



中國人民大學

RENMIN UNIVERSITY OF CHINA

信息学院

SCHOOL OF INFORMATION

程序设计荣誉课程

7. 算法2——排序和查找

授课教师：游伟 副教授、孙亚辉 副教授

授课时间：周二14:00 – 15:30，周四10:00 – 11:30（教学三楼3304）

上机时间：周二18:00 – 21:00（理工配楼二层机房）

课程主页：<https://www.youwei.site/course/programming>

引子：班级通讯录

【背景】班级通讯录里记载这每个同学的详细信息。在C语言中，使用结构体数组保存班级通讯录数据。

【需求】

- 排序：根据学号排序，根据姓名排序，根据生日排序
- 查找：根据学号查找同学

目录

1. 查找
2. 排序

7.1 查找

■ 顺序查找

- 适用范围：一般针对无序表的查找
- 方法：从数组的第一个元素开始，将被查找数与数组元素进行比较，直到找到或确定不存在为止

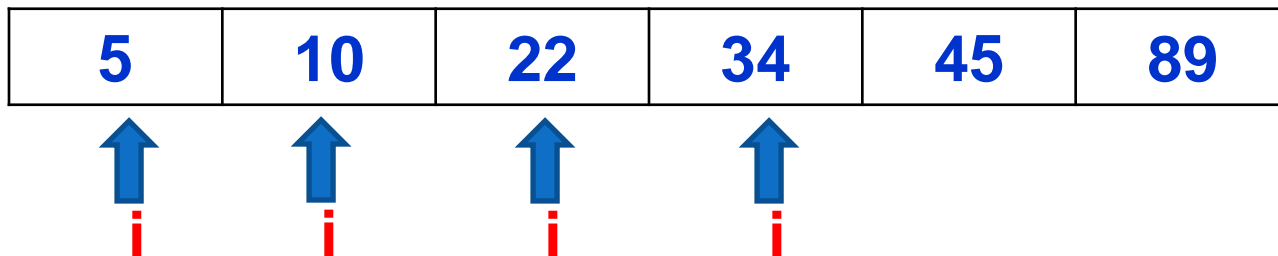
■ 折半查找

- 适用范围：有序表
- 方法：
 - 设置变量low和high分别指向查找数据段的起止位置，用mid指向中间位置，
 - 将被查找数与mid位置指向的数进行比较
 - 若被查找数大于mid位置指向的数，则将low与mid之间的数折掉，low定位于mid+1位置；
 - 若被查找数小于mid位置指向的数，则将mid与high之间的数折掉，high定位于mid-1位置；
 - 继续求mid位置，并比较被查找数与mid位置指向的数，直到找到或确定不存在为止

7.1.1 顺序查找

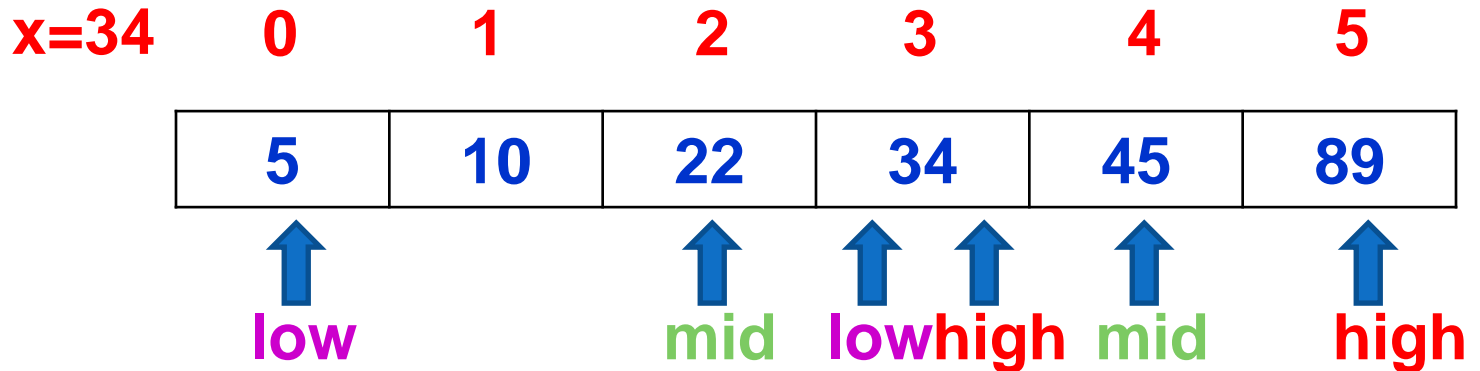
```
int main(int argc, char **argv) {
    int a[6] = {5, 10, 22, 34, 45, 89}; //a: 在数组a中进行查找
    int n = sizeof(a)/sizeof(int); //n: 在数组a的大小
    int x; //x: 待查找的目标
    int flag = 0; //flag: 标志是否找到目标
    scanf("%d", &x);
    for (int i=0; i<n; i++)
    {
        if (a[i] == x)
        {
            printf("found at index: %d\n", i);
            flag = 1;
            break;
        }
    }
    if (!flag) printf("not find\n");
}
```

x=34



思考：查找35这个数字的过程是什么样的？

7.1.2 折半查找



```
int main(int argc, char **argv) {
    int a[6] = {5, 10, 22, 34, 45, 89};
    int n = sizeof(a)/sizeof(int);
    int low = 0, high = n-1, mid;
    int x;
    int flag = 0;
    scanf("%d", &x);
    while (low <= high) {
        mid = (high + low) / 2;
        if (x > a[mid]) low = mid + 1;
        else if (x < a[mid]) high = mid - 1;
        else { printf("found at index: %d\n", mid); flag = 1; break; }
    }
    if (!flag) printf("not find\n");
}
```



搜索算法的效率

顺序搜索的平均时间性能

$$(1 + 2 + 3 + \dots + n) / n = (n + 1) / 2$$

二分查找的最坏情况的时间性能

$$n / 2 / 2 \dots / 2 / 2 = 1$$

K次

$$k = \log_2 n$$

搜索算法的效率

n	$\log_2 n$
10	3
100	7
1000	10
1 000 000	20
1 000 000 000	30

7.2 排序

- 选择排序
- 冒泡排序

7.2.1 选择排序

思路

从所有的数中找出最小的一个，将其放在最前面；接着在余下的数中找出最小的一个，将其放在第二位，依次类推，数列由前往后逐渐成型。

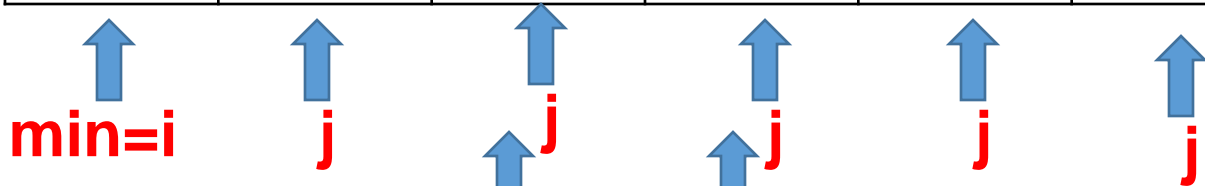
下面以6个数(22、34、10、5、89、45)为例，用图示说明。

选择法第一轮：先找出序列中最小的一个放在 $i=0$ 位置。

i	0	1	2	3	4	5
	22	34	10	5	89	45

排 $i=0$ 位置

$i=0$						
	22	34	10	5	89	45



$min=j$ $min=j$



$min \neq i?$

5	34	10	22	89	45
---	----	----	----	----	----

选择法第二轮：找出序列中次小的一个放在*i=1*位置。

i	0	1	2	3	4	5
	5	34	10	22	89	45

排*i=1*位置

		i=1				
	5	34	10	22	89	45



min=j



min != i?

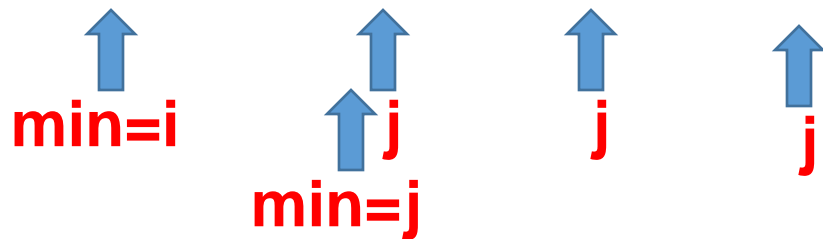
5	10	34	22	89	45
---	----	----	----	----	----

选择法第三轮：找出序列中第三小的放在 $i=2$ 位置。

i	0	1	2	3	4	5
	5	10	34	22	89	45

排 $i=2$ 位置

			i=2			
	5	10	34	22	89	45



min != i?

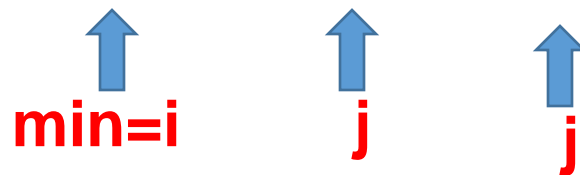
5	10	22	34	89	45
---	----	----	----	----	----

选择法第四轮：找出序列中第四小的放在*i=3*位置。

i	0	1	2	3	4	5
	5	10	22	34	89	45

排*i=3*位置

			i=3			
	5	10	22	34	89	45



34 > 89? 34 > 45?

min != i?

5	10	22	34	89	45
---	----	----	----	----	----

选择法第五轮：找出序列中第五小的放在*i=4*位置。

i	0	1	2	3	4	5
	5	10	22	34	89	45

排*i=4*位置

	5	10	22	34	i=4 89	45
--	---	----	----	----	------------------	----

min=i
min=j

89 > 45?

min != i?

	5	10	22	34	45	89
--	---	----	----	----	----	----

选择排序的核心代码

```
1. int main(void) {
2.     int n = 10;
3.     int arr[n], i, j, temp, min;
4.     printf("Please input %d numbers:\n", n);
5.     for (i = 0; i < n; i++) scanf("%d", &arr[i]);

6.     for (i = 0; i < n - 1; i++)
7.     {
8.         min = i;
9.         for(j = i + 1; j < n; j++)
10.            if(arr[min] > arr[j]) min = j;

11.         if (min != i)
12.         {
13.             temp = arr[i];
14.             arr[i] = arr[min];
15.             arr[min] = temp;
16.         }
17.     }

18.     printf("The sorted numbers:\n");
19.     for (i = 0; i < n; i++) printf("%4d", arr[i]);
20.     printf("\n");

21.     return 0;
22. }
```


选择排序的效率

对n个元素的排序来说，找出第一个元素要比较n次，找出第二个元素比较n-1次，...，找出第n个元素比较一次。因此，总的比较次数为：

$$1 + 2 + 3 + \dots + n = n(n + 1) / 2$$

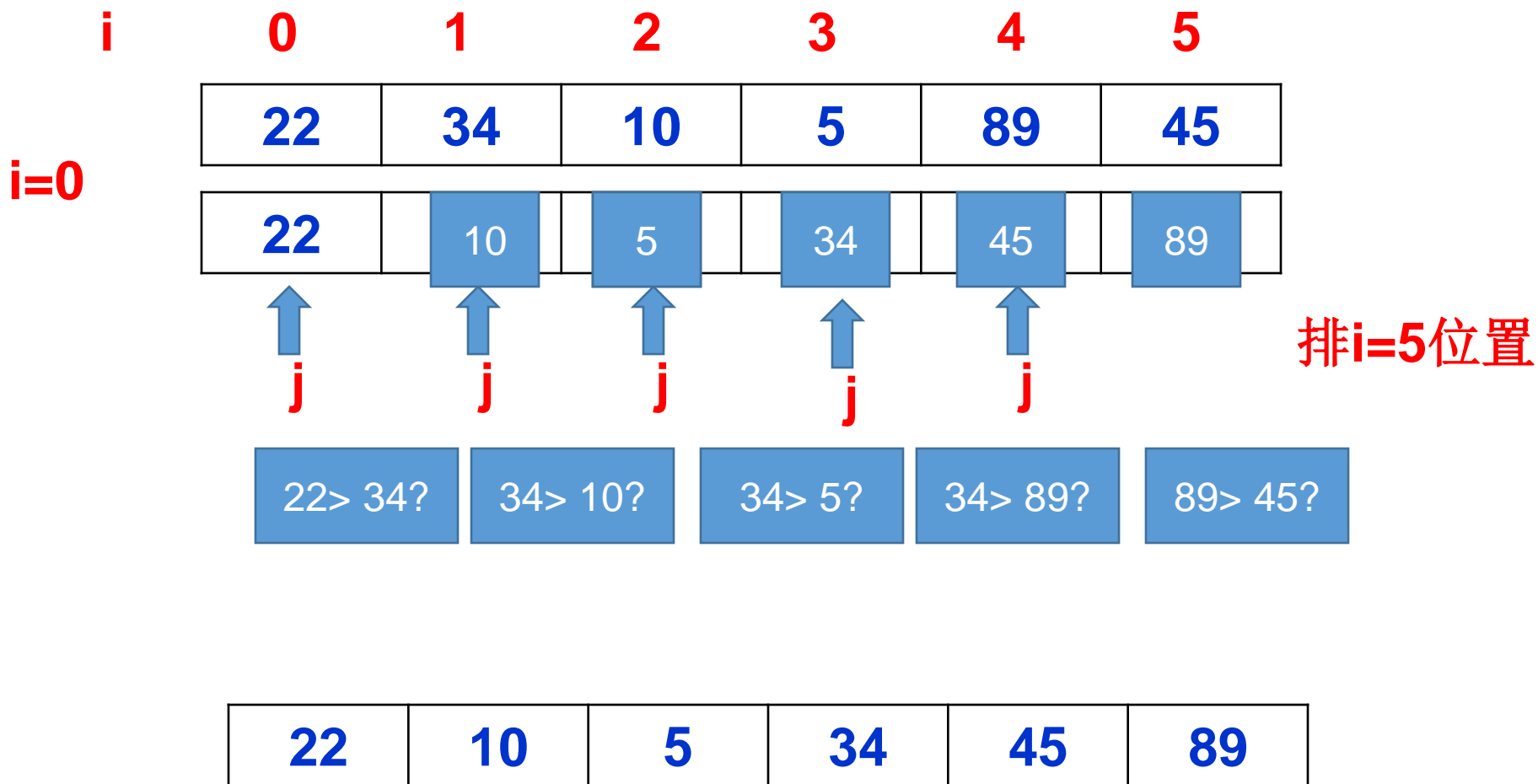
则称时间复杂性为 $O(n^2)$

7.2.2 冒泡排序

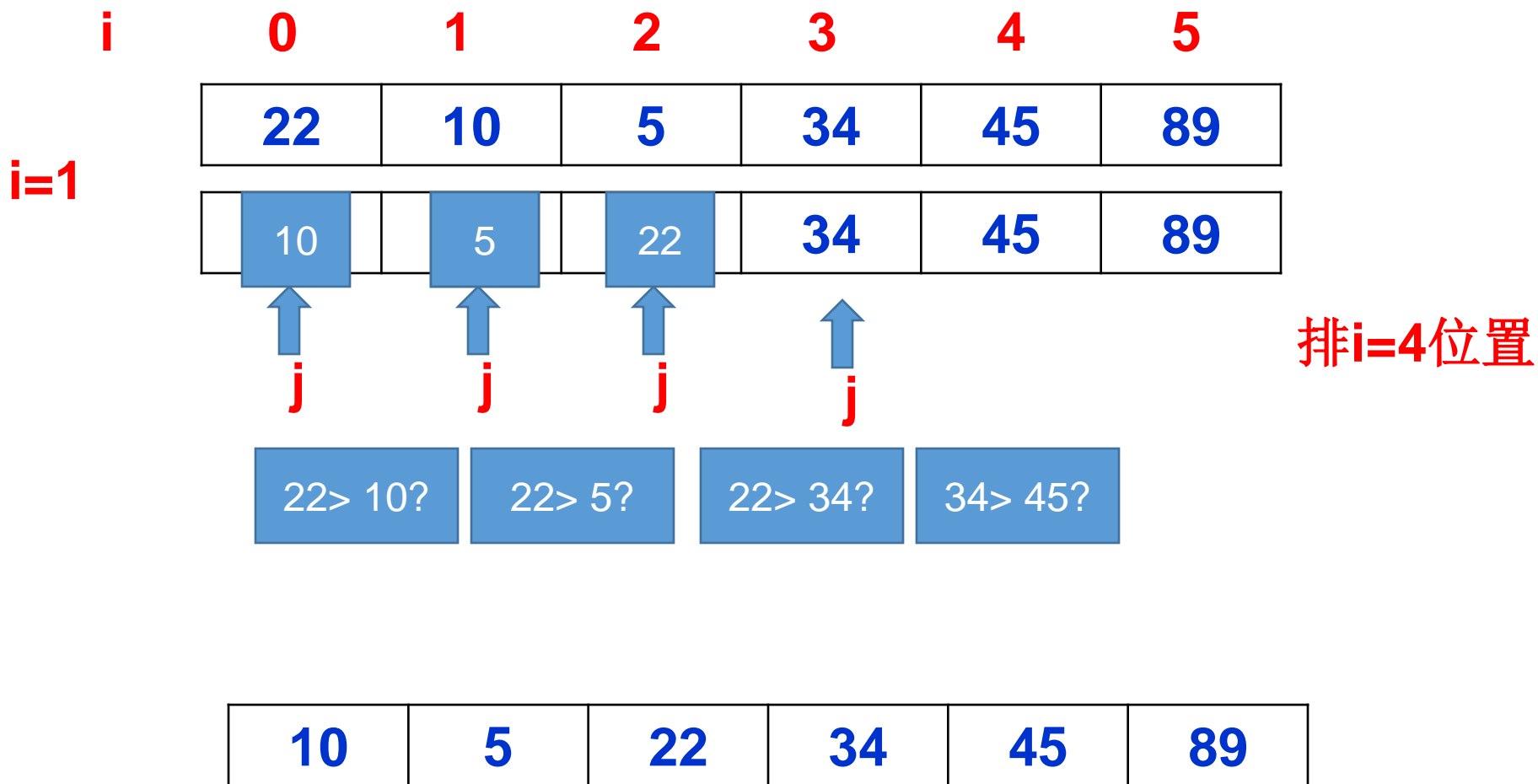
思路

对相邻两个数进行比较，将较小的调到前面，两两比较一轮之后，最大的一个数被放置在最后面；接着从头开始重复执行以上操作，次大的数被放置在倒数第二位，依次类推，数列由后往前逐渐成型。

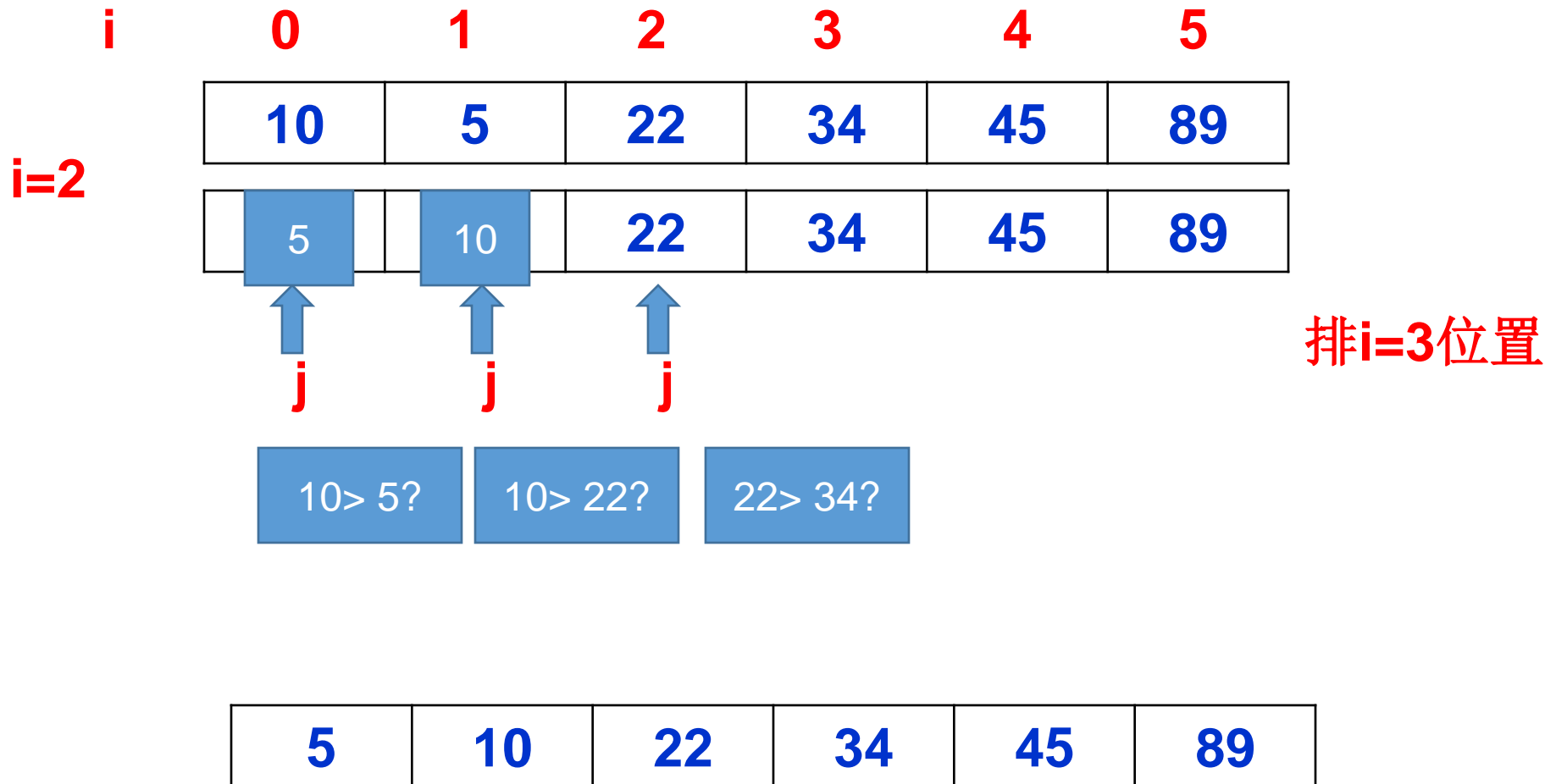
- 冒泡法的核心：小数上浮，大数下沉。
- 冒泡法第一轮：使最大的数放在最后一个位置上



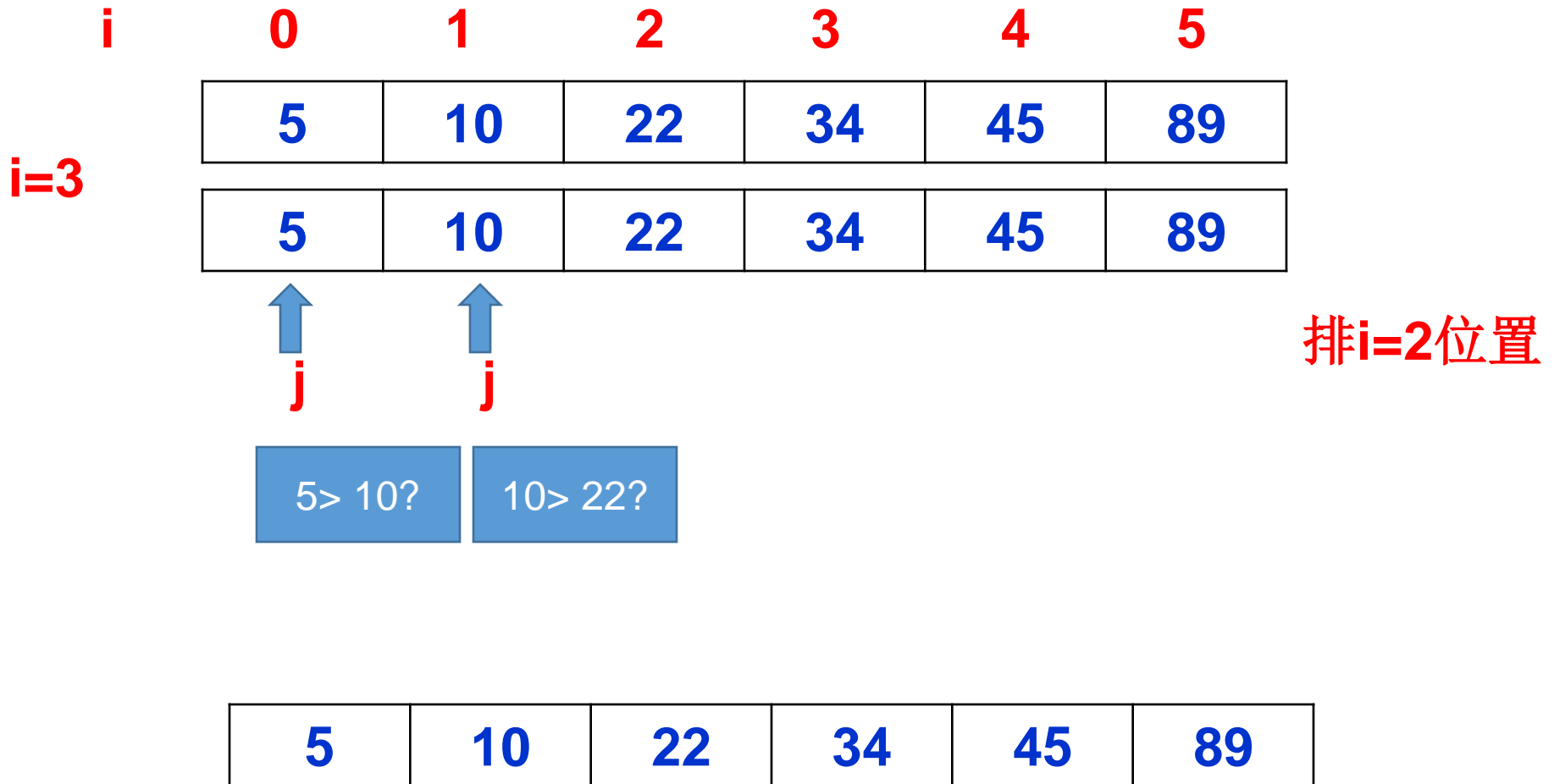
- 冒泡法的核心：小数上浮，大数下沉。
- 冒泡法第二轮：使次大的数放在倒数第二个位置上



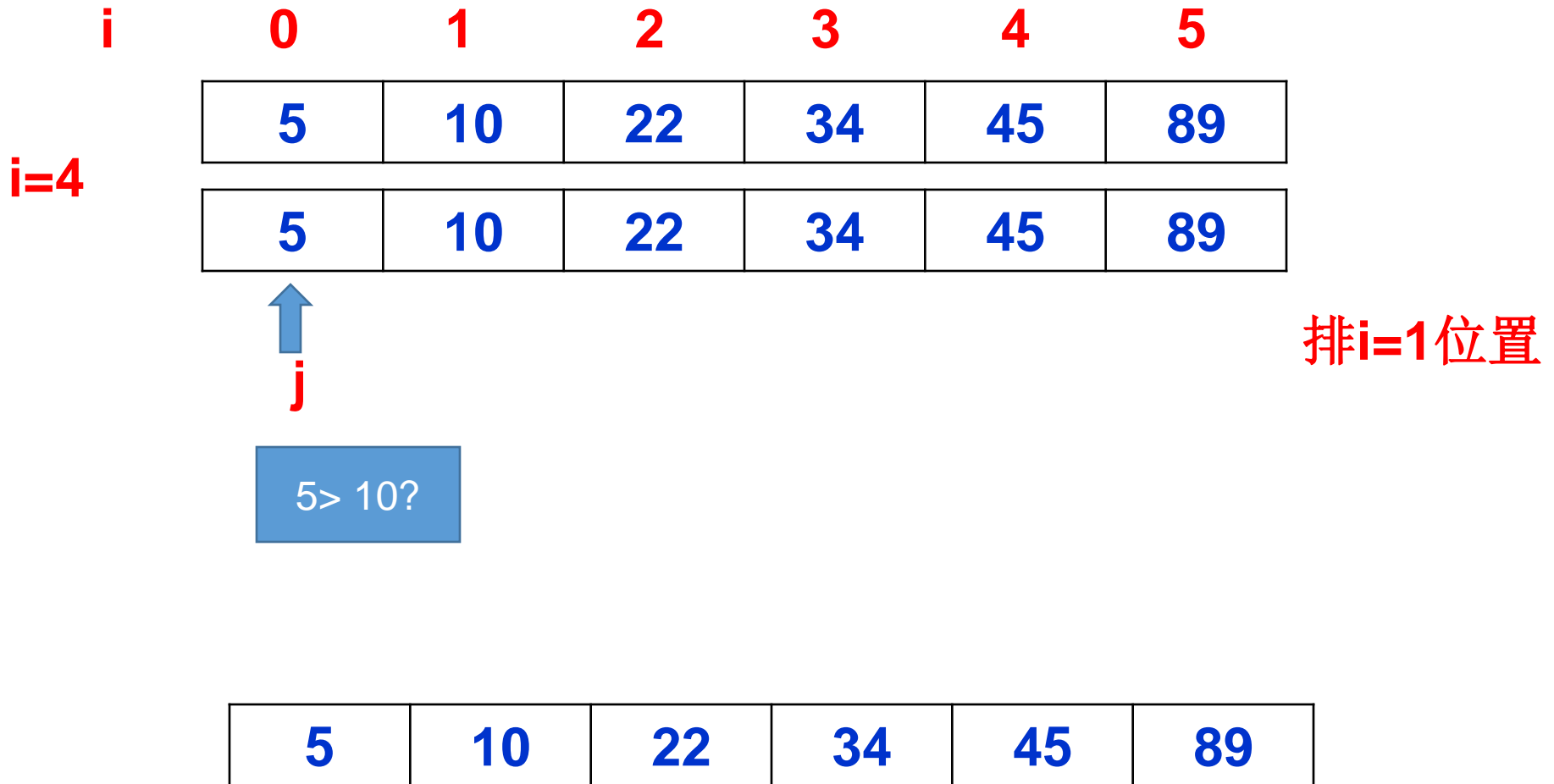
- 冒泡法的核心：小数上浮，大数下沉。
- 冒泡法第三轮：使第三大的数放在倒数第三个位置上



- 冒泡法的核心：小数上浮，大数下沉。
- 冒泡法第四轮：使第四大的数放在倒数第四个位置上



- 冒泡法的核心：小数上浮，大数下沉。
- 冒泡法第五轮：使第五大的数放在倒数第五个位置上



冒泡排序的核心代码

```
1. int main(void) {
2.     int n = 10;
3.     int arr[n], i, j, temp;

4.     printf("Please input %d numbers:\n", n);
5.     for ( i = 0; i < n; i++) scanf("%d", &arr[i]);

6.     for (i = 0; i < n - 1; i++)
7.     {
8.         for ( j = 0; j < n - i - 1; j++)
9.         {
10.            if(arr[j] > arr[j+1])
11.            {
12.                temp = arr[j];
13.                arr[j] = arr[j+1];
14.                arr[j+1] = temp;
15.            }
16.        }
17.    }

18.    printf("The sorted numbers:\n");
19.    for ( i = 0; i < n; i++) printf("%4d", arr[i]);
20.    printf("\n");

21.    return 0;
22.}
```


冒泡排序的核心代码（带优化）

```
1. int main(void) {
2.     int n = 10;
3.     int arr[n], i, j, temp, flag;

4.     printf("Please input %d numbers:\n", n);
5.     for ( i = 0; i < n; i++) scanf("%d", &arr[i]);

6.     for (i = 0; i < n - 1; i++) {
7.         flag = 0;
8.         for ( j = 0; j < n - i - 1; j++) {
9.             if(arr[j] > arr[j+1]) {
10.                flag = 1;
11.                temp = arr[j];
12.                arr[j] = arr[j+1];
13.                arr[j+1] = temp;
14.            }
15.        }
16.        if (!flag) break;
17.    }

18.    printf("The sorted numbers:\n");
19.    for ( i = 0; i < n; i++) printf("%4d", arr[i]);
20.    printf("\n");

21.    return 0;
22.}
```

冒泡排序的效率

若文件的初始状态是正序的，一趟扫描即可完成排序。所需的关键字比较次数 C 和记录移动次数 M 均达到最小值：

$$C_{\min} = n - 1, M_{\min} = 0。$$

所以，冒泡排序最好的时间复杂度为 $O(n)$ 。

若初始文件是反序的，需要进行 $n - 1$ 趟排序。每趟排序要进行 $n - i$ 次关键字的比较 ($1 \leq i \leq n - 1$)，且每次比较都必须移动记录三次来达到交换记录位置。在这种情况下，比较和移动次数均达到最大值：

$$C_{\max} = \frac{n(n-1)}{2} = O(n^2)$$

$$M_{\max} = \frac{3n(n-1)}{2} = O(n^2)$$

冒泡排序的最坏时间复杂度为 $O(n^2)$ 。

综上，因此冒泡排序总的平均时间复杂度为 $O(n^2)$ 。

排序和查找算法应用举例

■ 1. 使用选择排序将通讯录按学号信息排序

```
1. #include <stdio.h>

2. typedef enum { bai, chaoxian, han, tujia, yao } Nation;
3. typedef struct { int year; int month; int day; } Date;
4. typedef struct {
5.     int sno;
6.     char name[20];
7.     char sex;
8.     char class;
9.     Nation nation;
10.    int age;
11.    Date birthday;
12.} Student;

13.int main(void)
14.{
15.    char nations[][10] = {"bai", "chaoxian", "han", "tujia", "yao"};
16.    Student students[] =
17.    {
18.        {2021123456, "ZhangSan", 'F', 'T', han, 18, {2003, 8, 1}},
19.        {2021200834, "LiSi", 'M', 'D', yao, 19, {2002, 6, 25}},
20.    };
```

排序和查找算法应用举例

■ 1. 使用选择排序将通讯录按学号信息排序

```
21. int i, j, min;
22. Student temp;

23. for (i = 0; i < sizeof(students) / sizeof(Student) - 1; i++) {
24.     min = i;
25.     for (j = i + 1; j < sizeof(students) / sizeof(Student); j++)
26.         if (students[min].sno > students[j].sno)
27.             min = j;
28.     if (min != i) {
29.         temp = students[i];
30.         students[i] = students[min];
31.         students[min] = temp;
32.     }
33. }

34. for (i = 0; i < sizeof(students) / sizeof(Student); i++)
35.     printf("%d %s %c %c %s %d %d-%d-%d\n",
36.           students[i].sno, students[i].name,
37.           students[i].sex, students[i].class,
38.           nations[students[i].nation], students[i].age,
39.           students[i].birthday.year, students[i].birthday.month, students[i].birthday.day);
40. }
```

排序和查找算法应用举例

■ 2. 使用冒泡排序将通讯录按姓名信息排序

```
1.  int i, j;
2.  Student temp;

3.  for (i = 0; i < sizeof(students) / sizeof(Student) - 1; i++)
4.  {
5.      for (j = 0; j < sizeof(students) / sizeof(Student) - i - 1; j++)
6.          if (strcmp(students[j].name, students[j + 1].name) > 0)
7.              {
8.                  temp = students[j];
9.                  students[j] = students[j + 1];
10.                 students[j + 1] = temp;
11.             }
12. }
```

排序和查找算法应用举例

适用场景：各字段
类型相同且取值范
围固定

■ 3. 多字段排序——按生日排序

```
1.  int i, j;
2.  int is_big, encode1, encode2;
3.  Student temp;

4.  for (i = 0; i < sizeof(students) / sizeof(Student) - 1; i++)
5.  {
6.      for (j = 0; j < sizeof(students) / sizeof(Student) - i - 1; j++)
7.      {
8.          encode1 = students[j].birthday.year * 10000 + students[j].birthday.month * 100 +
9.          students[j].birthday.day;
10.         encode2 = students[j+1].birthday.year * 10000 + students[j+1].birthday.month * 100 +
11.         students[j + 1].birthday.day;

12.         if (encode1 < encode2)
13.         {
14.             temp = students[j];
15.             students[j] = students[j + 1];
16.             students[j + 1] = temp;
17.         }
18.     }
19. }
```

巧用自定义编码
进行多字段排序

排序和查找算法应用举例

■ 3. 多字段排序——按生日排序

```
1.  int i, j;
2.  int is_big, diff_year, diff_month, diff_day;
3.  Student temp;
4.  for (i = 0; i < sizeof(students) / sizeof(Student) - 1; i++)
5.  {
6.      for (j = 0; j < sizeof(students) / sizeof(Student) - i - 1; j++)
7.      {
8.          diff_year = students[j].birthday.year - students[j + 1].birthday.year;
9.          diff_month = students[j].birthday.month - students[j + 1].birthday.month;
10.         diff_day = students[j].birthday.day - students[j + 1].birthday.day;
11.
12.         is_big = (diff_year != 0 ? diff_year < 0 :
13.                (diff_month != 0 ? diff_month < 0 : diff_day < 0));
14.
15.         if (is_big)
16.         {
17.             temp = students[j];
18.             students[j] = students[j + 1];
19.             students[j + 1] = temp;
20.         }
21.     }
22. }
```

相比于自定义编码的方式更为通用

巧用逻辑表达式进行多字段排序

排序和查找算法应用举例

■ 4. 顺序查找学号

```
1.  int i, sno, found = -1;
2.
3.  scanf("%d", &sno);
4.
5.  for (i = 0; i < sizeof(students) / sizeof(Student) - 1; i++)
6.  {
7.      if (students[i].sno == sno) {
8.          found = i;
9.          break;
10.     }
11. }
12.
13. if (found >= 0) {
14.     printf("%d %s %c %c %s %d %d-%d-%d\n",
15.           students[found].sno, students[found].name,
16.           students[found].sex, students[found].class,
17.           nations[students[found].nation], students[found].age,
18.           students[found].birthday.year,
19.           students[found].birthday.month,
20.           students[found].birthday.day);
21. } else printf("not found\n");
```


排序和查找算法应用举例

■ 5. 折半查找学号

```
1.  int i, sno, found = -1;
2.  int low = 0, high = sizeof(students) / sizeof(Student) - 1;
3.  int mid;

4.  scanf("%d", &sno);

5.  while (low <= high)
6.  {
7.      mid = (low + high) / 2;
8.      if (sno > students[mid].sno)
9.          low = mid + 1;
10.     else if (sno < students[mid].sno)
11.         high = mid - 1;
12.     else
13.     {
14.         found = mid;
15.         break;
16.     }
17. }
```