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：**22 级　　**上机实践成绩：**

指导教师：金澈清　　**姓名：**郭夏辉

上机实践名称：排序算法　　**学号：**10211900416　　**上机实践日期：**2023 年 3 月 2 日

上机实践编号：No.1　　**组号：**416

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

VScode

四、实验过程

1. *写出数据生成器和三种算法的源代码；*
2. *分别画出各个实验报告的折线图*

1.数据生成器

基于题目的要求，我先写出了简单的数据生成器，代码如下所示.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
//用于生成随机数据
//生成的数据数量是 N,范围在[s,t],T 是生成数据的类型
//T 1 顺序递增 2 顺序递减 3 随机取值
//这里的递增和递减指的是严格单调递增和严格严格单调递减
int super_rand(int s,int t);
```

```
void f1(FILE *fp,int N,int s,int t){
    int tmp=super_rand(s,t);
    fprintf(fp,"%u ",tmp);
    int tmp1;
    for(int i=2;i<=N;i++){
        tmp1=super_rand(s,t);
        while(tmp1<tmp){
            tmp1=super_rand(s,t);
        }
        tmp=tmp1;
        fprintf(fp,"%u ",tmp);
    }
}
void f2(FILE *fp,int N,int s,int t){
    int tmp=super_rand(s,t);
    fprintf(fp,"%d ",tmp);
    int tmp1;
    for(int i=2;i<=N;i++){
        tmp1=super_rand(s,t);
        while(tmp1>tmp){
            tmp1=super_rand(s,t);
        }
        tmp=tmp1;
        fprintf(fp,"%d ",tmp);
    }
}
void f3(FILE *fp,int N,int s,int t){
    int tmp;
    for(int i=1;i<=N;i++){
        tmp=super_rand(s,t);
        fprintf(fp,"%d ",tmp);
    }
}

int super_rand(int s,int t){
    //为什么我要写一个 super_rand?因为我的库函数中 RAND_MAX 仅仅为 32767
    //rand()生成的数据在[0,RAND_MAX] 这么小的数据，显然是无法很好地获得随机数据
    //我通过线性映射，近似地将[0,RAND_MAX]映射到了我们需要的[s,t]
    long long randlen=RAND_MAX+1;
    long long mylen=(long long)t-(long long)s+1; //由于 t-s+1 可能超过 int 的范围，我用
long long 类型来存

    double tmp1=(double)(rand())/randlen;
    long long tmp2=mylen*tmp1;
    int ans=(int)(tmp2+(long long)s);
    return ans;
}
```

```
/*
int main(){
    int N,s,t,T;
    scanf("%d%d%d%d",&N,&s,&t,&T);
    srand(time(NULL));
    FILE *fp1;
    fp1=fopen("get-random-num-result.txt","w");
    if(T==1)f1(fp1,N,s,t);
    else if(T==2)f2(fp1,N,s,t);
    else f3(fp1,N,s,t);

    if(fclose(fp1)){
        printf("cannot close the file!\n");
        exit(0);
    }

    return 0;
}
*/
```

然而，由于这个代码中的 super_rand()函数频繁地进行比较，在实际运行中消耗时间很大。为了较为简单地获得所需数据，我想到了一个直接的思路:通过直接调用 rand()得到了一批数据，然后利用 C++的<algorithm>库中自带的 sort()函数正着排一下反着排一下得到三种所需的数据。

```
#include <iostream>
#include <fstream>
#include <algorithm>
#include <cstdlib>
#include <ctime>
using namespace std;
bool cmp(int a, int b){return b < a;}
int n,s,t,type,a[1000000+666];
int main() {
    srand(time(0));
    cin>>n>>s>>t>>type;
    ofstream fout("randnum.txt");
    for(int i = 0; i < n; i++)a[i] = s + rand() % (t-s+1);
    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;
}
```

这个程序是简单的，但是题中要求的数据集数量还是不少的，我们一个一个手动生成显得过于笨拙，经过合适的包装，我写出了一个生成题目要求的数据(10^2-10^6 且各规模都是三种)的小程序。

```cpp
#include <iostream>
#include <fstream>
#include <algorithm>
#include <cstdlib>
#include <ctime>
#include <string.h>
using namespace std;
//没办法，这个是用来快速得到需要的数据集而生的
bool cmp(int a, int b){return b < a;}
int s,t,a[1000000+66];

char filename[17][128];
char tmp1[]="./dataset/randnum-",tmp2[]=".txt",tmp3[10]="100";
char tmp4[]="-increase",tmp5[]="-decrease",tmp6[]="-rand";

void Solve_Filename(){
    for(int i=1;i<=15;i++){
        strcpy(filename[i],tmp1);
    }
    for(int i=2;i<=6;i++){
        strcat(filename[1+(i-2)*3],tmp3);
        strcat(filename[2+(i-2)*3],tmp3);
        strcat(filename[3+(i-2)*3],tmp3);

        strcat(filename[1+(i-2)*3],tmp4);
        strcat(filename[2+(i-2)*3],tmp5);
        strcat(filename[3+(i-2)*3],tmp6);

        strcat(filename[1+(i-2)*3],tmp2);
        strcat(filename[2+(i-2)*3],tmp2);
        strcat(filename[3+(i-2)*3],tmp2);
        tmp3[i+1]='0';
        tmp3[i+2]='\0';
    }
}

int main() {
    s=0,t=0x7FFFFF00;//防止溢出
    srand(time(0));
    Solve_Filename();
    int tmp=100;
    for(int i=2;i<=6;i++){

        ofstream fout1(filename[1+(i-2)*3]);
        ofstream fout2(filename[2+(i-2)*3]);
        ofstream fout3(filename[3+(i-2)*3]);

        for(int i=0;i<tmp;i++)a[i]=s+rand()%(t-s+1);
```

```
        for(int i=0;i<tmp;i++)fout3<<a[i]<<" ";
        sort(a,a+tmp);
        for(int i=0;i<tmp;i++)fout1<<a[i]<<" ";
        sort(a,a+tmp,cmp);
        for(int i=0;i<tmp;i++)fout2<<a[i]<<" ";
        fout1.close();
        fout2.close();
        fout3.close();

        tmp*=10;
    }


    return 0;
}
```

2. 三种排序的源代码

归并排序 mergingsort.cpp

```
#include <cstdio>
#include <ctime>
using namespace std;
#define MAXN 10000000
//归并排序，第一次上机作业
int n;
int a[MAXN],b[MAXN];
void merge(int arr[],int left,int mid,int right,int tmp[]){
    int i=left;
    int j=mid+1;
    int t = 0;
    while (i<=mid && j<=right){
        if (arr[i]<=arr[j]){
            tmp[t++]=arr[i++];
        }else{
            tmp[t++]=arr[j++];
        }
    }
    while(i<=mid)tmp[t++]=arr[i++];
    while(j<=right)tmp[t++]=arr[j++];
    t=0;
    while(left<=right)arr[left++]=tmp[t++];

}
void mergesort(int arr[],int left,int right,int tmp[]){
    if (left<right){
        int mid=(left+right)/2;
        mergesort(arr,left,mid,tmp);
        mergesort(arr,mid+1,right,tmp);
        merge(arr,left,mid,right,tmp);
```

```
    }

}

int main(){
    scanf("%d",&n);
    for(int i=0;i<n;i++)scanf("%d",&a[i]);
    mergesort(a,0,n-1,b);
    for(int i=0;i<n;i++)printf("%d ",a[i]);
    return 0;
}
```

插入排序 insertion-sort.cpp

```
#include<cstdio>
using namespace std;
#define MAXN 10000000
//插入排序，第一次上机作业
int n;
int a[MAXN];
void insertionsort(int arr[],int len){
    int tmp,i;
    for(int j=1;j<len;j++){
        tmp=arr[j];
        i=j-1;
        while(i>0 && arr[i]>tmp){
            arr[i+1]=arr[i];
            i--;
        }
        arr[i+1]=tmp;
    }
}

int main(){
    scanf("%d",&n);
    for(int i=0;i<n;i++)scanf("%d",&a[i]);
    insertionsort(a,n);
    for (int i=0;i<n;i++)printf("%d ",a[i]);
    return 0;
}
```

冒泡排序 bubble-sort.cpp

```
#include <stdio.h>
using namespace std;
#define MAXN 10000000
//冒泡排序，第一次上机作业

void bubblesort(int a[],int n){
    int tmp;
    for (int i=0;i<n-1;i++){
        for (int j=0;j+i+1<n;j++) {
```

```
            if (a[j]>a[j+1]){
                tmp=a[j];
                a[j]=a[j+1];
                a[j+1]=tmp;
            }
        }
    }
    return;
}
int n;
int a[MAXN];
int main(){
    scanf("%d",&n);
    for(int i=0;i<n;i++)scanf("%d",&a[i]);
    bubblesort(a,n);
    for(int i=0;i<n;i++)printf("%d ",a[i]);
    return 0;
}
```

然后为了后续的操作，我将三种排序的函数进行了封装，放到了一个文件中

three-sort.cpp

```
//写好接口，方便调用
#include <stdio.h>
using namespace std;
void bubblesort(int a[],int n){
    int tmp;
    for (int i=0;i<n-1;i++){
        for (int j=0;j+i+1<n;j++) {
            if (a[j]>a[j+1]){
                tmp=a[j];
                a[j]=a[j+1];
                a[j+1]=tmp;
            }
        }
    }
    return;
}
void merge(int arr[],int left,int mid,int right,int tmp[]){
    int i=left;
    int j=mid+1;
    int t = 0;
    while (i<=mid && j<=right){
        if (arr[i]<=arr[j]){
            tmp[t++]=arr[i++];
        }else{
            tmp[t++]=arr[j++];
        }
    }
    while(i<=mid)tmp[t++]=arr[i++];
```

```
    while(j<=right)tmp[t++]=arr[j++];
    t=0;
    while(left<=right)arr[left++]=tmp[t++];

}
void mergesort(int arr[],int left,int right,int tmp[]){
    if (left<right){
        int mid=(left+right)/2;
        mergesort(arr,left,mid,tmp);
        mergesort(arr,mid+1,right,tmp);
        merge(arr,left,mid,right,tmp);
    }

}
void insertionsort(int arr[],int len){
    int tmp,i;
    for(int j=1;j<len;j++){
        tmp=arr[j];
        i=j-1;
        while(i>0 && arr[i]>tmp){
            arr[i+1]=arr[i];
            i--;
        }
        arr[i+1]=tmp;
    }
}
```

3.批量获得各数据在三种排序算法下的运行时间

还是为了简便操作，我写了一个批量测试的程序，这个程序会将各次的结果记录在文件中。

测试程序 test-getans.cpp

```
//要和 three-sort.cpp 一起编译
// g++ .\test-getans.cpp .\three-sort.cpp -o testcases.exe
#include<iostream>
#include<fstream>
#include<cstdio>
#include<stdio.h>
#include<ctime>
#include<string.h>
using namespace std;
int a[1000000+66],b[1000000+66],n;
char filename[17][128];
char tmp1[]="./dataset/randnum-",tmp2[]=".txt",tmp3[10]="100";
char tmp4[]="-increase",tmp5[]="-decrease",tmp6[]="-rand";

extern void bubblesort(int a[],int n);
extern void mergesort(int arr[],int left,int right,int tmp[]);
extern void insertionsort(int arr[],int len);
```

```
void Solve_Filename(){
    for(int i=1;i<=15;i++){
        strcpy(filename[i],tmp1);
    }
    for(int i=2;i<=6;i++){
        strcat(filename[1+(i-2)*3],tmp3);
        strcat(filename[2+(i-2)*3],tmp3);
        strcat(filename[3+(i-2)*3],tmp3);

        strcat(filename[1+(i-2)*3],tmp4);
        strcat(filename[2+(i-2)*3],tmp5);
        strcat(filename[3+(i-2)*3],tmp6);

        strcat(filename[1+(i-2)*3],tmp2);
        strcat(filename[2+(i-2)*3],tmp2);
        strcat(filename[3+(i-2)*3],tmp2);
        tmp3[i+1]='0';
        tmp3[i+2]='\0';
    }
}

int main(){
    ofstream fout("data.txt");
    clock_t startTime, endTime;
    Solve_Filename();
    int scale=100;
    for(int i=2;i<=6;i++){

        ifstream fin1(filename[1+(i-2)*3]);
        ifstream fin2(filename[2+(i-2)*3]);
        ifstream fin3(filename[3+(i-2)*3]);

        for(n=0;n<scale;n++)fin1>>a[n];
        startTime=clock();
        mergesort(a,0,n-1,b);
        endTime = clock();

        fout<<"Mergesort"<<n<<"Increase"<<"RunTime: "<<(1000*( (double) (endTime -
startTime) / CLOCKS_PER_SEC))<<"ms"<<endl;
        fin1.close();

        for(n=0;n<scale;n++)fin2>>a[n];
        startTime=clock();
        mergesort(a,0,n-1,b);
        endTime = clock();

        fout<<"Mergesort"<<n<<"Decrease"<<"RunTime: "<<(1000*( (double) (endTime -
startTime) / CLOCKS_PER_SEC))<<"ms"<<endl;
```

```
        fin2.close();

        for(n=0;n<scale;n++)fin3>>a[n];
        startTime=clock();
        mergesort(a,0,n-1,b);
        endTime = clock();

        fout<<"Mergesort"<<n<<"Rand"<<"RunTime: "<<(1000*( (double) (endTime -
startTime) / CLOCKS_PER_SEC))<<"ms"<<endl;
        fin3.close();

        scale*=10;
    }

    scale=100;
    for(int i=2;i<=6;i++){

        ifstream fin1(filename[1+(i-2)*3]);
        ifstream fin2(filename[2+(i-2)*3]);
        ifstream fin3(filename[3+(i-2)*3]);

        for(n=0;n<scale;n++)fin1>>a[n];

        startTime=clock();
        insertionsort(a,n);
        endTime = clock();

        fout<<"Insertsort"<<n<<"Increase"<<"RunTime: "<<(1000*( (double) (endTime -
startTime) / CLOCKS_PER_SEC))<<"ms"<<endl;
        fin1.close();

        for(n=0;n<scale;n++)fin2>>a[n];
        startTime=clock();
        insertionsort(a,n);
        endTime = clock();

        fout<<"Insertsort"<<n<<"Decrease"<<"RunTime: "<<(1000*( (double) (endTime -
startTime) / CLOCKS_PER_SEC))<<"ms"<<endl;
        fin2.close();


        for(n=0;n<scale;n++)fin3>>a[n];
        startTime=clock();
        insertionsort(a,n);
        endTime = clock();

        fout<<"Insertsort"<<n<<"Rand"<<"RunTime: "<<(1000*( (double) (endTime -
startTime) / CLOCKS_PER_SEC))<<"ms"<<endl;
        fin3.close();
```

```
        scale*=10;
    }



    scale=100;
    for(int i=2;i<=6;i++){

        ifstream fin1(filename[1+(i-2)*3]);
        ifstream fin2(filename[2+(i-2)*3]);
        ifstream fin3(filename[3+(i-2)*3]);

        for(n=0;n<scale;n++)fin1>>a[n];
        startTime=clock();
        bubblesort(a,n);
        endTime = clock();

        fout<<"Bubblesort"<<scale<<"Increase"<<"RunTime: "<<(1000*( (double)
(endTime - startTime) / CLOCKS_PER_SEC))<<"ms"<<endl;
        fin1.close();

        for(n=0;n<scale;n++)fin2>>a[n];
        startTime=clock();
        bubblesort(a,n);
        endTime = clock();

        fout<<"Bubblesort"<<scale<<"Decrease"<<"RunTime: "<<(1000*( (double)
(endTime - startTime) / CLOCKS_PER_SEC))<<"ms"<<endl;
        fin2.close();

        for(n=0;n<scale;n++)fin3>>a[n];
        startTime=clock();
        bubblesort(a,n);
        endTime = clock();

        fout<<"Bubblesort"<<scale<<"Rand"<<"RunTime: "<<(1000*( (double) (endTime -
startTime) / CLOCKS_PER_SEC))<<"ms"<<endl;
        fin3.close();

        scale*=10;
    }
    fout.close();
    return 0;
}
```

由于这个程序调用了 three-sort.cpp 中的函数，故应该链接起来

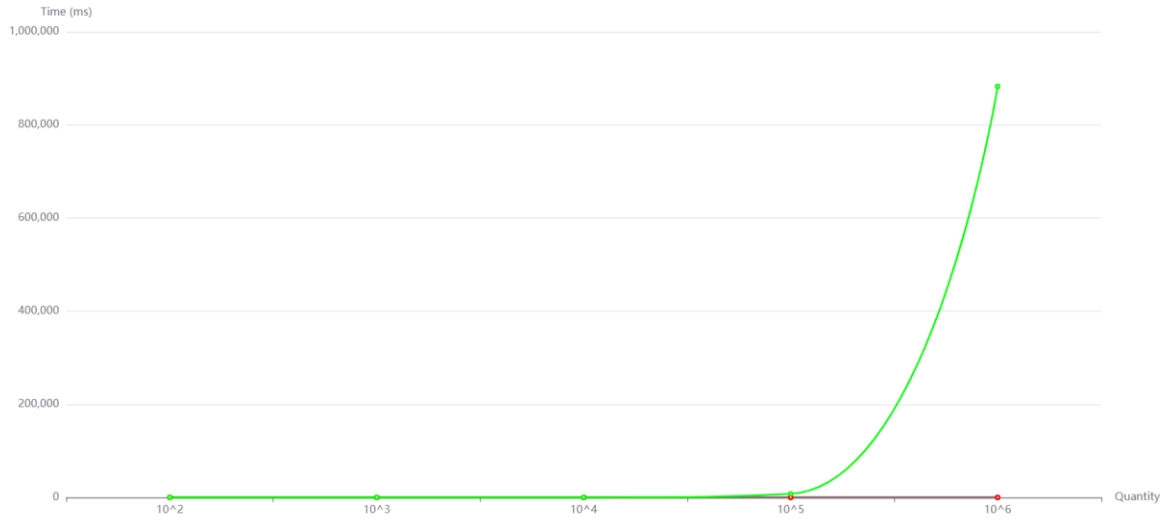
```
g++ .\test-getans.cpp .\three-sort.cpp -o testcases.exe
```

在运行了 testcases.exe 之后(要等一段时间)，在该目录下的 data.txt 得到测试结果

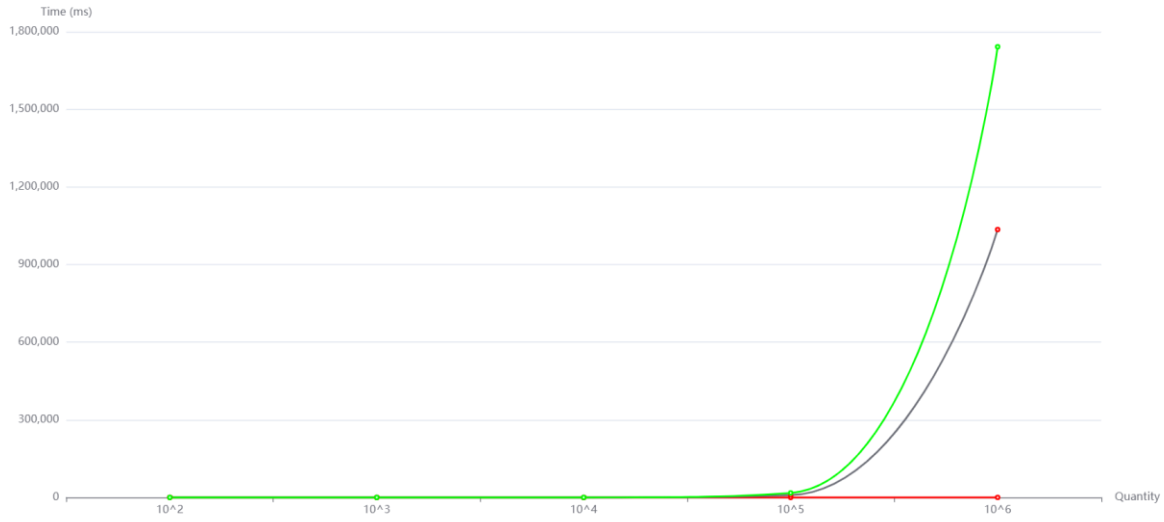
Mergesort100IncreaseRunTime: 0ms
Mergesort100DecreaseRunTime: 0ms
Mergesort100RandRunTime: 0ms
Mergesort1000IncreaseRunTime: 0ms
Mergesort1000DecreaseRunTime: 0ms
Mergesort1000RandRunTime: 0ms
Mergesort10000IncreaseRunTime: 1ms
Mergesort10000DecreaseRunTime: 1ms
Mergesort10000RandRunTime: 1ms
Mergesort100000IncreaseRunTime: 6ms
Mergesort100000DecreaseRunTime: 6ms
Mergesort100000RandRunTime: 13ms
Mergesort1000000IncreaseRunTime: 67ms
Mergesort1000000DecreaseRunTime: 79ms
Mergesort1000000RandRunTime: 137ms
Insertsort100IncreaseRunTime: 0ms
Insertsort100DecreaseRunTime: 0ms
Insertsort100RandRunTime: 0ms
Insertsort1000IncreaseRunTime: 0ms
Insertsort1000DecreaseRunTime: 1ms
Insertsort1000RandRunTime: 0ms
Insertsort10000IncreaseRunTime: 0ms
Insertsort10000DecreaseRunTime: 88ms
Insertsort10000RandRunTime: 44ms
Insertsort100000IncreaseRunTime: 0ms
Insertsort100000DecreaseRunTime: 8931ms
Insertsort100000RandRunTime: 4449ms
Insertsort1000000IncreaseRunTime: 2ms
Insertsort1000000DecreaseRunTime: 1.03531e+006ms
Insertsort1000000RandRunTime: 503802ms
Bubblesort100IncreaseRunTime: 0ms
Bubblesort100DecreaseRunTime: 0ms
Bubblesort100RandRunTime: 0ms
Bubblesort1000IncreaseRunTime: 1ms
Bubblesort1000DecreaseRunTime: 2ms
Bubblesort1000RandRunTime: 2ms
Bubblesort10000IncreaseRunTime: 71ms
Bubblesort10000DecreaseRunTime: 173ms
Bubblesort10000RandRunTime: 151ms
Bubblesort100000IncreaseRunTime: 7783ms
Bubblesort100000DecreaseRunTime: 17293ms
Bubblesort100000RandRunTime: 21591ms
Bubblesort1000000IncreaseRunTime: 882446ms
Bubblesort1000000DecreaseRunTime: 1.74162e+006ms
Bubblesort1000000RandRunTime: 2.21848e+006ms

然后根据得到的数据我用 ecahrt 做出了三种排序在不同数据集下的运行时间图(该图进行了平滑展示)

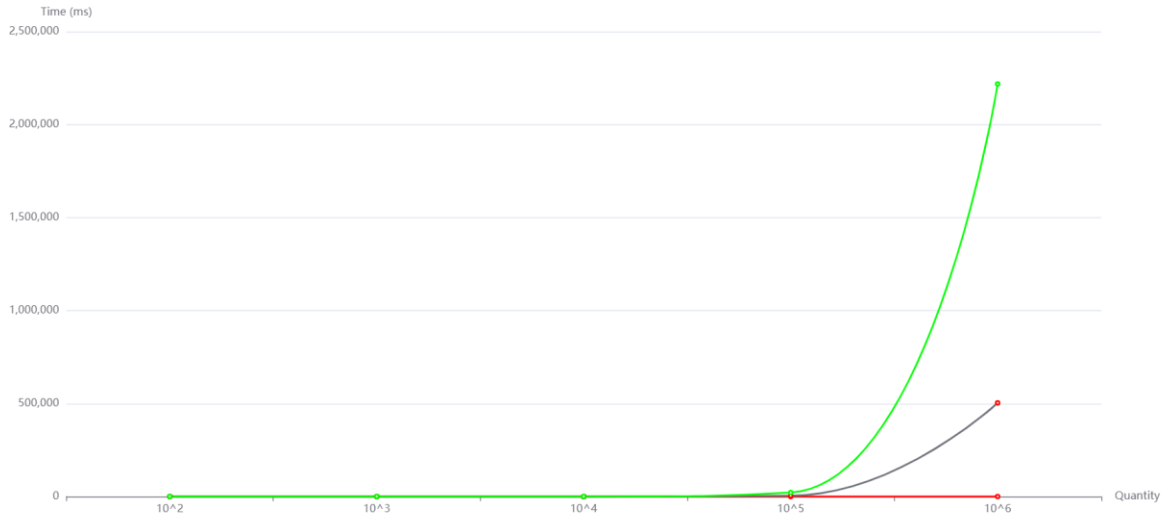
数据是递增情况:



数据是递减情况



数据是随机情况



(三个图中:红线是 Mergesort 蓝线是 InsertSort 绿线是 BubbleSort)

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。

算法是用来解决特定问题的优化方法。在这个实验中，通过基础的排序算法，我见识了面对庞大数据时算法到底有多高效，对其中明显的时间差异印象深刻。

同时，在完成这个实验的过程中，我采用了较为灵活的方法，就是实现自动化的程序生成随机数据并进行测试，这个过程中遇到了各种问题，比如自己修改 rand()导致效率过低。通过不断地改进，我终于完成了易于实现的自动化程序，运用了链接计数使代码更精简，但是在测试时没有很好地封装，希望未来再次有所突破。