华东师范大学数据科学与工程学院上机实践报告

课程名称：算法设计与分析	**年级**：19 级	**上机实践成绩**：
指导教师：金澈清	**姓名**：龚敬洋	
上机实践名称：排序算法	**学号**：10195501436	**上机实践日期**：2020/9/18
上机实践编号：No.1	**组号**：1-436	

一、目的

1．熟悉算法设计的基本思想
2．掌握排序算法的基本思想，并且能够分析算法性能

二、内容与设计思想

1. 设计一个数据生成器，输入参数包括 N, s, t, T；可随机生成一个大小为 N、数值范围在[s, t]之间、类型为 T 的数据集合；T 包括三种类型（顺序递增、顺序递减、随机取值）
2. 编程实现 merge sort 算法和 insertion sort 算法。
3. 对于顺序递增类型的数据集合而言，在不同数据规模情况下（数据规模为 10^2, 10^3, 10^4, 10^5, 10^6）下，两种算法的运行时间各是多少？
4. 对于顺序递减类型的数据集合而言，在不同数据规模情况下（数据规模为 10^2, 10^3, 10^4, 10^5, 10^6）下，两种算法的运行时间各是多少？
5. 对于随机取值类型的数据集合而言，在不同数据规模情况下（数据规模为 10^2, 10^3, 10^4, 10^5, 10^6）下，两种算法的运行时间各是多少？
6. 补充题：编程实现 bubble sort 算法，并与上面两个算法进行对比。

三、使用环境

推荐使用 C/C++集成编译环境。

四、实验过程

1. 写出数据生成器和三种算法的源代码；

数据生成器：

```
#include <iostream>
#include <fstream>
#include <algorithm>
#include <cstdlib>
#include <ctime>
using namespace std;
bool cmp(int a, int b){
    return b < a;
}
int main() {
```

```
    int n,s,t,type,a[1000000];
    srand(time(0));
    cin>>n>>s>>t>>type;
    ofstream fout("data.txt");
    for(int i = 0; i < n; i++) {
        a[i] = s + rand() % (t - s);
    }
    if(type == 1){
        sort(a, a + n);
    }
    else if(type == 2){
        sort(a, a + n, cmp);
    }
    for(int w = 0; w < n; w++){
        fout<<a[w]<<" ";
    }
    fout.close();
    return 0;
}
```

插入排序：

```
#include<iostream>
#include<fstream>
using namespace std;
int main(){
    int a[1000000], n, key, j;
    ifstream fin("data.txt");
    n = 0;
    while(!fin.eof()){
        fin>>a[n];
        n++;
    }
    n--;
    for(int i = 1; i < n; i++){
        key = a[i];
        j = i - 1;
        while(j >= 0 && a[j] > key){
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = key;
    }
    for(int w = 0; w < n; w++){
        cout<<a[w]<<" ";
    }
    fin.close();
    return 0;
}
```

归并排序：

```
#include<iostream>
#include<fstream>
using namespace std;
int a[1000000], t[1000000], n;
void merge(int left, int right) {
    if (right - left <= 1) return;
    int mid = left + (right - left >> 1);
    merge(left, mid);
    merge(mid, right);
    int p = left, q = mid, cur = left;
    while (cur < right) {
        if (p >= mid || (q < right && a[p] > a[q]))
            t[cur++] = a[q++];
        else
            t[cur++] = a[p++];
    }
    for (int i = left; i < right; i++) a[i] = t[i];
}

int main() {
    ifstream fin("data.txt");
    n = 0;
    while (!fin.eof()) {
        fin >> a[n];
        n++;
    }
    n--;
    merge(0, n);
    for (int w = 0; w < n; w++) {
        cout << a[w] << " ";
    }
    fin.close();
    return 0;
}
```

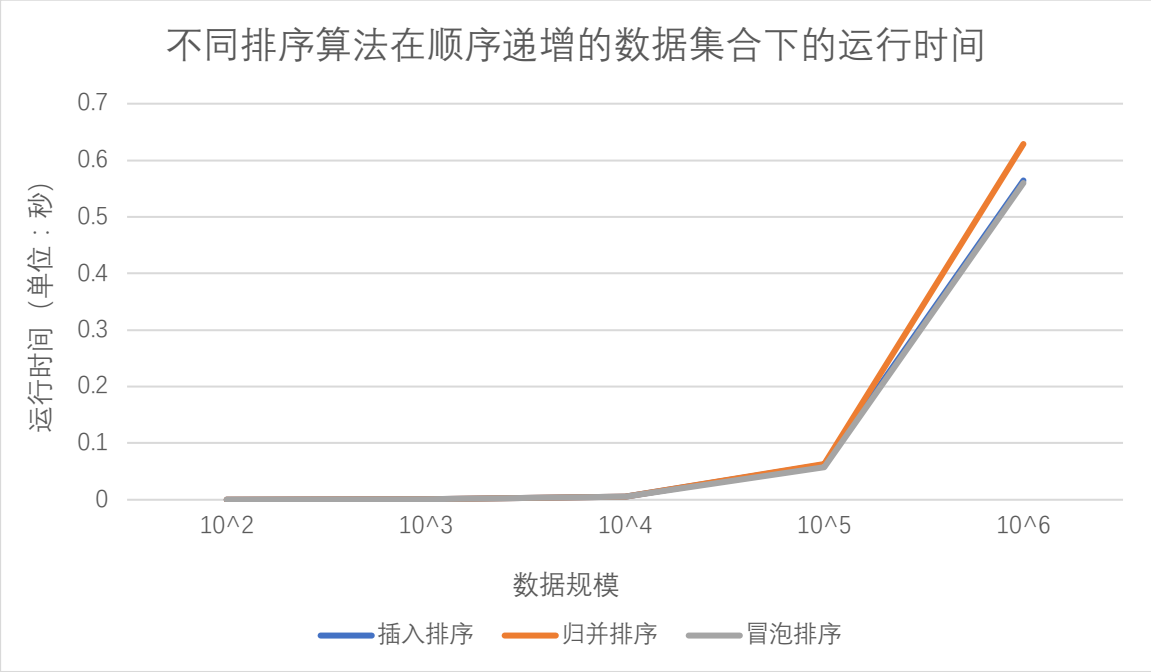
冒泡排序：

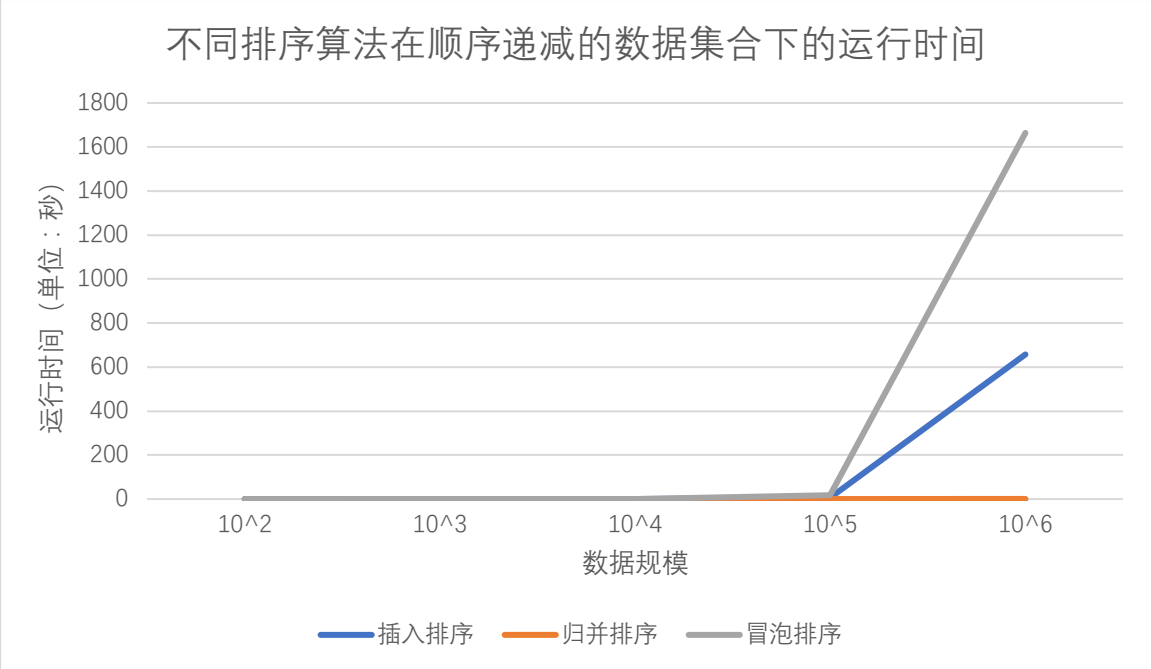
```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    int a[1000000], n, temp;
    bool flag = true;
    ifstream fin("data.txt");
    n = 0;
    while(!fin.eof()){
        fin>>a[n];
        n++;
    }
```

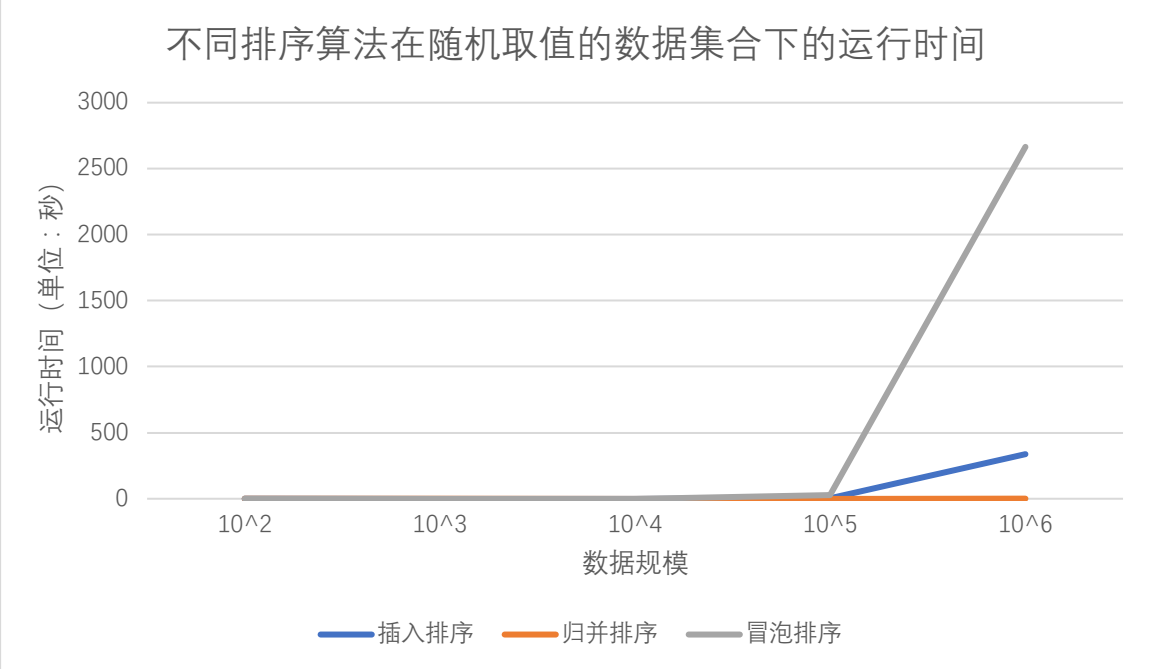
```
        n--;
        while(flag){
            flag = false;
            for(int i = 0; i < n - 1; i++){
                if(a[i] > a[i + 1]){
                    flag = true;
                    temp = a[i];
                    a[i] = a[i + 1];
                    a[i + 1] = temp;
                }
            }
        }
        for(int w = 0; w < n; w++) cout<<a[w]<<" ";
        fin.close();
        return 0;
    }
```

2. *分别画出各个实验报告的折线图*

时间记录使用了 C++自带的 clock()函数，通过在程序开头和结尾分别调用 clock()函数并将两值相减，即可得到程序运行时间。结果如下：







五、总结

对上机实践结果进行分析，问题回答，上机的心得体会及改进意见。

从上面的图表中可以发现，当数据集合为已经排序好（顺序递增）的集合时，插入排序和冒泡排序的运行效率高于归并排序。但当数据集合为顺序递减或随机取值时，归并排序的运行效率要明显高于插入排序和冒泡排序。且随着数据规模的增大，归并排序所需的运行时间增长较为缓慢，而插入排序和冒泡排序的运行时间迅速增长，且冒泡排序的增长幅度要高于插入排序。通过理论计算可以得知插入排序的最好时间复杂度为$O(n)$，平均和最坏时间复杂度均为$O(n^2)$；归并排序的最好，平均，最坏时间复杂度均为$O(nlogn)$；冒泡排序的最好时间复杂度为$O(n)$，平均和最坏时间复杂度均为$O(n^2)$，这与实验结果基本吻合。