H[j]) )
        return {边界小板*（边界小板到 loc 距离）+ max*（边界大板到 loc 距离）}
        else{
            cal(i,loc) + cal(loc,j);
        }
    }
```

时间复杂度分析，最坏 o(n^2);

最好情况 o(1)

3)

贪心策略：A，B 各排个序（同大到小或者同小到大），然后一一对应就行。

优化子结构与贪心选择性证明：

先对 a,b 数组排序。

设 A 中的最小值为 A，则 A 中所有的值可以这样表示,

A0 = A + a0,A1 = A + a1,A2 = A + a2..

同理 B 数组也可以这样表示

B0 = A + b0,B1 = A +b1,B2 = A + b2..

|Ai – Bj| = |ai – bj|

其中 ai >= 0,a0 = 0,bj 有小于 0 的也有大于 0 的。不妨设当前的搭配是 a0-bj,b0-ai(组合一)。是最优解，我们考虑另外一种组合 a0-b0,bj-ai(组合二)。

情况一，如果 b0 <= 0, bj < 0:

sum(组合一) = |0 – bj| + |ai – b0| = ai – b0 – bj;

sum(组合二) = |0 – b0| + |ai – bj| = ai – b0 – bj;两者相等。

情况二，如果 $b0 <= 0, bj > 0$:

sum(组合一) = $|0 – bj| + |ai – b0| = ai – b0 + bj$;

sum(组合二) = $|0 – b0| + |ai – bj| = |ai – bj| – b0$;sum(组合二) < sum(组合一)

情况三，如果 $b0 >0,bj > 0$;

sum(组合一) = $|0 – bj| + |ai – b0| = |ai - b0| + bj$;

sum(组合二) = $|0 – b0| + |ai – bj| = |ai – bj| + b0$;

不好比较。

此时有两种思路,第一种不妨取 A，B 数组中最小值中更大的那个作为基准，比如说 A 中最 小值大于 B 中最小值，这样 $a0 = 0$, $b0 <= 0$,反之如果排好序后,$B0 > A0$，则取 B0 为基准,$A0 = B + a0$,$Ai = B +ai$;$Bi = B + bi$,这样 $a0 <= 0.b0 = 0$。这样的话就只会出现情况一和情况二，足够证明 sum(组合二) <= sum(组合一),a0-b0,ai-bj 组合是最优解 。

另一种证明方式是强行比较情况三,

(1)如果 $ai<=b0<=bj$,

sum(组合一) = $|0 – bj| + |ai – b0| = b0 -ai+ bj$;

sum(组合二) = $|0 – b0| + |ai – bj| = bj - ai + b0$;两者相等。

(2)如果 $b0<=ai<=bj$,

sum(组合一) = $|0 – bj| + |ai – b0| = ai - b0+ bj >= bj$;

sum(组合二) = $|0 – b0| + |ai – bj| = bj - ai + b0 <= bj$;sum(组合二) <= sum(组合一) 。

(3)如果 $b0<=bj<=ai$;

sum(组合一) = $|0 – bj| + |ai – b0| = b0+ bj - ai$;

sum(组合二) = $|0 – b0| + |ai – bj| = bj - ai + b0$;两者相等。

同样可以证明 sum(组合二) <= sum(组合一) ,所以 a0-b0,ai-bj，比 a0-bj,ai-b0 更优。

伪代码：

```
qsort(A);
qsort(B);
```

时间复杂度，o(nlgn);

4)

贪心策略，

在所有可染色的节点中，选取节点权值最大的节点染色。

证明优化子结构和贪心选择性：

对于每个染色列表集合看成一个状态(a0,a1,a2….ai,aj)，不妨设(a0-ai-1)都是必须先染的节点,那么该集合可由(a0,a1...ai-1,ai)或者(a0,a1,a2…..ai-1,aj)这两种集合推出.不妨设染了(a0-ai-1)节点后的时间是 ti = i,当前先染权值小的 aj,再染权值大的 ai，则花费的代价是 aj * i + ai * (i + 1),而显然权值大的 ai，后染权值小的 aj 花费的代价是 ai*i + aj*(i + 1)小于前者，后者是更优解。

伪代码:

```
int max;
treeNode tree;
while(未队列数量 > 0){
    for(节点 A：可以染色队列)
    {
        if(如果 A 的子节点权值 > max){
                max = 如果 A 的子节点权值;
                tree = 节点 A;
        }
    }
            计算 cost;
            节点 A 的子节点加入可以染色队列
}
```

时间复杂度，最坏情况 o(n^2);

最好情况 o(n);

5)

贪心策略，标准的 Huffman Tree。

(1) 将 w1、w2、…，wn 看成是有 n 棵树的森林(每棵树仅有一个结点)；

(2) 在森林中选出两个根结点的权值最小的树合并，作为一棵新树的左、右子树，且新树的根结点权值为其左、右子树根结点权值之和；

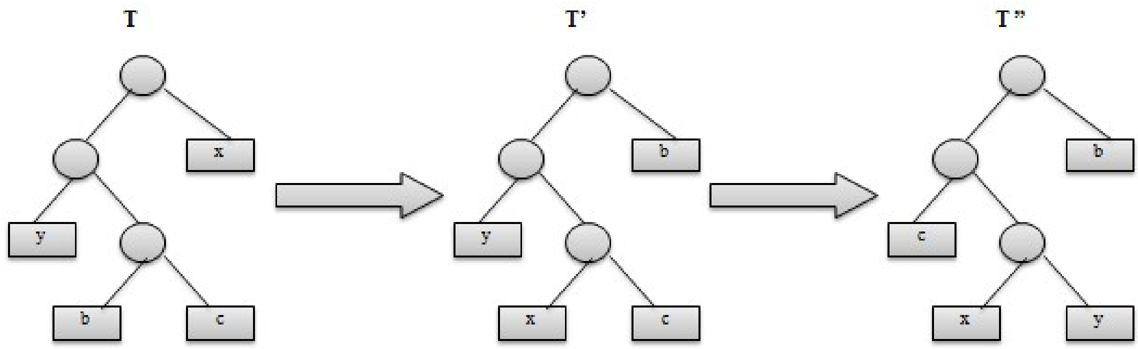
(3)从森林中删除选取的两棵树，并将新树加入森林；

(4)重复(2)、(3)步，直到森林中只剩一棵树为止，该树即为所求得的哈夫曼树。

证明贪心选择性和优化子结构：

两两合并最终一定是构成一棵二叉树，代价是每个叶子节点×路径长度然后求和。

假如得到的二叉树 T 不是最优二叉树。对 T 作适当修改后得到一棵新的二叉树 T”，在 T”中 x 和 y 是最深叶子且为兄弟，设 b 和 c 是二叉树 T 的最深叶子，且为兄弟。设 f(b)<=f(c)，f(x)<=f(y)。有 f(x)<=f(b)，f(y)<=f(c)。首先，在树 T 中交换叶子 b 和 x 的位置得到 T'，然后再树 T'中交换叶子 c 和 y 的位置，得到树 T"。如图所示：



$$B(T)-B(T')=\sum_{c\in C} f(c)d_T(c)-\sum_{c\in C} f(c)d_{T'}(c)$$
$$=f(x)d_T(x)+f(b)d_T(b)-f(x)d_{T'}(x)-f(b)d_{T'}(b)$$
$$=f(x)d_T(x)+f(b)d_T(b)-f(x)d_T(b)-f(b)d_T(x)$$
$$=(f(b)-f(x))(d_T(b)-d_T(x))\geq 0$$

$$\left.\begin{array}{r} B(T'')\leq B(T')\leq B(T) \\ B(T'')\geq B(T) \end{array}\right\} B(T'')=B(T)$$

最优子结构

$$1)\quad c\in C-\{x,y\},d_T(c)=d_{T'}(c), so\ f(c)d_T(c)=f(c)d_{T'}(c)$$
$$2)\quad f(x)d_T(x)+f(y)d_T(y)=f(x)+f(y)+f(z)d_{T'}(z)$$

$$B(T)=B(T')+f(x)+f(y)$$

伪代码：

1.数组 huffTree 初始化，所有元素结点的双亲、左右孩子都置为-1；

2.数组 huffTree 的前 n 个元素的权值置给定权值 w[n]；

3.进行 n-1 次合并

　　3.1 在二叉树集合中选取两个权值最小的根结点，其下标分别为 i1,i2；

　　3.2 将二叉树 i1、i2 合并为一棵新的二叉树 k；

时间复杂度：o(nlgn)