

《数据结构》课程设计总结

学号： 1652228

姓名： 王哲源

专业： 计算机科学与技术

2018 年 9

目录

第一部分 算法实现设计说明	2
1.1 题目	2
1.2 软件功能	2
1.3 设计思想	2
1.4 逻辑结构与物理结构	2
1.5 开发平台	3
1.6 系统的运行结果分析说明	3
1.7 操作说明	7
第二部分 综合应用设计说明	11
2.1 题目	11
2.2 软件功能	11
2.3 设计思想	11
2.4 逻辑结构与物理结构	13
2.5 开发平台	14
2.6 系统的运行结果分析说明	14
2.7 操作说明	17
第三部分 实践总结	21
第四部分 参考文献	22

第一部分 算法实现设计说明

1.1 题目

堆的建立和筛选

输入一组关键值，用堆排序的方法进行从小到大的排序

1.2 软件功能

- (1) 可以实现从小到大的排序，输出并显示该结果
- (2) 可随时显示操作的结果

1.3 设计思想

输入时利用数组暂存，通过 spinbox 向用户逐个获取被排序数组的数值，以此来回避因用户键盘误操作导致的输入数据出错的情况

在用户输入完毕并点击完成键后进入排序状态，每当用户点击一次按钮，堆进行一次出堆以及更新操作，由此来展示堆的排序过程。

而堆则是直接建立于一个数组之上，每次将堆顶元素即排序好的数字放至堆尾，并将堆的大小减一，由此节省空间

1.4 逻辑结构与物理结构

1.4.1 逻辑结构

根据堆的特性，其逻辑结构为一棵二叉树，根据题目要求，本次维护的为一棵大根堆，（注：由于排序是从小到大排序，而堆排序每次会将堆顶元素移至堆尾，因此需要维护的是大根堆

而不是小根堆)即对每一棵子树而言,其根节点的值永远大于左右子树中任意一个节点的值。

1.4.2 物理结构

本次算法实现设计一共使用了两个存储单元,一个对输入的值进行暂存,一个则为堆的存储。

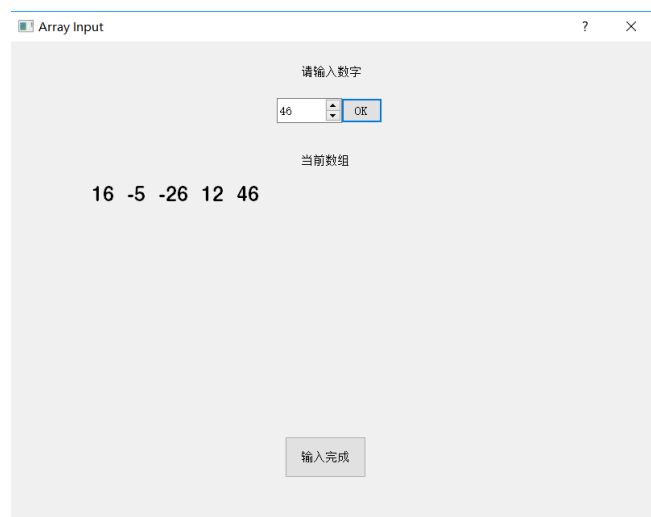
考虑到时间效率与操作复杂性的问题,两者均为顺序存储,但互相略有区别:暂存数组由于无法确定输入的数值个数,因此采用的是预设大小,当大小不足时,对当前空间进行翻倍申请,以此来适配不确定输入的情况。而堆的存储由于输入已经完成,因此只需直接申请所需的空间即可

1.5 开发平台

基于 Qt Creator 4.6.2 平台,使用 C++ 语言进行开发设计

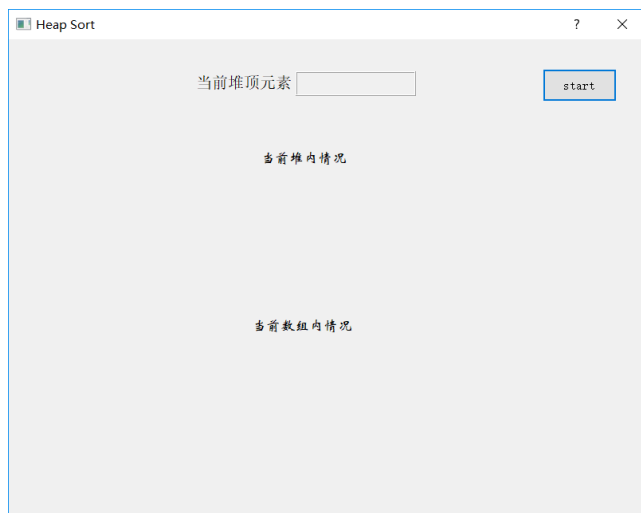
1.6 系统的运行结果分析说明

数组的输入:



成功得到对应数组反馈

输入完成，跳转至排序界面

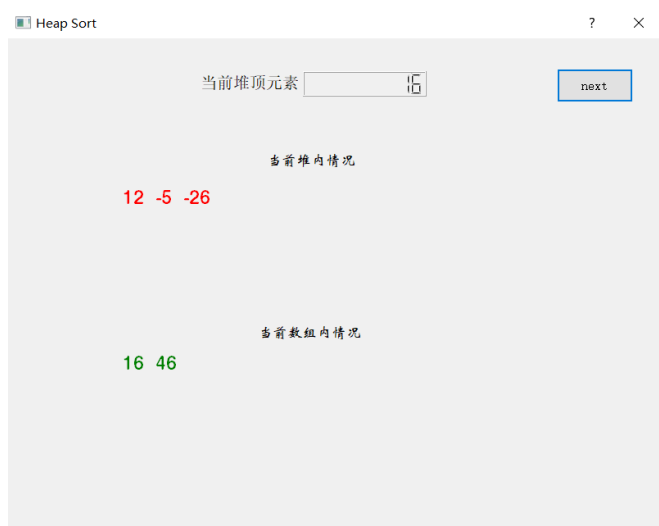


排序暂未开始，无任何反馈

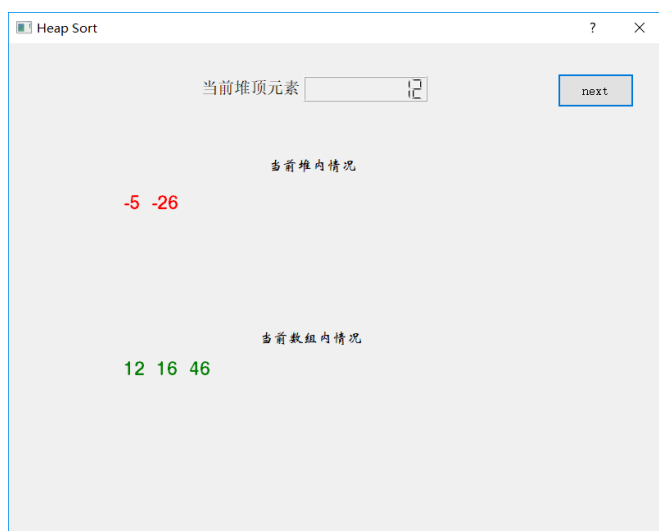
排序开始



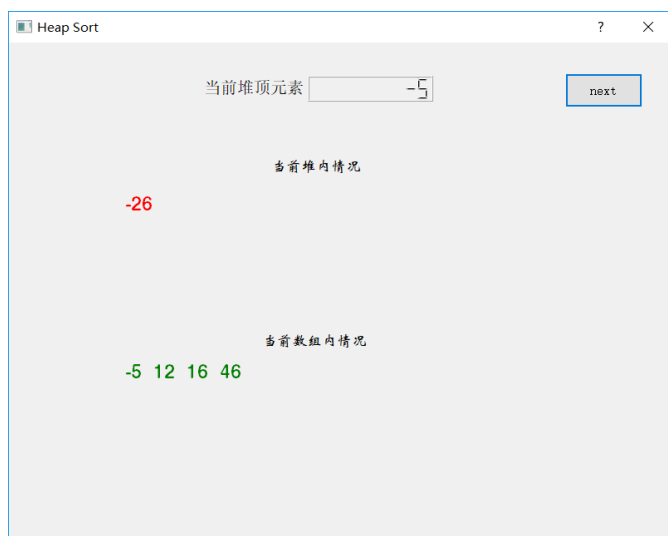
获得第一个堆顶元素 46，同时可以得到当前堆内对应数值结构



获得第二个元素



获得第三个元素



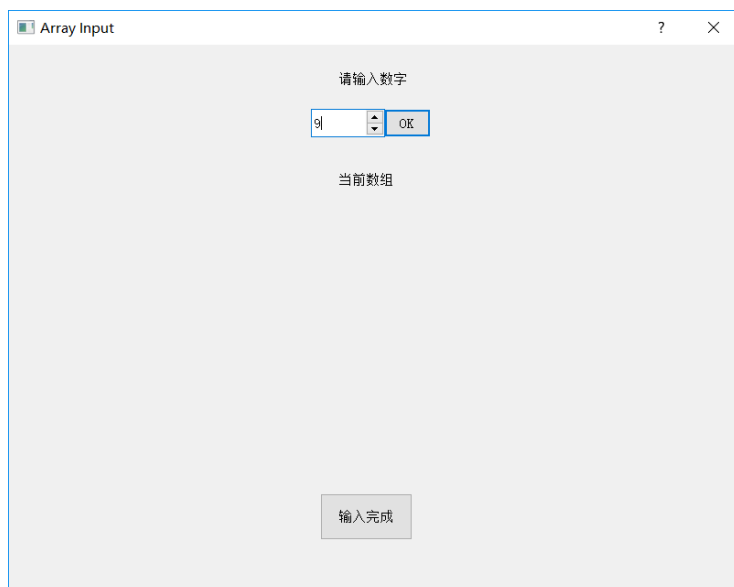
获得第四个元素



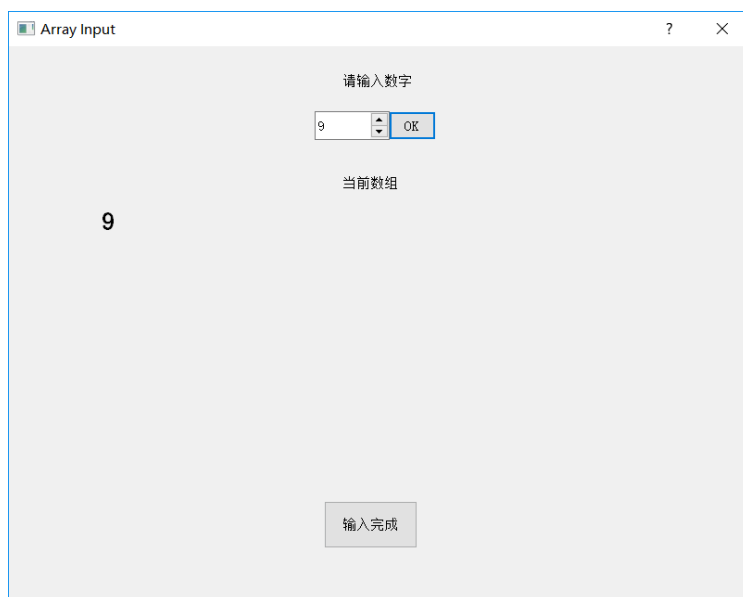
排序结束，可以看到数值已经从小到大进行分布

1.7 操作说明

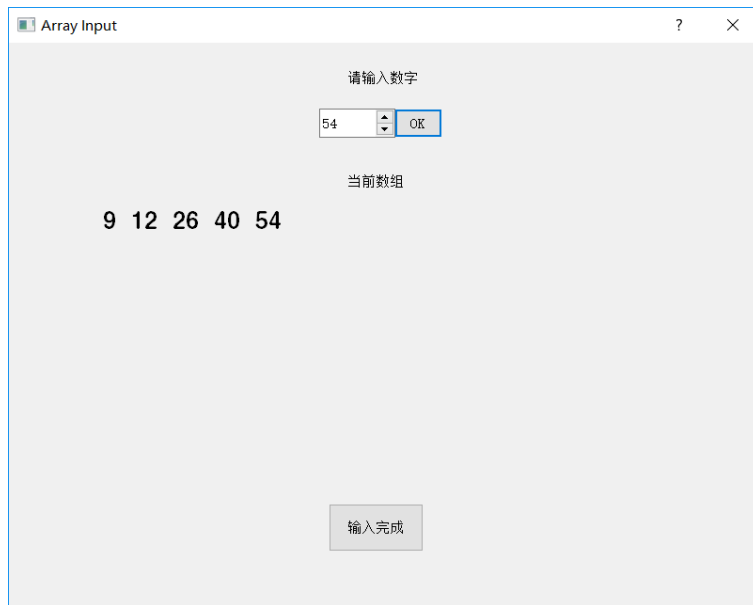
1.7.1 数组的输入



窗口上方的 spinbox 可以通过内置的箭号、滚轮、数字键输入的方式进行数值的键入，但需要注意，可键入的数值范围在 -2147483646 到 2147483647 之间

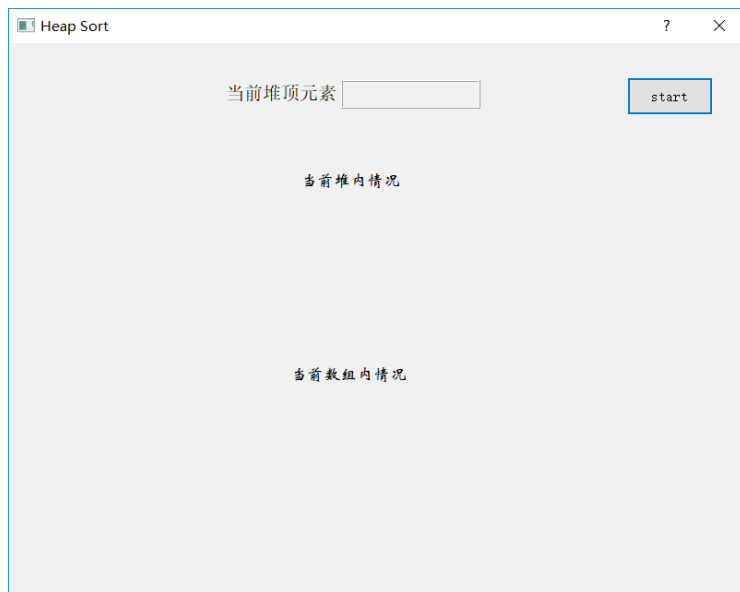


点击 OK 键之后，下方会得到回显，对应的即为当前数组内的数值情况



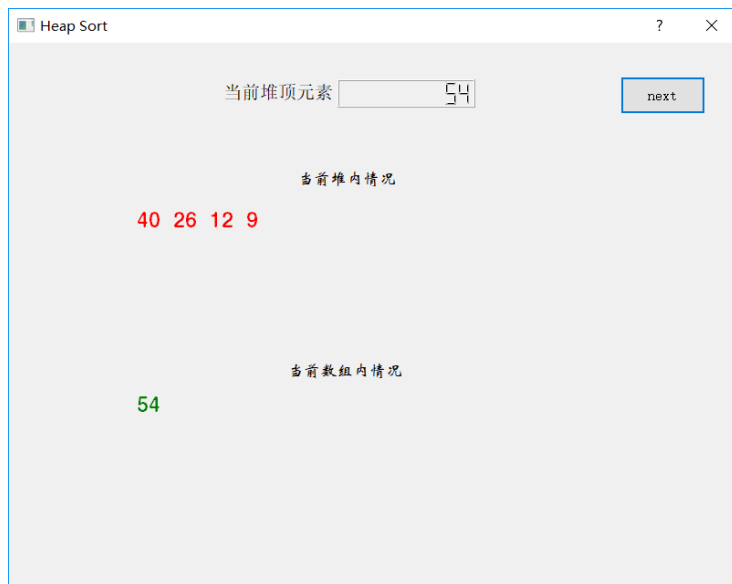
A screenshot of a Java Swing dialog box titled "Array Input". The dialog has a light gray background and a blue border. At the top, there is a title bar with a green icon, the text "Array Input", and standard window controls (minimize, maximize, close). The main content area contains the following elements: a label "请输入数字" (Please enter a number) in black text; a text input field containing the number "54" with a small blue border; an "OK" button with a blue border; a label "当前数组" (Current array) in black text; a list of numbers "9 12 26 40 54" in bold black text; and a "输入完成" (Input completed) button with a gray border at the bottom center.

输入完成后点击下方输入完成，输入界面会关闭



进入新窗口准备排序，此时排序窗口无回显，按钮显示 start

点击 start 后进入排序，按钮变为 next



上方显示为上一次排序结束的堆顶元素，中间一栏为当前堆内状况，最后一栏显示的是将上一次排序结束的堆顶元素移动至堆尾得到的有序数组情况

不断点击 next 至堆内为空后，按钮变为 finish



可以看到下方数组已经从小到大排序，再点击 finish，窗口自行关闭，整个排序完成

第二部分 综合应用设计说明

2.1 题目

上海的地铁交通网路已基本成型，建成的地铁线十多条，站点上百个，现需建立一个换成指南打印系统，通过输入起点站和终点站，打印出地铁换成指南，指南内容包括起点站、换乘站、终点站

2.2 软件功能

- (1) 图形化显示地铁网络结构，能动态添加地铁线路和地铁站点
- (2) 根据输入起点站和终点站，显示地铁换成指南
- (3) 通过图形界面显示乘车路径

2.3 设计思想

2.3.1 地铁站点的存储

由于实际站点名为英文/中文的缘故，无法对其直接编号，因此采用 STL 中的 map 进行存储。STL 的 map 底层为红黑树实现，其包含两个成员，key 与 value，key 为唯一的键值，在同一个 map 内只能出现一次，并且可以根据 key 值对其对应的 value 值进行快速查询，因此将地铁站点名作为 key 值加入 map 中，并对其进行编号作为 value，这样可以快速将站点离散化，方便进一步操作

2.3.2 地铁线路的存储

在将地铁站点离散化后，线路的存储则直接使用离散化后的 value 值。而对于线路的存储，

则是进行两份存储，一份存于数据结构之中，使用边表进行存储，方便寻路时快速遍历；另一份存储于图形化界面之中，使用 multimap 存储。由于 map 中 key 值不可重复的特性，其不适合存储地铁线路，而 multimap 允许 key 值重复的特性则完美避免了这一情况。并且为了解决相邻站点被多条线路经过的情况，这里对 value 值进行修改，存储类型变为 pair<int, int>类，前者为目标节点，后者为所属线路，由此解决线路重复问题。同时由于地铁双向边的特性，存储时默认起点为编号小的节点，避免重复存储的情况出现。

2.3.3 地铁站点/线路的添加

利用 map 与 multimap 快速查找的特性，可以查询当前站点与线路是否已经被添加过，若存在则不对本次添加操作进行响应，避免重复添加导致的系统错乱与内存浪费。同时为了防止线路添加时输入非法站点名，此处采用下拉栏选项的方式供用户进行操作

2.4.4 路径的寻找

根据地铁线路较为稀疏的特性，暂时仅采用了广度优先搜索（bfs）的方式进行寻路。中途不考虑换乘与时间消耗的情况，仅对经过的点进行存储，在打印时再对所选乘线路进行判断。对于路径的存储，这里对每个节点建立一个 fro 值，用于记录其是被哪个节点扩展的，寻路结束后通过反向遍历 fro 的方式即可获得对应路径

当然，具体实现时还需考虑换成耗时等情况，但受限于时间因素这里仅提出一个设想，具体实现可待未来优化继续实现：

采用多线模拟的方式，对于线路重复站点建立多层节点，并针对稀疏图特性采用 spfa 即 bellman-ford+队列优化算法进行最短路计算

2.4.5 换乘信息的获取

由最短路获取到路径上各个节点后，通过遍历的方式，每次可以获取到路径上相邻的两个节点，这时通过在 multimap 中寻找对应的边，可以获取到若干条经过这两个点的线路。将其暂存于一个数组之中，当获得下一条路径时与其比对，若仍有重复的线路，则保存被重复的线路，否则说明需要在本站换乘

2.4.6 图形化界面的打印

由于地铁线路存在不同走向的结构，这对于作图而言存在着极大的不便利，因此这里才用活动节点式，使节点能够被鼠标拖动，以此来规避由于地铁走向而导致加点时造成的不便利。并且根据所属线路不同，其边的颜色也会有所不同

2.4 逻辑结构与物理结构

由于这里采用了部分 STL 数据结构进行辅助存储，因此只叙述最短路径所应用的存储结构

2.4.1 逻辑结构

图的存储采用了图形结构，即边表，一个数组用于存储从每个点出发的边所属的边表链头，另外两个数组则存储每条边与其共起点的前一条边的编号以及当前边所能到达的目的地。

2.4.2 物理结构

这里的物理结构采用了用顺序结构模拟的链式结构，这么做的优点在于不用频繁的申请内存，在边表占满后只需要加倍申请即可

2.5 开发平台

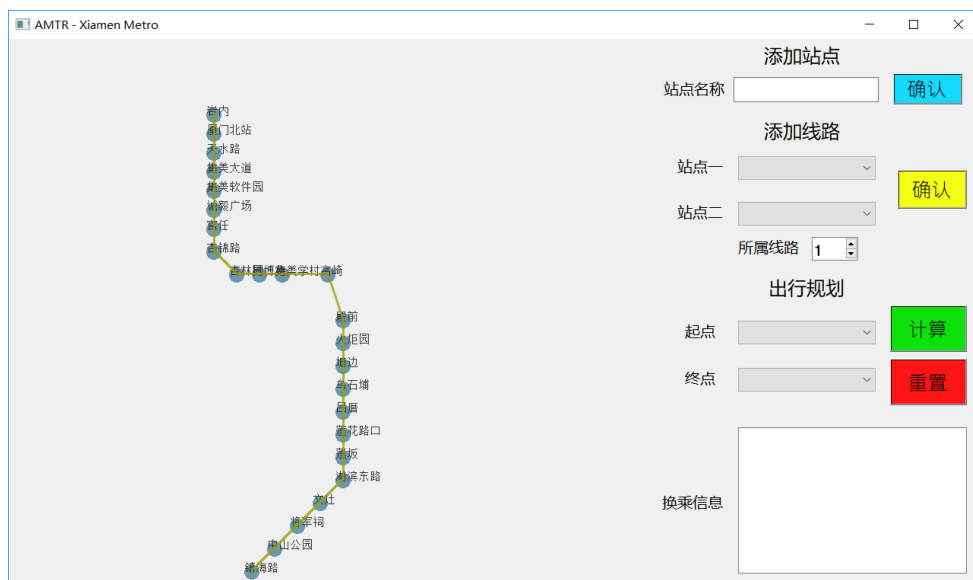
基于 Qt Creator 4.6.2，使用 C++ 以及其标准库进行开发

2.6 系统的运行结果分析说明

(注意一：这里采用厦门地铁线路作为测试样例，已内置一号线全线)

(注意二：由于在编写过程中对一些绘制函数进行重写，但其中某些形参并未被使用，因此会报“warning: C4100: “event”: 未引用的形参”但不影响程序使用即稳定性，特此说明)

程序运行进入后初始界面

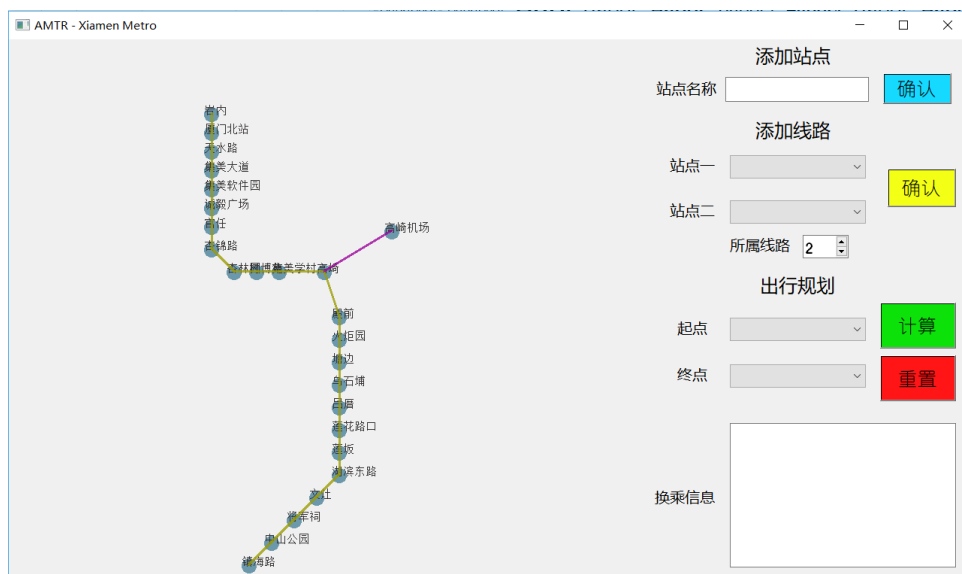


添加站点“高崎机场”后



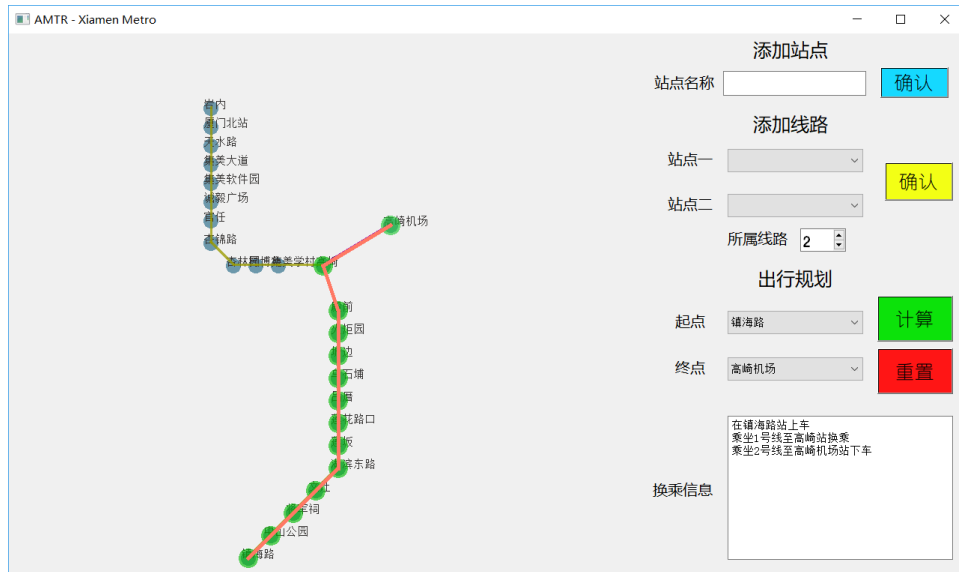
可以看到站点被添加

稍微鼠标修正位置，将其与“高崎”站，利用二号线连接



线路被添加，并且显示为与一号线不同的紫色

在规划里选择从“镇海路”出发，前往“高崎机场”

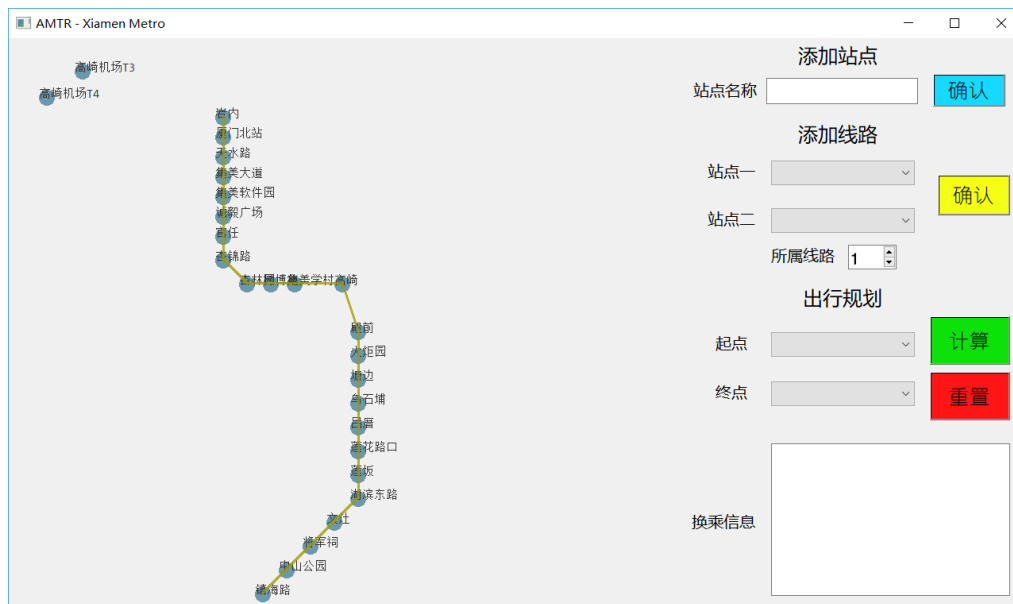
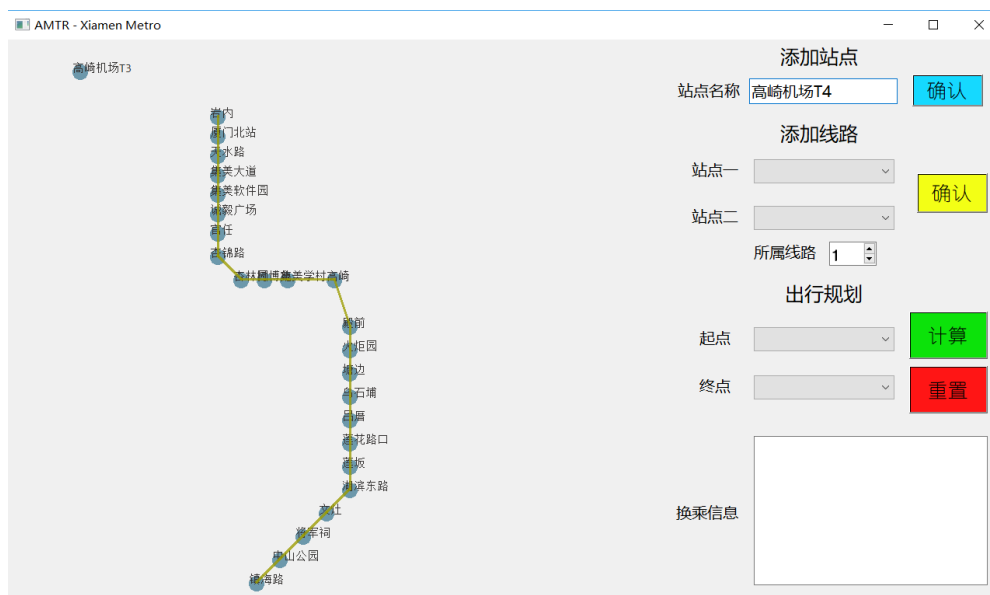


路线被加粗，所经站点变色，同时右下角显示对应换乘信息

2.7 操作说明

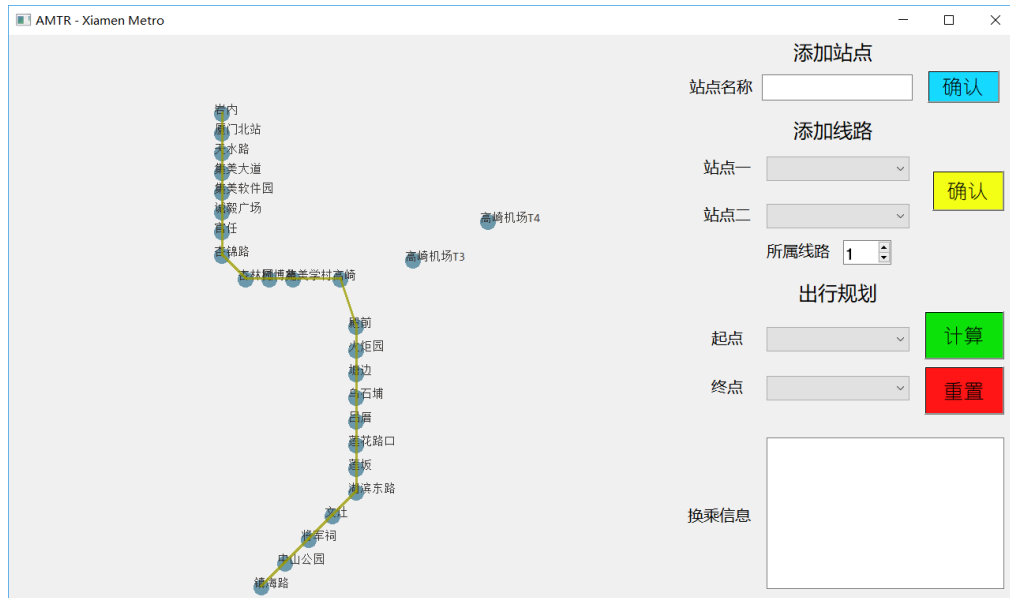
2.7.1 站点的添加

在右侧添加站点框下输入站点名，再点击确认，对应站点便会被加至左上角任意位置



2.7.2 站点位置的移动

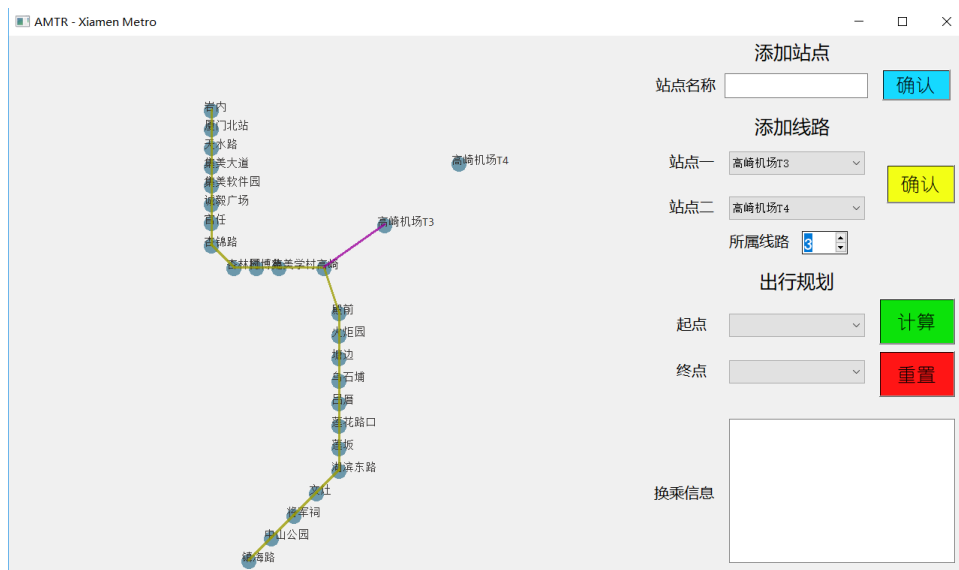
鼠标点击对应站点，按住鼠标左键不松即可拖动站点，再松开左键即可停止移动

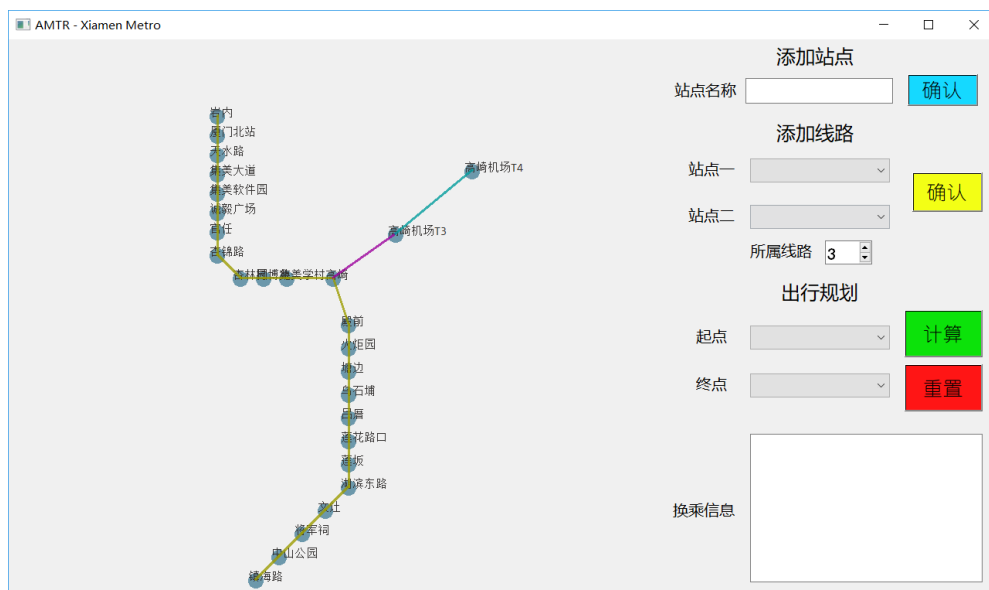


(图为移动好后的站点位置)

2.7.3 线路的添加

在添加线路对应下拉栏中选择线路的两端，并选择所属线路

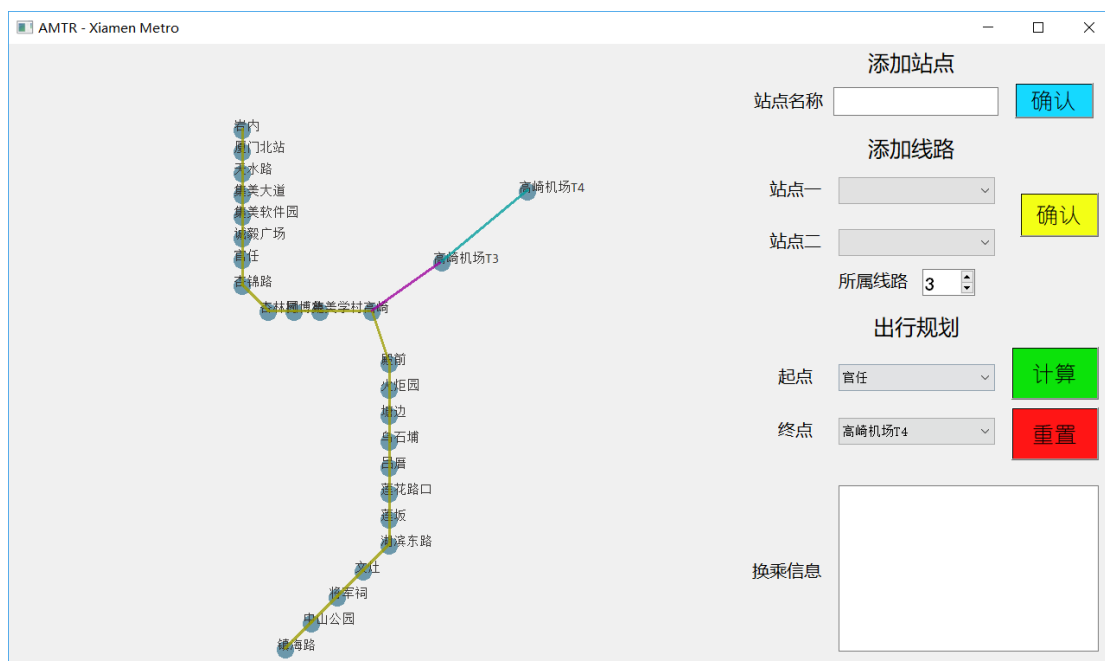




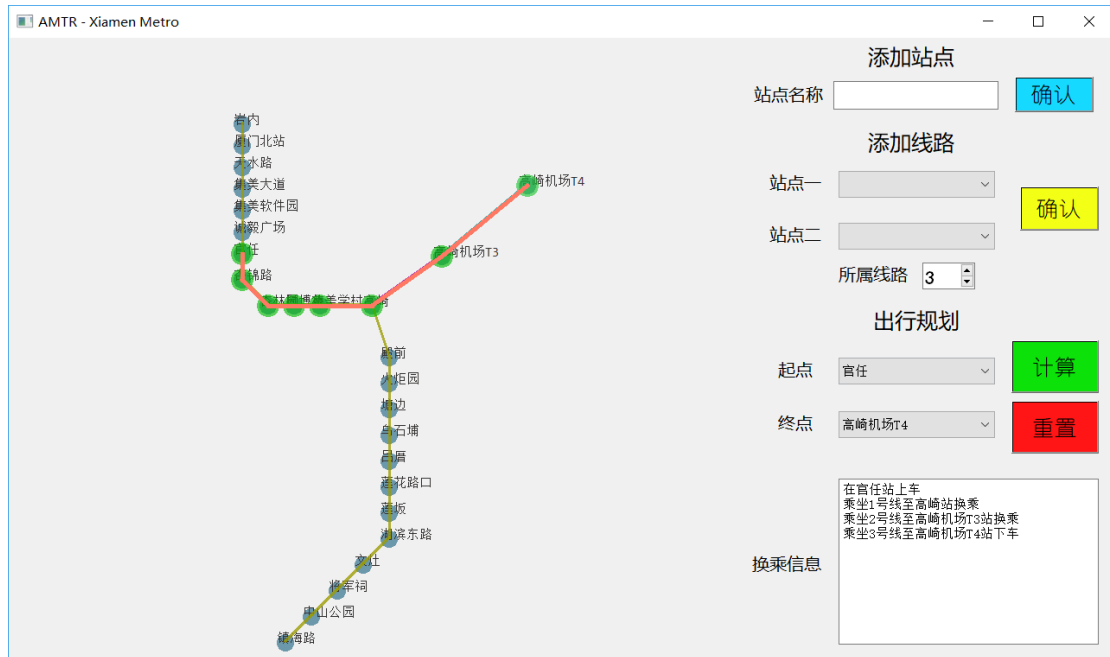
这里添加高崎到高崎机场 T3 的二号线与高崎机场 T3 至高崎机场 T4 的三号线

2.7.4 路径的计算

在出行规划的下拉栏中选择起点站与终点站

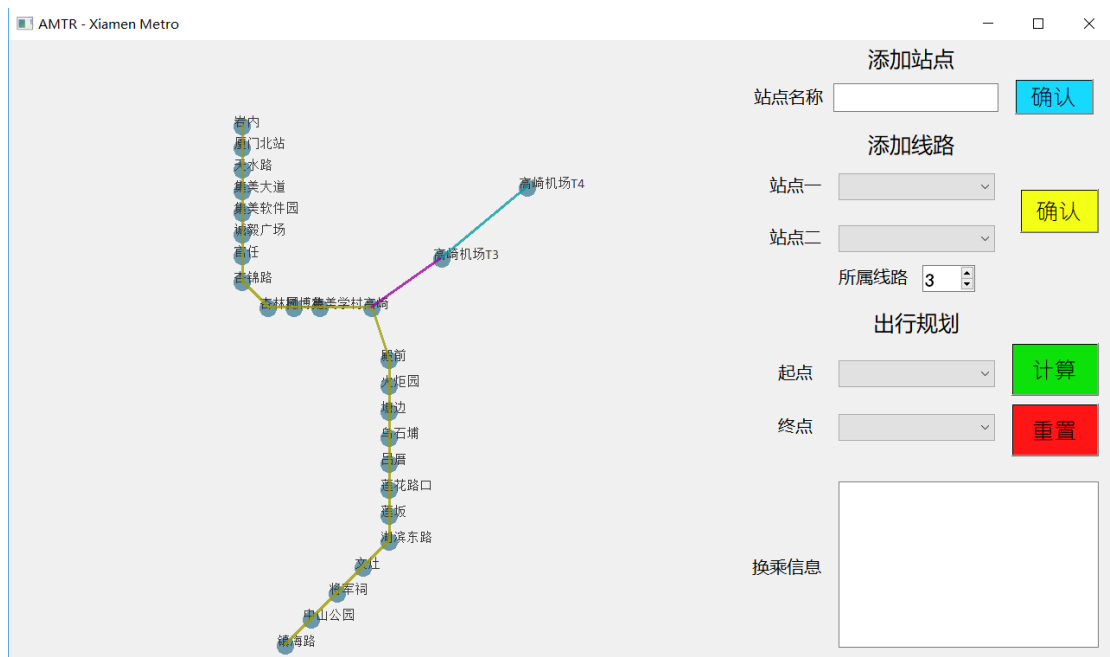


点击计算



即可获得对应的路线

若需重置线路，选择重置即可回到初始



第三部分 实践总结

3.1 所做的工作

本次课程设计主要通过 Qt Creator 内置的设计师模式，对 ui 界面以及对应的按键等槽函数进行了编写。

通过 C++ 对程序的主要数据结构进行了编写，并应用 STL 标准库内置的一些模板类对数据等内容进行了辅助存储

3.2 总结与收获

通过这次数据结构的课程设计，使我真正的将自己所学到的数据结构知识应用在了实际应用之中。之前仅仅是停留在做题的层面，而这一次则是通过亲自设计应用 UI 界面以及编写内部存储结构的方式初步做成了一个可以被使用的“软件”。

同时，Qt Creator 让我学习到了与命令窗口编程不同的 GUI 编程，学习到了如何使用 Qt 库为自己的程序设计 UI 界面，并实现诸如键盘、鼠标等操作方式。而对于在设计之中所遇到的诸多问题，如数据结构的设计、UI 操作的设计等问题，通过自己的思考而解决，培养了自己解决问题的能力，并且通过自学的模式获取了课堂所不能获得的一系列经验知识，为将来的学习甚至工作积累了宝贵的经验。

当然，在设计之中也不免暴露了自己一些缺点，数次因为函数名重复、大小写出错而浪费了不少时间。而自己对于头文件使用的不够娴熟也让自己在开发设计之初一度面对着各种报错找不着头脑，设计中所遇到的由于设计之初未考虑完全导致程序需要大改造成的时间和精力浪费，这些都是应当十分注意的。

第四部分 参考文献

[1] <http://www.qter.org/thread-629-1-1.html>

Qt 学习之路

[2] <https://www.devbean.net/2012/08/qt-study-road-2-catelog/>

Qt 学习之路 2

[3] https://blog.csdn.net/mahabharata_/article/details/75571624?locationnum=7&fps=1

[Qt] 图的可视化编程 - Graph Visualization

[4] <https://www.cnblogs.com/CLXiao-1029/p/6892355.html>

Qt 之新手打包发布程序