河南工業大學

《算法设计与分析》综合性实验
实验报告

题　　目：	快速排序
姓　　名：	田劲锋
班　　级：	软件 1305 班
学　　号：	201316920311
指导教师：	靳小波
完成时间：	2014 年 12 月 12 日

一、 实验题目

快速排序

二、 实验目的

1. 掌握分治算法
2. 实现快速排序

三、 实验要求

使用快速排序算法对一系列数进行排序。

```
8
3 41 52 26 38 57 9 49
```

四、 程序流程图

与归并排序类似，快速排序也是基于分治算法思想的。我们还是分三步来对子数组 $A[p..r]$ 进行处理：

分解 把 $A[p..r]$ 分成 $A[p..q-1]$ 和 $A[q+1..r]$ 两个子数组，使得 $A[p..q-1]$ 中的每个元素都小于等于 $A[q]$，而 $A[q+1..r]$ 中的每个元素则都大于等于 $A[q]$。此步就是计算这个 q 。

解决 递归对 $A[p..q-1]$ 和 $A[q+1..r]$ 进行排序。

合并 因为子数组已经有序，那么 $A[p..r]$ 排序完成。

下面的就是快速排序过程：

```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```

对数组 A 排序则调用 QUICKSORT$(A, 1, A.length)$ 即可。

算法的关键是将 $A[p..r]$ 重新排列的划分过程 PARTITION：

PARTITION(A, p, r)

1 $x = A[r]$

2 $i = p - 1$

3 **for** $j = p$ **to** $r - 1$

4 **if** $A[j] \leq x$

5 $i++$

6 交换 $A[i] \leftrightarrow A[j]$

7 交换 $A[i + 1] \leftrightarrow A[r]$

8 **return** i + 1

PARTITION 每次选择一个元素 $x = A[r]$ 作为比较基准，i 和 j 两个指针分别用来分割比 x 小和比 x 大的子数组，3–6 行的循环用来把不合适的元素进行交换。

快速排序的均摊时间复杂度是 $O(n \lg n)$，在某些特殊情况下会退化成 $O(n^2)$。

五、 程序代码

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

template <class Type>
int Partition(vector<Type> &A, int p, int r)
{
    Type x = A[r];
    int i = p - 1;
    for (int j = p; j <= r - 1; j++) {
        if (A[j] <= x) {
            i++;
            swap(A[i], A[j]);
        }
    }
    swap(A[i + 1], A[r]);
    return i + 1;
}

template <class Type>
void Quicksort(vector<Type> &A, int p, int r)
{
    if (p < r) {
```

```
        int q = Partition(A, p, r);
        Quicksort(A, p, q - 1);
        Quicksort(A, q + 1, r);
    }
}

int main()
{
    int n;
    cin >> n;

    vector<int> a(n + 1);

    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    Quicksort(a, 1, n);

    for (int i = 1; i <= n; i++) {
        cout << a[i] << " ";
    }
    cout << endl;

    return 0;
}
```

六、 实验结果

```
3 9 26 38 41 49 52 57
```

七、 实验体会

这次的伪代码依然来自《算法导论》[1] 并按其改写成C++程序。顾名思义，快速排序的算法在均摊意义上是基于比较的最快的排序算法，因此大多数的排序都采用快速排序。快速排序是不稳定的，通过为每个元素分配唯一的辅助键值可以达成稳定排序的目的。C `<stdlib.h>` 中提供了 `qsort` 函数，即快速排序；C++ STL `<algorithm>` 库中提供了 `sort` 函数，对于大数据采用了快速排序。

参考文献

[1] *Introduction to Algorithms*, Third Edition, Thomas H. Cormen and Charles E. Leiserson and Ronald L. Rivest and Clifford Stein, 2011