# 选择排序

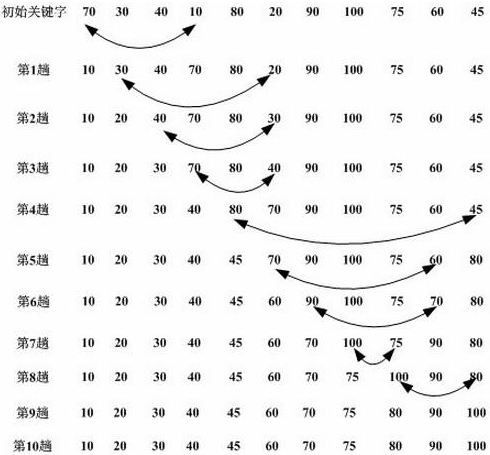
## 实验简介

选择排序中的两个经典算法：简单选择排序和堆排序。简单排序的思想是通过n-1次数据元素的比较，从n-i+1个记录中选择最小的数据，并与第i个数据进行交换，它的时间复杂度是O(n^2)。堆排序就是利用堆的特征来进行排序，它的时间复杂度是O(nlogn)。

## 一、简单选择排序

这一章我们来讲解选择排序，首先我们来讲解其中最简单的简单选择排序。

简单选择排序的基本思想是通过n-1次数据元素的比较，从n-i+1个记录中选择最小的数据，并与第i个数据进行交换，如下图所示。



见选择排序演示

简单选择排序的代码实现：

#include <stdio.h>#include <stdlib.h>

int n;

/\*

\* 选择排序

\*/void SelectSort(int \*array){

int i, j, k, temp;

for (i = 0; i < n; i++)

{

k = i;

for (j = i + 1; j < n; j++)

{

if (array[j] < array[k])

{

k = j;

}

}

if (k != i)

{

temp = array[i];

array[i] = array[k];

array[k] = temp;

}

}

}

int main(){

int i;

int \*array;

printf("请输入数组的大小：");

scanf("%d", &n);

array = (int\*) malloc(sizeof(int) \* n);

printf("请输入数据（用空格分隔）：");

for (i = 0; i < n; i++)

{

scanf("%d", &array[i]);

}

SelectSort(array);

printf("排序后为：");

for (i = 0; i < n; i++)

{

printf("%d ", array[i]);

}

printf("\n");

}

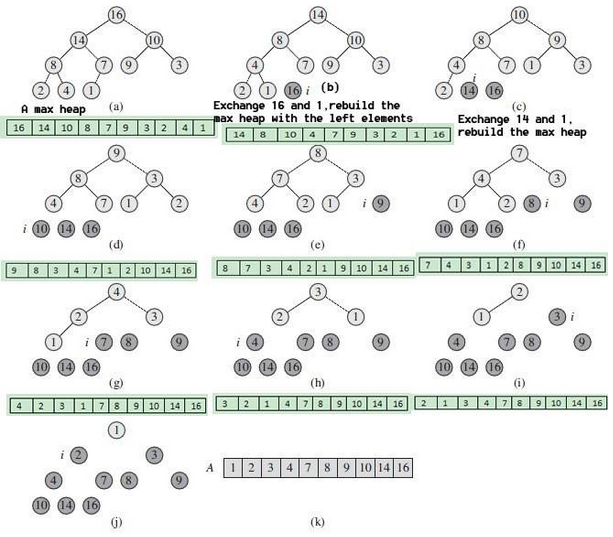
## 二、堆排序

通过前面二叉树的学习，我们知道堆是完全二叉树，有最大堆和最小堆，其中最大堆是父结点的值比子结点大，相应的最小堆就是父结点的值比子节点小。

堆排序就是利用了最大堆（或最小堆）堆顶记录的关键字最大（或最小）这一特征，使得在当前无序区中选取最大（或最小）关键字变得简单。以最大堆为例，它的基本思想就是：

1. 先将初始文件R[1..n]建成一个最大堆，此堆为初始的无序区；
2. 再将关键字最大的记录R[1]（即堆顶）和无序区的最后一个记录R[n]交换，由此得到新的无序区R[1..n-1]和有序区R[n]，且满足R[1..n-1].keys≤R[n].key；
3. 由于交换后新的根R[1]可能违反堆性质，故应将当前无序区R[1..n-1]调整为堆。然后再次将R[1..n-1]中关键字最大的记录R[1]和该区间的最后一个记录R[n-1]交换，由此得到新的无序区R[1..n-2]和有序区R[n-1..n]，且仍满足关系R[1..n-2].keys≤R[n1..n].keys，同样要将R[1..n-2]调整为堆； 重复此操作直到全部有序。

下面是示例图：



见 堆排序演示

堆排序的代码实现：

#include <stdio.h>#include <stdlib.h>

int n;

/\*

\* 生成堆

\*/void HeapAdjust(int \*array, int s, int m){

int i;

array[0] = array[s];

for (i = s \* 2; i <= m; i \*= 2)

{

if (i < m && array[i] < array[i + 1])

{

i++;

}

if (!(array[0] < array[i]))

{

break;

}

array[s] = array[i];

s = i;

}

array[s] = array[0];

}

/\*

\* 堆排序

\*/void HeapSort(int \*array){

int i;

for (i = n / 2; i > 0; i--)

{

HeapAdjust(array, i, n);

}

for (i = n; i > 1; i--)

{

array[0] = array[1];

array[1] = array[i];

array[i] = array[0];

HeapAdjust(array, 1, i - 1);

}

}

int main(){

int i;

int \*array;

printf("请输入数组的大小：");

scanf("%d", &n);

array = (int\*) malloc(sizeof(int) \* (n + 1));

printf("请输入数据（用空格分隔）：");

for (i = 1; i <= n; i++)

{

scanf("%d", &array[i]);

}

HeapSort(array);

printf("排序后为：");

for (i = 1; i <= n; i++)

{

printf("%d ", array[i]);

}

printf("\n");

}

## 三、小结

这一章讲解了选择排序中的两个经典算法，简单选择排序和堆排序，这两种都是不稳定的算法。简单排序的思想是通过n-1次数据元素的比较，从n-i+1个记录中选择最小的数据，并与第i个数据进行交换，它的时间复杂度是O(n^2)。堆排序就是利用堆的特征来进行排序，它的时间复杂度是O(nlogn)，相比于快速排序来说，它最大的优点就是在最坏情况下的时间复杂度也为O(nlogn)。

## 作业

把这章讲的两个排序算法改写成降序排列。