数组练习

皆存问题：指判断一个区间内的元素是否都存在的问题。

解题思路：

1、先将区间内的元素折算到一个大小固定的数组里

2、让数组作为计数器，记录每个元素出现的次数(`arr[tmp]++`)

3、循环遍历数组，检查是否有元素没有出现

位运算优化：

1、由于只需要判断是否存在，所以只需要一个二进制位

2、让整型变量作为数组，将出现的元素或上去(`res |= 1 << tmp`)

3、检查整型变量中的目标二进制位是否都为1(`res == 0x3f`)

数组平移：可以用于数组的插入和删除，口诀："删除从前向后遍历，插入从后向前遍历"

数组旋转：将数组里的元素循环平移若干次。方法（以左旋为例）：

1、拼接法：直接将原数组分成两半，分别拷贝进目标数组

2、逆序法：先将左边的n个逆序，再将剩下的逆序，最后整体逆序

※逆序法适用于任何局部不变的逆序，如果逆序要求中间某些部分不能变，则可以先逆序那些不能改变的部分，再逆序整体。

排序

排序的稳定性：代码确定之后，相同元素的先后次序是否会因为输入序列的不同而变化。是则不稳定，否则稳定。

选择排序

做法：对数组取极值，然后交换到数组头部保护起来。随着*i*的步进，每次缩短1的规模。

稳定性：不稳定。

优点：好想

代码：

```
void selection_sort(int a[], int n)
{
    int min, minpos;

    for(i = 0; i < n - 1; i++) //i的位置即为这一次取极值的起始位置
    {
        min = a[i];
        minpos = i; //默认第一个元素是最小的
        for(j = i + 1; j < n; j++) //遍历剩余的，记录最小值的位置
        {
```

```
            if(min > a[j])
            {
                min = a[j];
                minpos = j;
            }
        }
        swap(&a[i], &a[minpos]); //交换到头部完成这一趟排序
    }
}
```

冒泡排序

做法：不断遍历比较相邻的两个元素，若前面大于后面则交换。

稳定性：稳定。

优点：好写

代码：

```
void bubble_sort(int src[], int n)
{
    int i, j, tmp;
    for (i = 0; i < n - 1; i++)
    {
        for (j = 0; j < n - 1 - i; i++) //唯一不同的点就在这里了，直接凭记忆写
        {
            if (src[j] > src[j + 1])
            {
                tmp = src[j];
                src[j] = src[j + 1];
                src[j + 1] = tmp;
            }
        }
    }
}
```

冒泡排序的优化（了解）：若一趟冒泡排序在某个位置之后没有再执行交换，证明这个位置之后的元素已经有序。由于优化后的冒泡会丢掉其最大的优势（好写），所以不常用。

插入排序

做法：先将前i个元素设定为有序数组，然后往里插入第$i+1$个元素，重复这个过程，每趟将有序数组的规模扩大1，直到所有的元素都有序插入。

稳定性：稳定。

优点：同复杂度下最快的排序

特点：原数组越有序就越快，小规模数据下的最优解。

代码：

```
void insert_sort(int src[], int n)
{
    int i, j, tmp;
    for (i = 1; i < n; i++) //从第2个元素开始插入
    {
        tmp = src[i]; //保护待插入的值
        for (j = i; j > 0 && tmp < src[j - 1]; j--) //边平移边向前寻找待插入的位置
        {
            src[j] = src[j - 1];
        }
        m_data[j] = tmp;
    }
}
```