ТЕМА 2. СОРТИРОВКА И ПОИСК

3.1. Сортировка массивов

Внутренняя сортировка

Внешняя сортировка

Три основных метода сортировки:

1. **Обмен**
2. **Выбор**
3. **Вставка**

3.2. Простые методы сортировки

3.2.1. Метод пузырька и шейкерная сортировка

```
void s_puz(int a[], int n)
```

```
void s_puz(int a[], int n)
{
  int i, j, t;
```

```
void s_puz(int a[], int n)
{
  int i, j, t;
  for(i=1;  i < n;  i++)
    for( j=n-1;  j >= i;  j--)
```

```
void s_puz(int a[], int n)
{
  int i, j, t;
  for(i=1; i < n; i++)
    for( j=n-1; j >= i; j--)
          if(a[j-1] > a[j])
```

```
void s_puz(int a[], int n)
{
  int i, j, t;
  for(i=1; i < n; i++)
    for( j=n-1; j >= i; j--)
        if(a[j-1] > a[j])
      {
            t = a[j-1];
       a[j-1] = a[j];
       a[j] = t;
        }
}
```

```
void s_shaker(int a[], int n)
{   int i,j,t, l=0, r=n, k;
 do
 {  for( j=n-1; j > l; j--)
     if(a[j-1] > a[j])
     {
       t = a[j-1];
        a[j-1] = a[j];
        a[j] = t;
   k=j;
      }
l=k+1;
```

```
    for(i=1; i < r; i++)
if(a[i-1] > a[i])
      {
       t = a[i-1];
        a[i-1] = a[i];
        a[i] = t;
    k=i;
}
r=k-1;
 } while(l<r);
}
```

3.2.2. Сортировка выбором

```
void s_vb(int a[], int n)
{    int  imin, i, j, t;
```

```
void s_vb(int a[], int n)
{    int imin, i, j, t;
       for (i=0;  i<n-1;  i++)
```

```
void s_vb(int a[], int n)
{    int imin, i, j, t;
       for(i=0; i<n-1; i++)
   {   imin=i;
for(j=i+1; j<n; j++)
            if (a[imin]>a[j]) imin=j;
```

```
void s_vb(int a[], int n)
{    int imin, i, j, t;
       for(i=0; i<n-1; i++)
   {   imin=i;
for(j=i+1; j<n; j++)
              if (a[imin]>a[j]) imin=j;
   if (imin != i)    {
         t = a[imin];
            a[imin] = a[i];
            a[i] = t;
                     }
     }
 }
```

3.2.3. Сортировка вставками

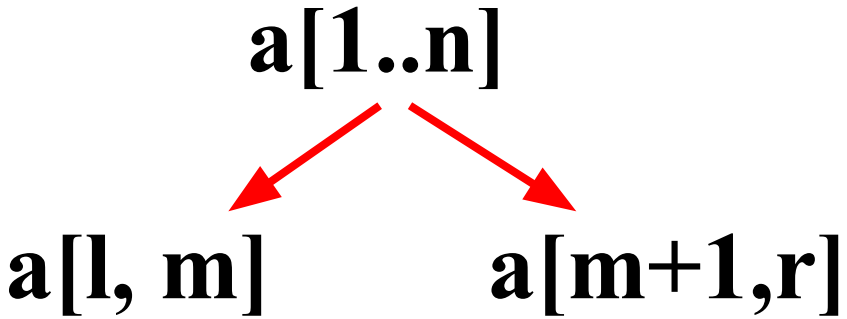
```
void s_vst(int a[], int n)
{
    int i, j, t;
for(i=1; i<n; i++)
 {
    t = a[i];
      for(j=i-1; j>=0 && t<a[j]; j--)  a[j+1]=a[j];
    a[j+1] = t;
 }
}
```

3.3. Улучшенные методы сортировки

3.3.1. Метод Шелла

```
void s_shell(int a[], int n)
{
   int i, j, t;
for(int d=3; d>0; d--)
      for(i=d; i<n; i++)
                                  {
 t = a[i];
 for(j=i-d; j>=0 && t<a[j]; j-=d) a[j+d]=a[j];
  a[j+d] = t;
                                     }
}
```

3.3.2. Сортировка слиянием



```
void slip(int l, int m, int r)
{
    int   i=l, j=m+1, k=0;
while ((i<=m) && (j<=r)) {
    if (a[i]<a[j]) {c[k]=a[i]; i++; k++; }
                else {c[k]=a[j]; j++; k++; }
                                        }
  while (i<=m) {c[k]=a[i]; i++; k++; }
  while (j<=r)   {c[k]=a[j]; j++; k++; }
for (k=0, i=l; i<=r; i++, k++) a[i]=c[k];
}
```

```
void s_sl(int l, int r)
{
if (l < r)
        {
       int m=(l+r)/2;
        s_sl(l,m);
        s_sl(m+1,r);
   slip(l,m,r);
        }
}
```

s_sl(0,n-1);

3.3.3. Метод QuickSort (быстрая сортировка или сортировка Хоара)

```
do {
   while (a[i] < x && i < right) i++;
       while (a[j] > x && j > left)   j--;

  if (i<=j) {
                t = a[i];
       a[i] = a[j];
       a[j] = t;
           i++;    j--;
          }

 }while(i<=j);
```

```
    if(left < j)     s_qsr(left, j);
     if(i < right) s_qsr(i, right);
}
```

```
void s_qs(char a[], int n)
{
   struct
     {
        int l;
        int r;
     }  stack[20];
```

```
int i, j, left, right, x, t, s=0;
stack[s].l = 0;
stack[s].r = n-1;

while (s != -1)
{
left=stack[s].l;  right=stack[s].r;
   s--;
```

```
while (left < right)
{
i=left;  j=right;    x=a[(left+right)/2];
while (i <= j)  {
while (a[i] < x) i++;
while (a[j] > x) j--;
  if (i<=j) {
        t=a[i];  a[i]=a[j];  a[j]=t;
              i++;  j--;
              }
                  }
```

```
if ((j-left)<(right-i))
        {
if (i<right) {s++;
                    stack[s].l=i;  stack[s].r=right; }
right=j;
        }
else {
if (left<j) {s++;
                    stack[s].l=left;  stack[s].r=j; }
left=i;
        }
}  }  }
```

3.5. Поиск в массиве структур

Линейный поиск

```
int p_lin1(int a[],int  n, int x)
```

```
int p_lin1(int a[],int  n, int x)
{
for(int i=0; i < n; i++)
```

```
int p_lin1(int a[],int  n, int x)
{
for(int i=0; i < n; i++)
        if (a[i]==x) return i;
```

```
int p_lin1(int a[],int  n, int x)
{
for(int i=0; i < n; i++)
        if (a[i]==x) return i;
   return -1;
}
```

```
int p_lin1(int a[],int  n, int x)
{
for(int i=0; i < n; i++)
        if (a[i]==x) return i;
   return -1;
}
```

```
int p_lin2(int a[],int  n, int x)
```

```
int p_lin2(int a[],int  n, int x)
{
    a[n]=x;
    int i=0;
```

```
int p_lin2(int a[],int  n, int x)
{
    a[n]=x;
    int i=0;
 while (a[i]!=x) i++;
```

```
int p_lin2(int a[],int  n, int x)
{
    a[n]=x;
    int i=0;
 while (a[i]!=x) i++;
       if (i==n) return -1;
```

```
int p_lin2(int a[],int  n, int x)
{
    a[n]=x;
    int i=0;
 while (a[i]!=x) i++;
       if (i==n) return -1;
                 else return i;
}
```

Поиск делением пополам

```
int p_dv(int a[], int  n, int x)
```

```
int p_dv(int a[], int  n, int x)
{
    int i=0, j=n-1, m;
```

```
int p_dv(int a[], int  n, int x)
{
    int i=0, j=n-1, m;
while(i<j)
```

```
int p_dv(int a[], int  n, int x)
{
    int i=0, j=n-1, m;
while(i<j)    {
   m=(i+j)/2;
```

```
int p_dv(int a[], int  n, int x)
{
    int i=0, j=n-1, m;
while(i<j)    {
   m=(i+j)/2;
      if (x > a[m]) i=m+1; else j=m;
                  }
```

```
int p_dv(int a[], int  n, int x)
{
    int i=0, j=n-1, m;
while(i<j)    {
   m=(i+j)/2;
      if (x > a[m]) i=m+1; else j=m;
                    }
    if (a[i]==x) return i;
```

```
int p_dv(int a[], int  n, int x)
{
    int i=0, j=n-1, m;
while(i<j)    {
   m=(i+j)/2;
      if (x > a[m]) i=m+1; else j=m;
                    }
    if (a[i]==x) return i;
            else return -1;
}
```

Интерполяционный поиск

```
int p_dv(int a[], int  n, int x)
```

```
int p_dv(int a[], int  n, int x)
{   int i=0, j=n-1, m;
```

```
int p_dv(int a[], int  n, int x)
{   int i=0, j=n-1, m;
  while(i<j)
```

```
int p_dv(int a[], int  n, int x)
{   int i=0, j=n-1, m;
  while(i<j)  {
 if (a[i]==a[j])     if (a[i]==x) return i;
                                 else return -1;
```

```
int p_dv(int a[], int  n, int x)
{   int i=0, j=n-1, m;
  while(i<j)  {
 if (a[i]==a[j])      if (a[i]==x) return i;
                                  else return -1;
m=i+(j-i)*(x-a[i])/(a[j]-a[i]);
```

```
int p_dv(int a[], int  n, int x)
{   int i=0, j=n-1, m;
  while(i<j)  {
 if (a[i]==a[j])     if (a[i]==x) return i;
                                  else return -1;
m=i+(j-i)*(x-a[i])/(a[j]-a[i]);
if (a[m]==x) return m;
```

```
int p_dv(int a[], int  n, int x)
{   int i=0, j=n-1, m;
  while(i<j)  {
 if (a[i]==a[j])     if (a[i]==x) return i;
                                  else return -1;
m=i+(j-i)*(x-a[i])/(a[j]-a[i]);
if (a[m]==x) return m;
          else
```

```
int p_dv(int a[], int  n, int x)
{   int i=0, j=n-1, m;
  while(i<j)  {
 if (a[i]==a[j])     if (a[i]==x) return i;
                                  else return -1;
m=i+(j-i)*(x-a[i])/(a[j]-a[i]);
if (a[m]==x) return m;
          else if (x > a[m]) i=m+1; else j=m-1;
```

```
int p_dv(int a[], int  n, int x)
{   int i=0, j=n-1, m;
  while(i<j)  {
 if (a[i]==a[j])     if (a[i]==x) return i;
                                  else return -1;
m=i+(j-i)*(x-a[i])/(a[j]-a[i]);
if (a[m]==x) return m;
          else if (x > a[m]) i=m+1; else j=m-1;
                  }
 return -1;
}
```