2-3:

```
int search(int x, int i, int j)
{ if(i==j) return i;
  int mid = (i+j)/2;
  if(a[mid]==x) return mid;
  else if(a[mid]<x) return search(x, mid+1, j);
  else if(a[mid]>x) return search(x, i, mid-1);}

void main
{ int i, j, t = search(X, 0, n-1);
  if(a[t]==x){i=t; j=t;}
  else if(a[t]>x)
    { if(t!=0){j=t; i=t-1}
      else {j=t;}}
  else if(a[t]<x)
    { i=t;
      if(t!=n-1)
        j=t+1;}
  return;
}
```

2.9.

```
int main
{ int a[]={...};
  int maxcount=0, maxnum=0;
  for(int i=0; i<n; i++)
  { a[i][j]++;
    if(a[i][j] > maxcount)
    { maxcount = a[i][j];
      maxnum = i[j]; }}
  if(maxcount > n/2)
  printf("有主元素,为...", maxnum);
  else
  printf("没有主元素");
  return 0;}
```

2.10. O(nlogn)

```
int fun(char a[], r, t)
{ if(r==t) return a[r]; int count=0, int n[];
  int n[] = fun(a[], r, (r+t)/2)
  int n[] = fun(a[], (r+t)/2+1, t)
  for(int i=1; i<=2; i++)
  { for(int j=r; j<=t; j++)
      if(a[j]=n[i])
        count++;
    if(count > (t-r)/2)
      return n[i];
      count=0; }
    return NULL;}

int main()
{ int majority = fun(a[], 0, n-1)
    if(majority)
    printf("有主元素%d", majority)
    else
    printf("没有主元素");
    return 0;}
```

O(n)算法

```
int search_most(a[], n)
{ int count=1
  int candidate = a[0];
  for(int i=1; i<n; i++)
  { if(a[i]==candidate)
      count++;
    else
      count--;
    if(count==0)
    candidate = a[i];}
    return candidate;}

int is_majority(a[], n, candidate)
{ int count=0;
  for(int i=0; i<n; i++)
    if(a[i]=candidate)
      count++
  return 2*count/n;}

int main
{ int candidate = search_most(a[], n);
  if(is_majority(a[], n, candidate))
    printf("有主元素%d", candidate);
  else
    printf("无主元素");
  return 0;}
```