02. 编写双向冒泡排序算法，在正反两个方向交替进行扫描，即第一趟把关键字最大的元素放在序列的最后面，第二趟把关键字最小的元素放在序列的最前面，如此反复进行。

```
void DoubleDubbleSort(int a[], int n){
    int low = 0, high = n-1;
    bool flag = true;          //若某趟冒排序未发生元素交换，说明已经有序，可以提前结束.
    while(low < high && flag){
        flag = false;
        for(int i = low; i<high; i++){
            if(a[i] > a[i+1]){
                Swap(a[i], a[i+1]);
                flag = true;
            }
        }
    }
    high --;
    for(int i = high; i > 0; i--){
            if(a[i-1] > a[i]){
                Swap(a[i-1], a[i]);
                flag = true;
            }
        }
    }
    low--;
    } // while.
}
```

03. 已知线性表按顺序存储，且每个元素都是不相同的整数型元素，设计把所有奇数移动到所有偶数前边的算法（要求时间最少，辅助空间最少）。

```
void move(int a[], int len){                    //基于快排的partition.
    int low=0, high=len-1;
    while(low<high){
        while(low<high && a[high]%2==0) high--;    //从后向前找一个奇数.
        while(low<high && a[low]%2!=0) low++;      //从前向后找一个偶数.
        if(low<high) swap(a[low], a[high]);
        low++;
        high--;
    }
}
```

04. 试重新编写考点精析中的快速排序的划分算法，使之每次选取的枢轴值都是随机地从当前子表中选择的。

```
int partition(int a[], int low, int high){
    int randPivot = low + rand() % (high-low+1);
    Swap(a[randPivot], a[low]);
    int pivot = a[low];
    int i = low;
    for(int j = low+1; j<=high; j++)
        if(a[j] < pivot)  swap(a[++i], a[j]);
    swap(a[i], a[low]);
    return i;
}
```

```
int partition(int a[], int low, int high){
    int randpivot = low + rand() % (high-low+1);
    swap(a[low], a[randpivot]);
    int pivot = a[low];
    while(low < high){
        while(low < high && a[high] >= pivot) high--;
        a[low] = a[high];
        while(low < high && a[low] <= pivot) low++;
        a[high] = a[low];
    }
    a[low] = pivot;
    return low;
}
```

05. 试编写一个算法，使之能够在数组 L[1...n] 中找出第 k 小的元素（即从小到大排序后处于第 k 个位置的元素）。

```
int findkmin(int a[], int k, int low, int high){
    int pivot = a[low];
    int i = low;
    for(int j=low+1; j<=high; j++)
        if(a[j] < pivot)
            swap(a[++i], a[j]);
    swap(a[low], a[i]);
    if(k == i) return a[i];
    if(k < i) return findkmin(a, k, low, i-1);
    return findkmin(a,k,i+1,high);
}
```

一趟划分.

0 1 2 3 4　　k=4.

3 2 1 5 4.

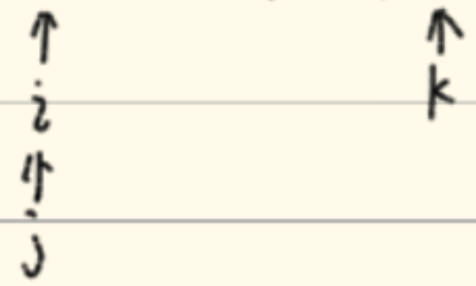
一趟: 1 2 3 5 4.　　∵k>i　∴在i右边

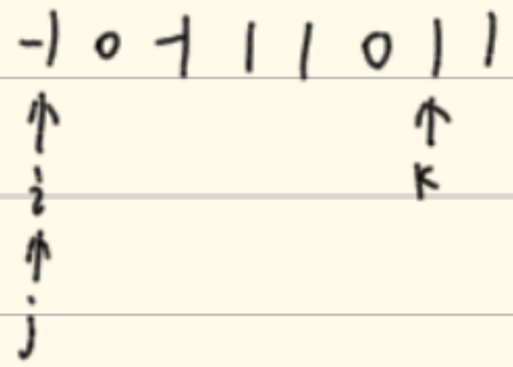
　　　　↑
　　　　i

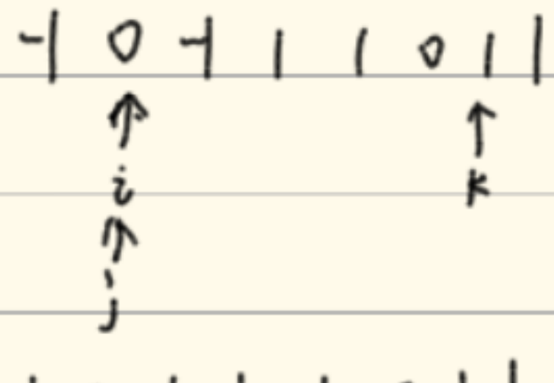
从号 06 荷兰国旗问题：设有一个仅由红、白、蓝三种颜色的条块组成的条块序列，请编写一个时间复杂度为 $O(n)$的算法，使得这些条块按红、白、蓝的顺序排好，即排成荷兰国旗图案。

红：-1，白：0，蓝：1

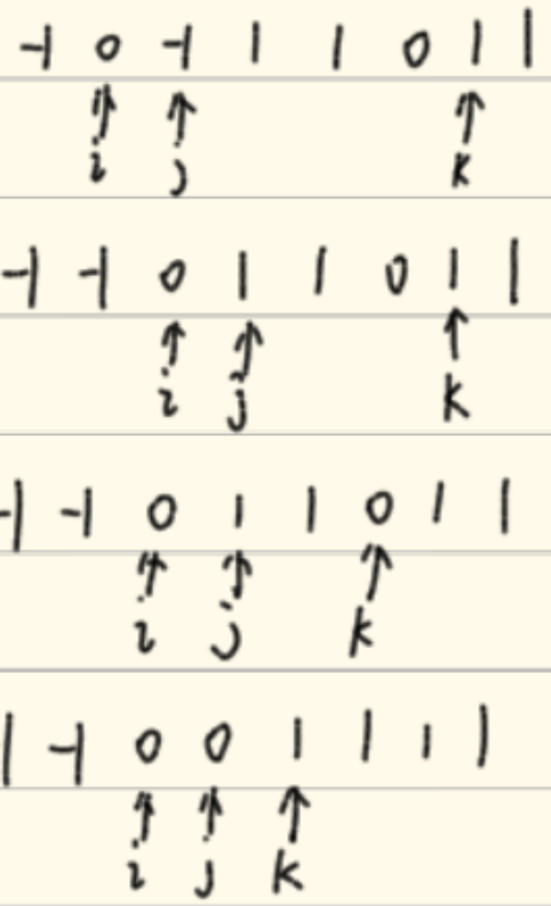
1,0,-1,1,0,1,-1　　i. j. k.

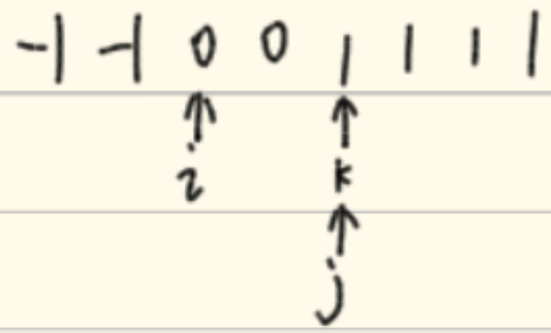


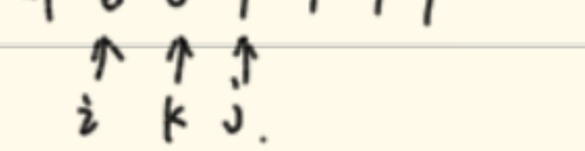




这里只k--，是防止蓝色和蓝色交换.







```
void sortflag(int flag[], int n){
    int i=0,j=0, k=n-1;
    while (j<=k){
        switch(flag[j]){
            case -1: swap(flag[i], flag[j]);
                     i++; j++; break;
            case 0: j++; break;
            case 1: swap(flag[k], flag[j]);
                    k--; break;
        }
    }//while.
}
```

07. 【2016 统考真题】已知由 n（$n \geq 2$）个正整数构成的集合 $A=\{a_k | 0 \leq k < n\}$，将其划分为两个不相交的子集 A_1 和 A_2，元素个数分别是 n_1 和 n_2，A_1 和 A_2 中的元素之和分别为 S_1 和 S_2。设计一个尽可能高效的划分算法，满足 $|n_1-n_2|$ 最小且 $|S_1-S_2|$ 最大。要求：

1）给出算法的基本设计思想。

2）根据设计思想，采用 C 或 C++语言描述算法，关键之处给出注释。

3）说明你所设计算法的平均时间复杂度和空间复杂度。

pivot=2, 2 3 6 4 5 1　　6/2 = 3

第1趟: 1 3 6 4 5

1 2 (6 4 5 3)　　low = 1 < 3.

第2趟: 6 4 5 3.

(3 4 5) 6　　low = 5 > 3.

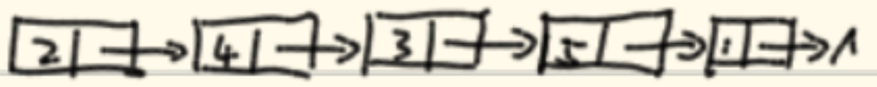
第3趟 3 4 5　　low = 2 < 3.

第4趟 4 5　　low = 3 = 3.

对于快排的划分，找到 pivot 为中间元素后计算即可.

```
int solve(int a[], int n){
    int pivot, low=0, high=n-1, mid=n/2, i=0;
    bool flag = true;
    while(flag){
        pivot = a[low];
        i = low;
        for(int j=low+1; j<=high; j++)
            if(a[j] < pivot) swap(a[++i], a[j]);
        swap(a[low], a[i]);
        if(i == mid) flag = false;
        if(i > mid) high = i-1;
        if(i < mid) low = i+1;
    }
    int cnt = 0;
    for(int j=0; j<mid; j++) cnt += a[j];
    for(int j=mid; j<n; j++) cnt -= a[j];
    return -cnt;
}
```

04. 编写一个算法，在基于单链表表示的待排序关键字序列上进行简单选择排序。



每次从链表中找一个最大的结点摘下，头插法插入链表，最后可得到一个升序链表。

```
void SelectSort(LinkList& L){
    LNode *max, *maxpre, *P, *pre, *n=L;
    L = NULL;
    while(n){
        max = p = n;
        maxpre = pre = NULL;
        while(p){
            if(P->data > max->data){
                maxpre = pre;
                max = P;
            }
            pre = P;
            P = P->next;
        }
        if(max == n)    n = n->next;
        else    maxPre->next = max->next;
        max->next = L;
        L = max;
    } // while(n)
}
```

L = NULL

2 -> 4 -> 3 -> 1 -> 1

5 -> null , L=5.

2 -> 3 -> 1 -> 1

4 -> 5 -> null , L=4.

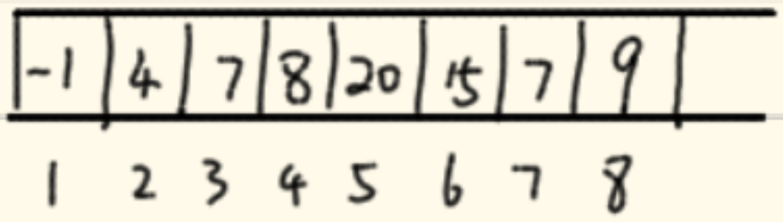
2 -> 1 -> 1

3 -> 4 -> 5 -> null , L=3.

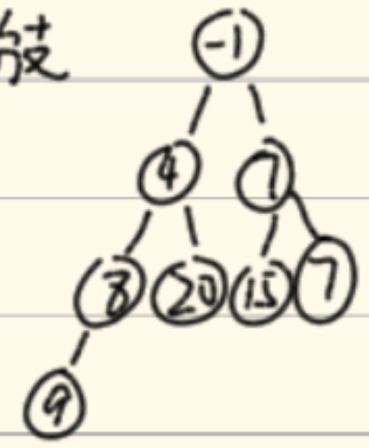
2 -> 3 -> 4 -> 5 -> null , L=2.

1 -> 2 -> 3 -> 4 -> 5 -> null. L=1.

05. 试设计一个算法，判断一个数据序列是否构成一个小根堆。




```
bool judge(int a[], int len){
    if(len % 2 == 0){   //有单分支结点.
        if(a[len/2] > a[len]) return false;  //单独判单分支
        for(int i = len/2 - 1; i >= 1; i--)
            if(a[i] > a[i*2] || a[i] > a[i*2+1])
                return false;
    }
    else {   //都是双分支.
        for(int i = len/2; i >= 1; i--)
            if(a[i] > a[i*2] || a[i] > a[i*2-1])
                return false;
    }
    return true;
}
```



将数据序列视为一棵完全二叉树.

扫描所有子结点，若有孩子小于父亲，返回false.

扫描结束返回true.

02. 设顺序表用数组A[]表示，表中元素存储在数组下标1~$m+n$的范围内，前m个元素递增有序，后n个元素递增有序，设计一个算法，使得整个顺序表有序。

1）给出算法的基本设计思想。

2）根据设计思想，采用C/C++描述算法，关键之处给出注释。

3）说明你所设计算法的时间复杂度与空间复杂度。


```
void mergeSort(int a[], int m, int n){      // 2路归并
    int *b = (int*)malloc(sizeof(int)*(m+n+1));   // 辅助数组.
    for(int i=1; i<=m+n; i++)
        b[i] = a[i];
    int i=1, j=m+1, k=1;
    while(i<=m && j<=m+n){
        if(b[i] <= b[j])  a[k++] = b[i++];
        else  a[k++] = b[j++];
    }
    while(i<=m)  a[k++] = b[i++];
    while(j<=m+n)  a[k++] = b[j++];
}
```

时间：$O(m+n)$，空间$O(m+n)$.

03. 有一种简单的排序算法，称为计数排序（count sorting）。这种排序算法对一个待排序的表（用数组表示）进行排序，并将排序结果存放到另一个新的表中。必须注意的是，表中所有待排序的关键码互不相同，计数排序算法针对表中的每个记录，扫描待排序的表一趟，统计表中有多少个记录的关键码比该记录的关键码小，假设针对某个记录统计出的计数值为 c，则这个记录在新有序表中的合适存放位置即为 c。

1）设计实现计数排序的算法。

2）对于有 n 个记录的表，关键码比较次数是多少？

3）与简单选择排序相比较，这种方法是否更好？为什么？

1).
```
void countSort(int a[], int b[], int n){
    int cnt = 0;
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
            if(a[j] == a[i]) cnt++;
    b[cnt] = a[i];
}
```

2). n^2

3). 不会更好 ① 使用了额外的空间，空间复杂度高

② 简单选择排序比较次数为 $\frac{(n-1)n}{2}$，计数排序比较次数为 n^2.

04. 设有一个数组中存放了一个无序的关键序列 $K_1, K_2, \cdots, K_n$。现要求将 K_n 放在将元素排序后的正确位置上，试编写实现该功能的算法，要求比较关键字的次数不超过 n。

一趟划分，最后元素为pivot，先从前开始.

```
int  Partition( int a[], int n){
    int i=1, j=n;
    int pivot = a[j];
    while (i<j){
        while (i<j && a[i] <= pivot) i++;
        a[j] = a[i];
        while (i<j && a[j] >= pivot) j--;
        a[i] = a[j];
    }
    a[i] = pivot;
    return i;
}
```