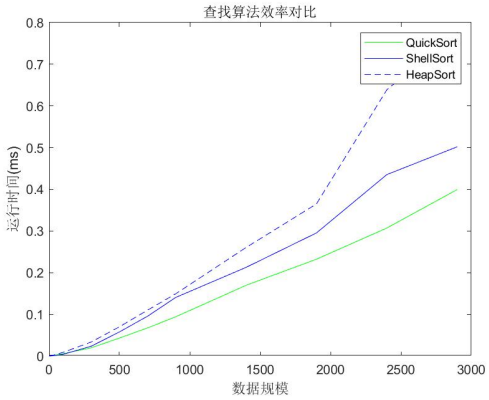
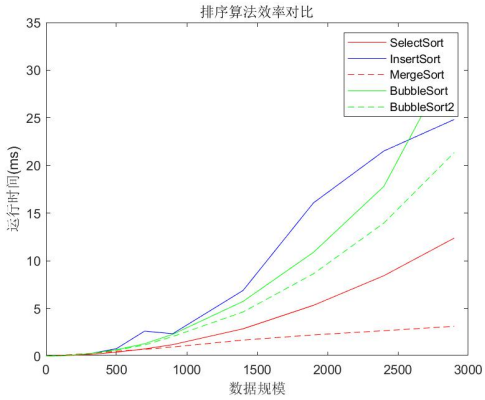
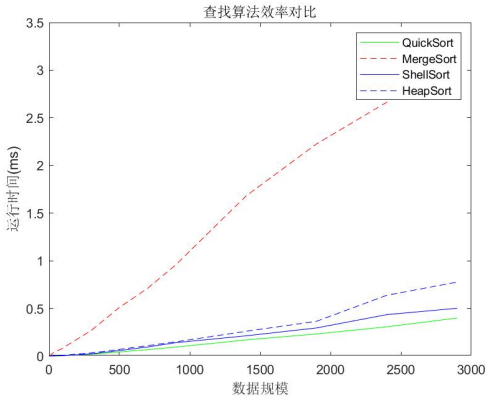
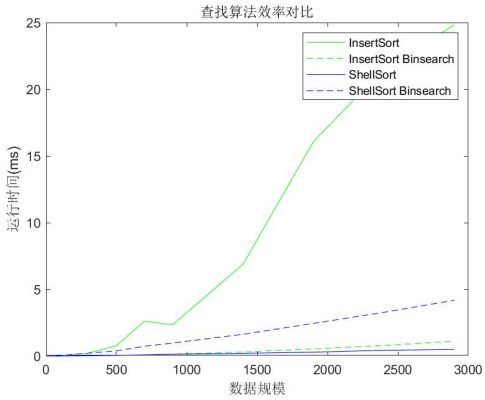
第一题及第三题：

分析比较排序算法效率，改进直接插入排序和 shell 排序。









注意：纵坐标单位长度不一样

第二题：

成功读取文件信息之后，若输入排序字段：学号、姓名、性别、院系、年级专业或班级，则排序；若输入：退出，则退出程序。

根据数据特点，使用结构体 student 储存每个学生的信息，用类封装实现“打印名单”等功能。考虑到数据已经基本有序，且数据量不大，采用直接插入排序（快排也写了，改一下可以直接用）。

排序时 struct student 之间的大小关系由排序字段决

定，所以应在排序函数的参数中加上一个比较函数。此时排序函数声明为：

```
template<class T>
  void qsort(T s[], int l, int r, CompareFunction<T> cmp)
```

其中，CompareFunction 为类型别名：一个指向特定函数的指针，此类函数返回值为 int，有两个相同类型的参数。

```
template<class T>
using CompareFunction= int (*)(T a, T b);
//这里我原本用 typedef，但是一直报错。查 cpp reference 建议用上述格式：
https://en.cppreference.com/w/cpp/language/type_alias
```

当 a>b 时返回 1，=返回 0，<返回-1；也可以>返回-1，<返回 1，使得排序结果为降序。

CompareFunction 的实现，例如：

```
int cmp_id(student a, student b)
{
    if (a.id > b.id)
        return 1;
    else if (a.id == b.id)
        return 0;
    else
        return -1;
}
```

也可以把 CompareFunction 定义为：当 a 与 b 的第一个排序关键字相同时，比较第二个。

之后根据输入的关键字，调用相应的比较函数作为排序的参数即可。

根据关键字判断调用比较函数一般使用 if...else if...else，有点麻烦。这里提出一个并不一定更好，但是我觉得非常有意思的解决方案。

```
string aim;
cin >> aim;
H(aim) = (-(int)aim.at(1)) % 14;
```

此时有：

关键字	学号	姓名	性别	院系	年级	班级	退出
(int)aim.at(1)	-89	-43	-44	-70	-22	-32	-53
H	5	1	2	0	8	4	11

在尝试取.at(1)之前我试过取 0 位，但是 0 位有重复，肯定不能单独取。

关于余数：我用 matlab 从 7 试到 19，只有 14 最合适。我甚至想过要平方取中或者折叠法设计一个复杂点的 hash 函数。大概这就是运气吧，直接 mod 14 就行了。而且一共 6 种 cmp，放 cmp 的数组大小只用 9。

事实上，这样做仅仅是好玩，如果运气不好，计算量会特别大...

这道题挺坑的，给的 txt 文件是 Unicode 编码，string 一定要读 ANSI。如果不转换，即使程序正确读出来也是乱码。题目要求对各字段分别排序，如果没想到函数参数能放函数，我估计要写 6 遍排序（说起来用复制粘贴+查找替换没准比用不熟悉的方法更快，反正粘 5 次就行，又不是 50 次）。再加上 6 个判断关键字的 if else if...写起来挺烦躁的。

第三题改进 shell 排序时不知道哪里出了问题，改进反而变慢了，可能是判断加得太多了，但是不加也不行 T_T