Project 1
火车车厢重排调度

班级：16 级软件工程教务 4 班

姓名：叶梓豪 学号：16340277

姓名：袁之浩 学号：16340282

姓名：易昕炜 学号：16340278

【题目要求】

一列火车要将 n 节车厢分别送往 n 个车站，车站按 1-n 的次序编号，火车按照 n,n-1,n-2,....,1 的编号次序经过车站。假设车厢的编号就是其目的地车站的编号。

要求：给定一个任意的车厢排列次序。重新排列车厢，使其按照从 1 到 n 的次序排列。给出调度详细步骤，并统计出所用栈的个数及调度的总步数。规定重新调度时车厢只能从入轨到缓冲轨，或者从缓冲轨到出轨。

【数据结构与算法】

程序使用的数据结构：

1. 使用了元素为 stack 的 vector 来进行所有暂存区（隧道）的存储
2. 使用了元素为 int 的 vector 来模拟车厢和暂存区（隧道）
3. 实际上使用了堆栈模拟了整个火车车厢进站（压栈）和出站（弹

栈）的过程

具体的算法分析：

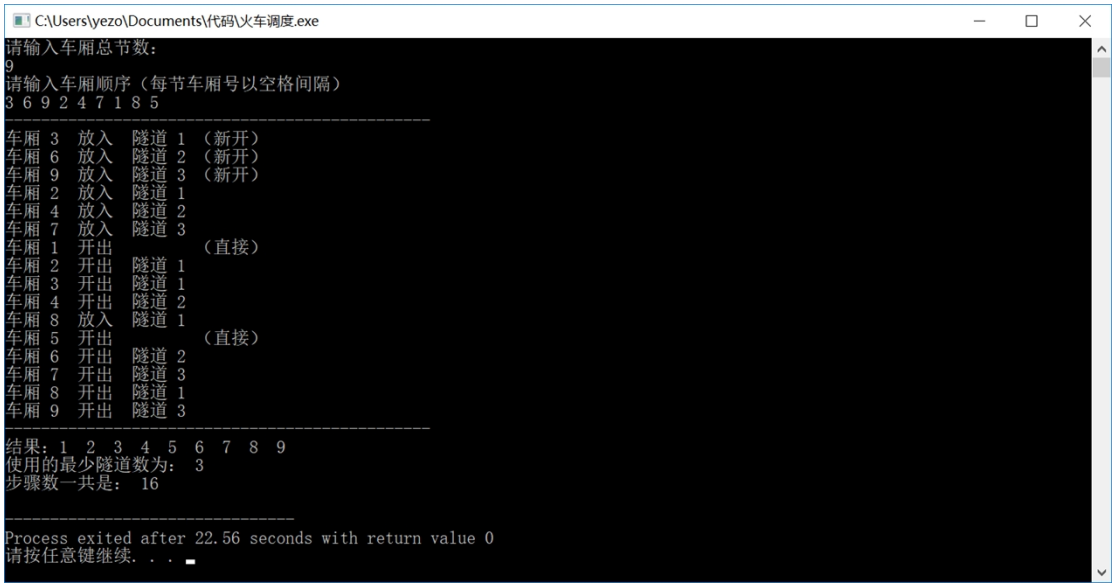
首先输入一串随机排列的数字，以输入从左到右为车厢的从头到尾，然后对从头开始对每一节车厢进行分析。先设定一个变量 J 值为 1，这个值的意义是如果下一节要判断的车厢号跟 j 值一致，就直接放出。

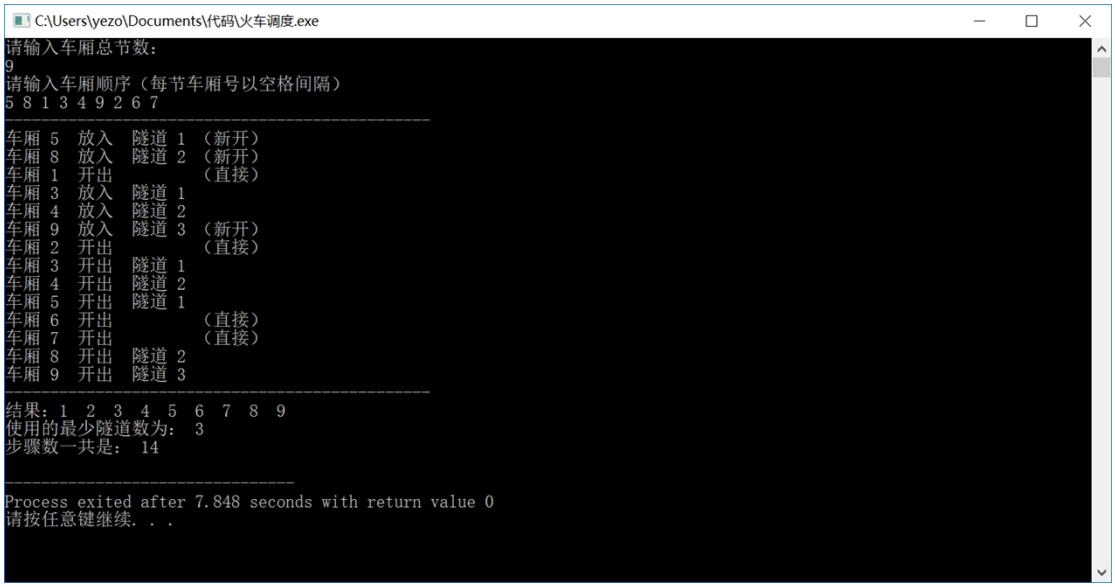
在每次准备判断新一节车厢的情况前，先将已有的隧道遍历一次，如果出现某一个隧道头部的车厢号=J 值，则放出该车厢，J 值加 1。调用递归，直到不存在上述情况。

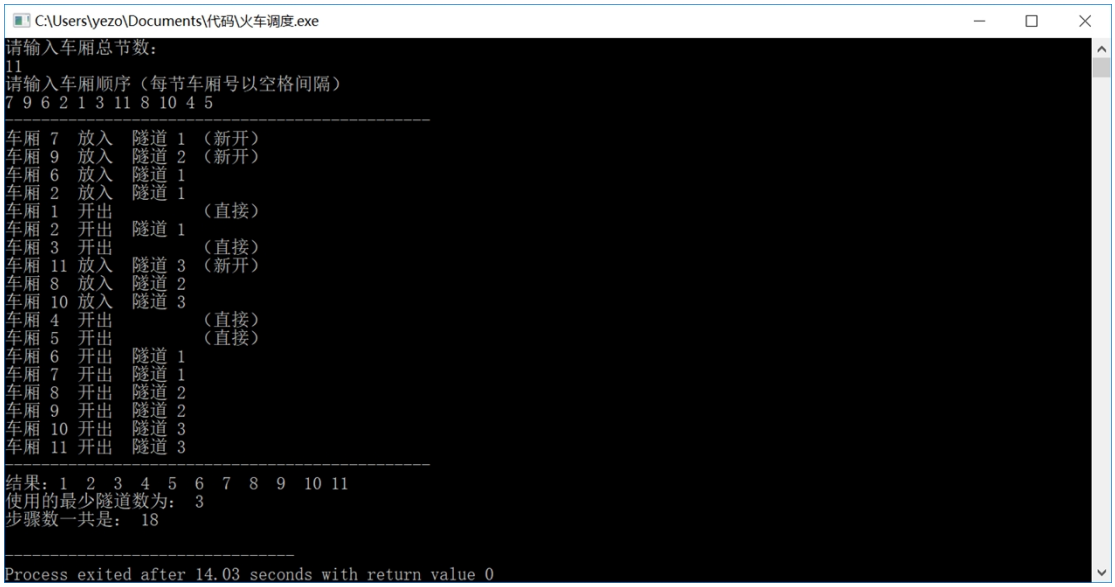
然后开始判断，设当前的车厢号为 A，如果为 A=J，就直接放出；如果 A != J 且已有的所有隧道中最上方的车厢号都小于这个 A 值，则新开一个隧道，放入 A 值；如果 A！=J 但存在某个隧道最上方的车厢号大于 A 值，则将这节车厢放到 |车厢号-A| 最小的那个隧道中。

开出的车放到另外一个元素为 int 的 vector 中，直到所有车厢按照顺序排列完成，结束程序。

【测试数据、结果及分析】


C:\Users\yezo\Documents\代码\火车调度.exe
请输入车厢总节数:
9
请输入车厢顺序（每节车厢号以空格间隔）
3 6 9 2 4 7 1 8 5
车厢 3 放入 隧道 1 （新开）
车厢 6 放入 隧道 2 （新开）
车厢 9 放入 隧道 3 （新开）
车厢 2 放入 隧道 1
车厢 4 放入 隧道 2
车厢 7 放入 隧道 3
车厢 1 开出 （直接）
车厢 2 开出 隧道 1
车厢 3 开出 隧道 1
车厢 4 开出 隧道 2
车厢 8 放入 隧道 1
车厢 5 开出 （直接）
车厢 6 开出 隧道 2
车厢 7 开出 隧道 3
车厢 8 开出 隧道 1
车厢 9 开出 隧道 3
结果：1 2 3 4 5 6 7 8 9
使用的最少隧道数为： 3
步骤数一共是： 16
Process exited after 22.56 seconds with return value 0
请按任意键继续. . .


C:\Users\yezo\Documents\代码\火车调度.exe
请输入车厢总节数:
9
请输入车厢顺序（每节车厢号以空格间隔）
5 8 1 3 4 9 2 6 7
车厢 5 放入 隧道 1 （新开）
车厢 8 放入 隧道 2 （新开）
车厢 1 开出 （直接）
车厢 3 放入 隧道 1
车厢 4 放入 隧道 2
车厢 9 放入 隧道 3 （新开）
车厢 2 开出 （直接）
车厢 3 开出 隧道 1
车厢 4 开出 隧道 2
车厢 5 开出 隧道 1
车厢 6 开出 （直接）
车厢 7 开出 （直接）
车厢 8 开出 隧道 2
车厢 9 开出 隧道 3
结果：1 2 3 4 5 6 7 8 9
使用的最少隧道数为： 3
步骤数一共是： 14
Process exited after 7.848 seconds with return value 0
请按任意键继续. . .



如图所示，程序先要求输入一串车厢号，必须是 1~n 中的所有数以随机顺序排列（即不可输入跳号的车厢）。

输入完成后，我们将每一个步骤都打印了出来。然后在最后打印了输出结果，一共使用了的隧道数和一共的步骤数。

【分工、贡献%、自我评分】

叶梓豪：

分工：构思并实现算法，完成代码注释，加入了统计隧道数和步骤数的功能，撰写报告。

贡献：40%

自我评分：98 分

袁之浩:

分工：实现算法，完成代码的框架和编写，完善了题目要求的地方，提出

了用 vector 和 stack 的建议，大大简便了代码量。

贡献：40%

自我评分：98 分

易昕炜:参与构思和实现算法，提出了建议，对程序进行测试

贡献：20%

自我评分：95 分

【项目总结】

通过这次 project 的实验，我们大大加深了对 stack（堆栈）的了解，获取了一些 stack 的使用技巧和使用情景模拟。Stack 是非常重要的一个数据结构，可以利用其解决很多生活中的问题。

而我们在实验当中的一个不足就是，我们一开始使用的是自定义的 stack 和数组，因此十分繁琐。后期改用了 stl 中的 stack 和 vector，大大降低了代码量。