**中国矿业大学**

**计算机科学与技术学院**

**2021级本科生课程设计报告**

课程名称 算法设计与分析A

设计题目 用不同策略求解TSP问题

报告时间 2023.11.20

学生姓名 杨学通

学 号 08213129

班 级 人工智能2021-1班

任课教师 张辰

目录

[1 问题引入 1](#_Toc151463031)

[2 TSP-穷举法 1](#_Toc151463032)

[2.1 问题分析 1](#_Toc151463033)

[2.2 数据结构 2](#_Toc151463034)

[2.3 实现函数 3](#_Toc151463035)

[2.4 输出分析 3](#_Toc151463036)

[3 TSP-贪心选择 4](#_Toc151463037)

[3.1 问题分析 4](#_Toc151463038)

[3.2 实现函数 4](#_Toc151463039)

[3.3 输出分析 5](#_Toc151463040)

[4 TSP-回溯法 5](#_Toc151463041)

[4.1 问题分析 5](#_Toc151463042)

[4.2 实现函数 6](#_Toc151463043)

[4.3 输出分析 7](#_Toc151463044)

[5 TSP-分支限界法 7](#_Toc151463045)

[5.1 问题分析 7](#_Toc151463046)

[5.2 数据结构 8](#_Toc151463047)

[5.3 函数实现 9](#_Toc151463048)

[5.4 输出分析 10](#_Toc151463049)

[6 TSP-动态规划法 10](#_Toc151463050)

[6.1 问题分析 10](#_Toc151463051)

[6.2 数据结构 10](#_Toc151463052)

[6.3 实现函数 11](#_Toc151463053)

[6.4 输出分析 12](#_Toc151463054)

[7 TSP-分治法 12](#_Toc151463055)

[7.1 问题分析 12](#_Toc151463056)

[7.2 实现函数 13](#_Toc151463057)

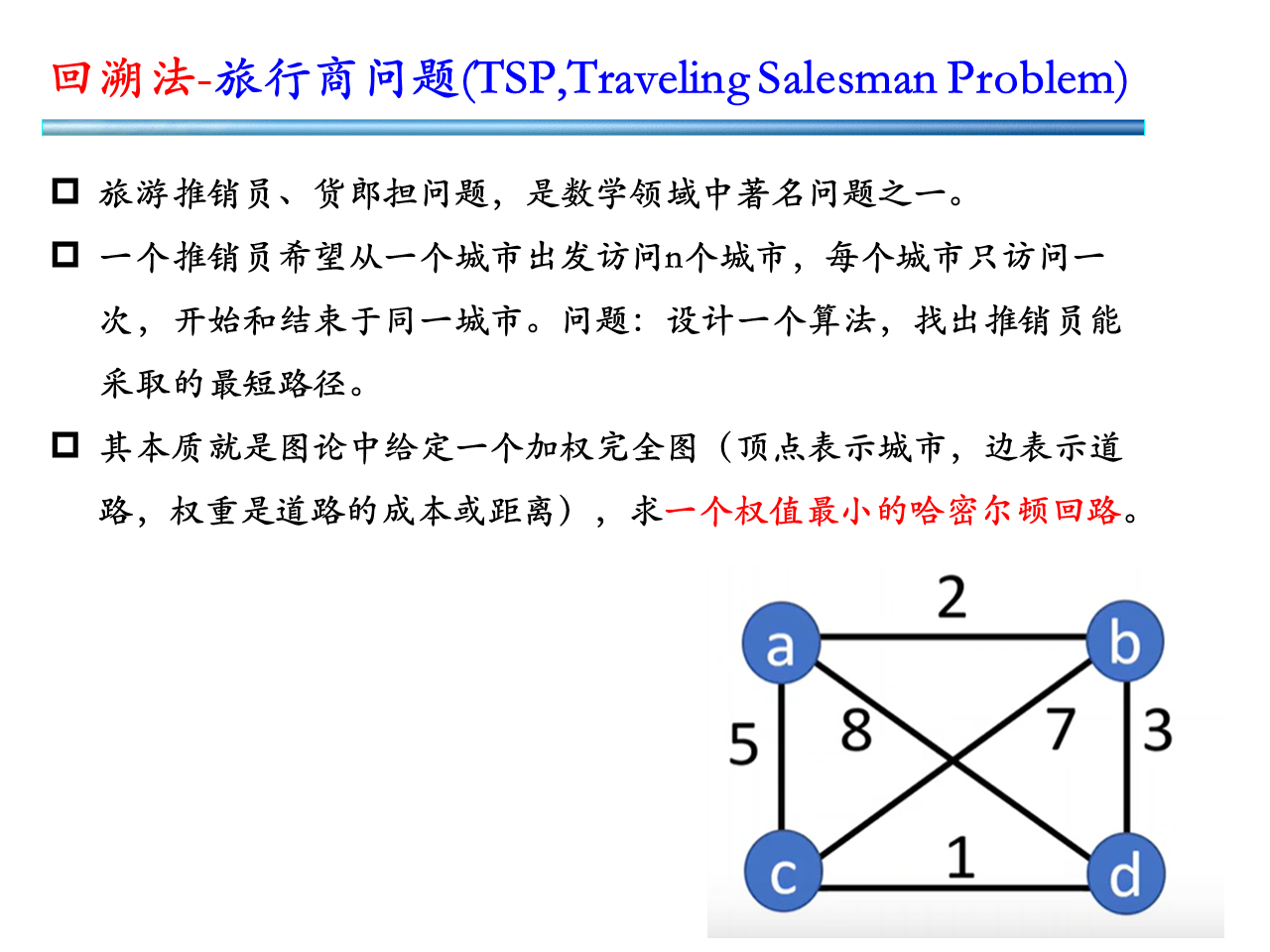
[7.3 输出分析 13](#_Toc151463058)

[8 感悟 14](#_Toc151463059)

# 1 问题引入

TSP（旅行商问题）是指：一个旅行商希望从一个城市出发访问n个城市，每个城市只访问一次，开始和结束与同一个城市。问题：设计一个算法，找出推销员能采取的最短路径。

旅行商问题的本质是：图论中给定一个加权完全图（顶点标识城市，边表示道路，权重是道路的成本或距离），求一个权值最小的哈密尔顿回路。

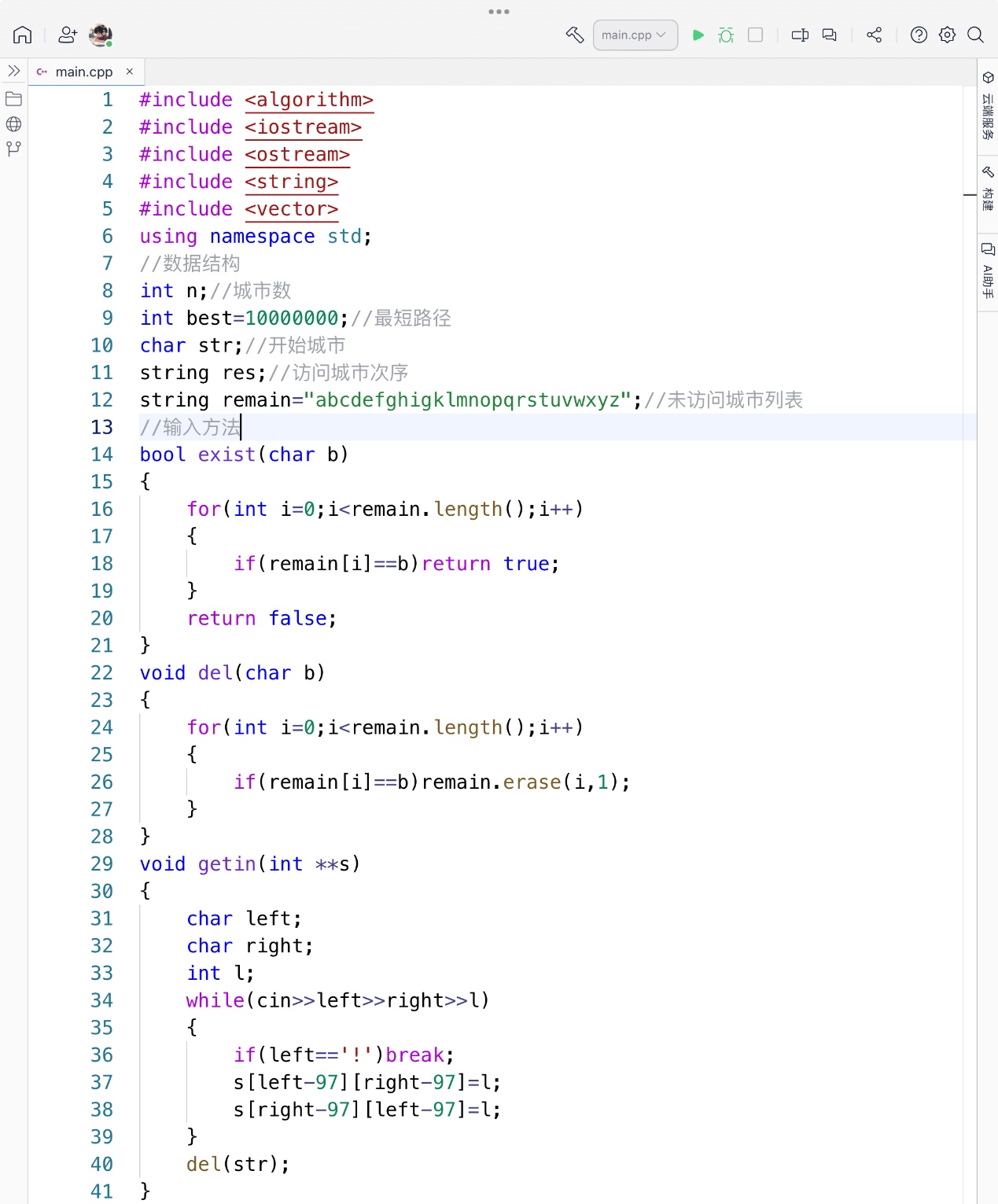


# 2 TSP-穷举法

## 2.1 问题分析

穷举法是指，每次到达一个城市，选择下一个城市作为目的地，直到最后遍历完所有的城市，返回出发城市。其求解过程是求除了出发城市以外城市的全排列。当遍历完所有的城市之后，更新最小路径，即出现一个比之前所有情况还要短的路径时，将当前最短路径更新为新值。

## 2.2 数据结构



主要的数据结构有：

* 邻接矩阵：他存放的是城市与城市之间的路径长度，元素的下标表示城市
* 剩余城市列表：存放未访问的城市列表
* 结果列表：存放已经访问的城市列表
* best：最短路径

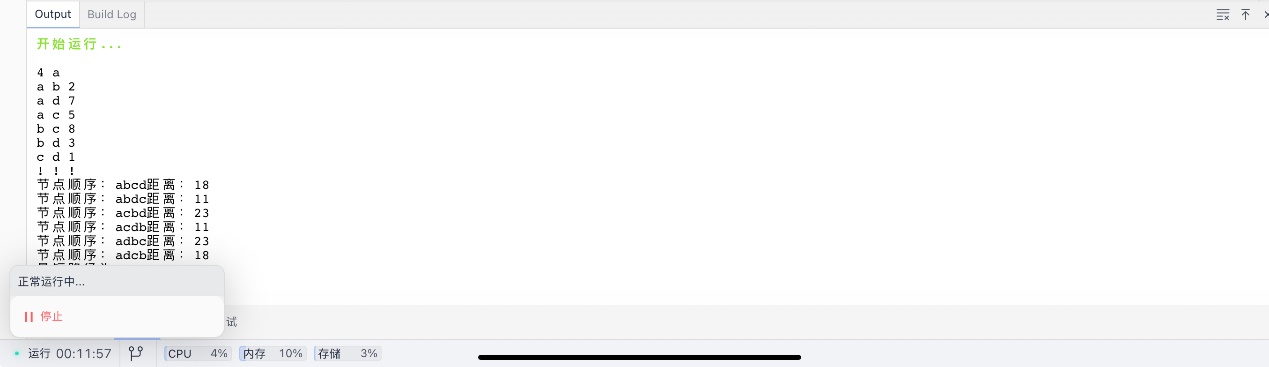
图中还定义了输入函数，后边不再赘述。

## 2.3 实现函数



采用排列树的思想：采用深度优先搜索，进入一个节点，就更新当前代价，并按照剩余城市列表里的顺序，访问下一个邻接节点。

## 2.4 输出分析



输出按照深度优先搜索，并输出所有可能存在的路径，比较出最短路径后输出。

# 3 TSP-贪心选择

## 3.1 问题分析

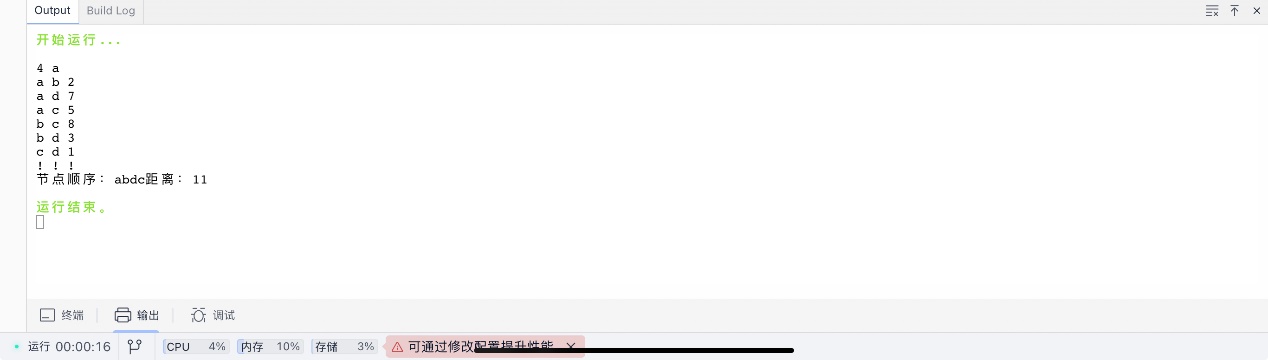
贪心算法，故名思意就是只考虑当前的最优策略。本题的贪心策略为，每次到达一个城市，就从剩余城市中选择与当前城市距离最短的城市，并更新代价。贪心选择策略虽然能找到一条路径，但不能保证该路径为最优路径。

## 3.2 实现函数

由于贪心选择策略的数据结构和枚举法一致，这里不再赘述。



## 3.3 输出分析



搜索按照贪心选择策略选取当前城市的最短出边，故只有一个输出结果

# 4 TSP-回溯法

## 4.1 问题分析

回溯法求解TSP问题的思路如下：

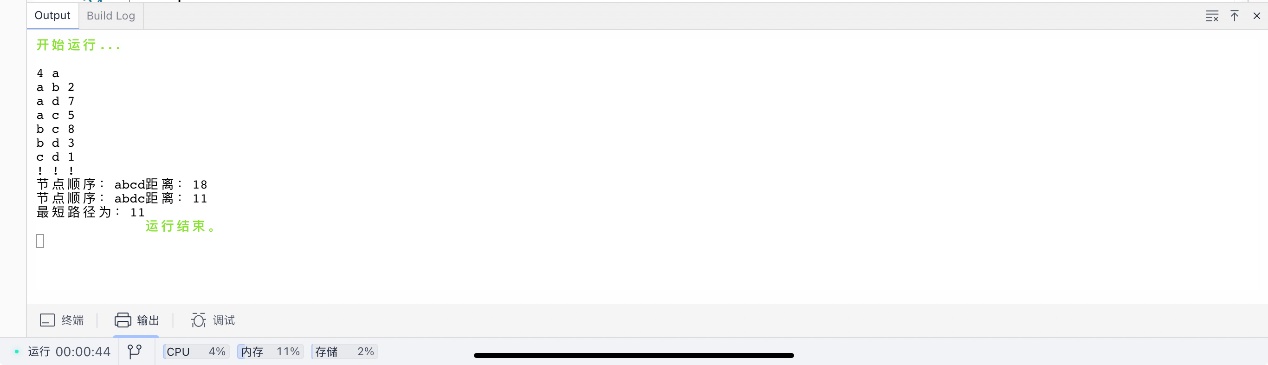
在深度优先搜索算法的基础上，增加剪枝函数。左剪枝函数思路为如果当前的代价已经大于当前最优路径，则剪枝；右剪枝的思路为如果剩余城市则最小出度和+当前代价>最优路径，则剪枝

## 4.2 实现函数

****

添加out函数求解剩余城市的最小初读之和。搜索路径的剪枝过程和回溯过程，已经在图中注释。

## 4.3 输出分析

****

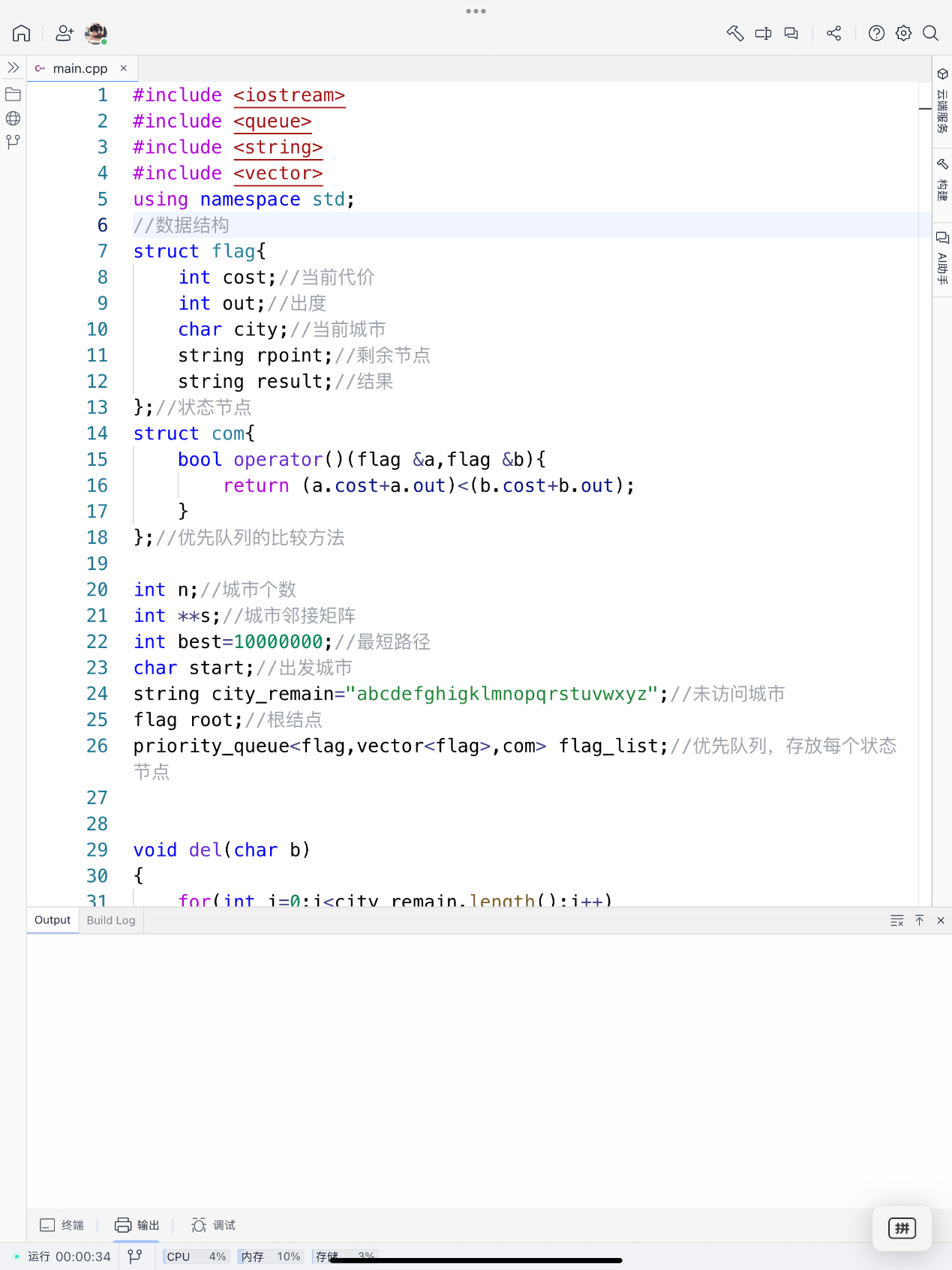
按照深度优先搜索的方式，查找路径。当找到一个当前最优路径时，更新最有路径的值。拓展左子节点时，遇到当前代价>best时，回溯；拓展右子节点时，遇到当前代价+剩余城市最小出度和>best时，剪枝。故输出按递减结果排列。

# 5 TSP-分支限界法

## 5.1 问题分析

采用广度优先遍历，对到达的一个城市，将其临接的城市及其各种信息（如代价、城市名等）压入优先队列，优先队列根据（剩余城市出度和+当前代价）为判断标准，对压入对列的城市节点排序。

## 5.2 数据结构



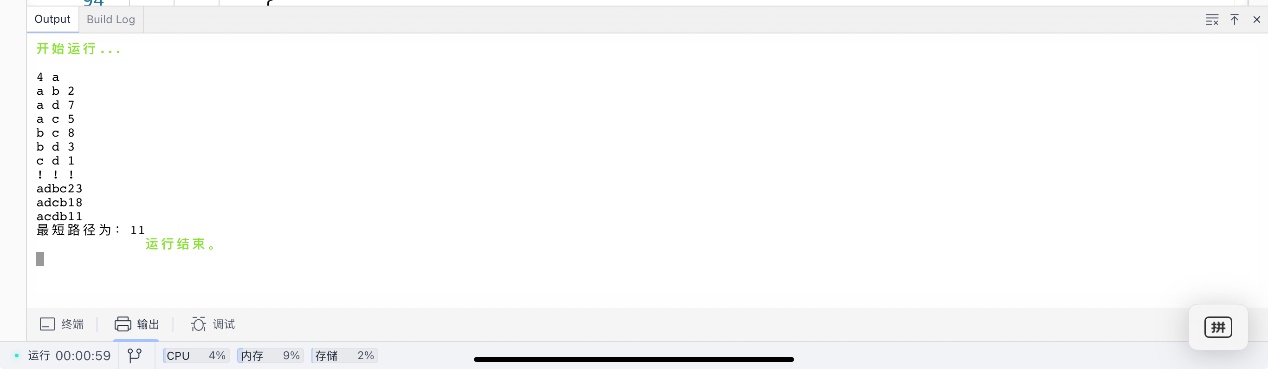
主要的数据结构如下：

* + 优先队列：按城市优先级，存放城市节点的各种状态信息
  + 城市节点结构体：存放拓展的子城市节点的状态（城市名称，当前代价，剩余城市节点最小出度、剩余城市列表、结果列表）
  + 优先队列排序方法结构体：定义了优先队列的排序方式

## 5.3 函数实现



## 5.4 输出分析



按照广度优先搜索的方式，查找路径。当找到一个当前最优路径时，更新最有路径的值。往后拓展子节点时，遇到当前代价+剩余城市最小出度和>best时，剪枝。故输出按递减结果排列。

# 6 TSP-动态规划法

## 6.1 问题分析

动态规划法是指，在穷举法搜索子问题的过程中，会有大量的冗余计算，为了降低时间开销，引入备忘录将子问题的最有解存储，便于后续直接查表直接使用结果。

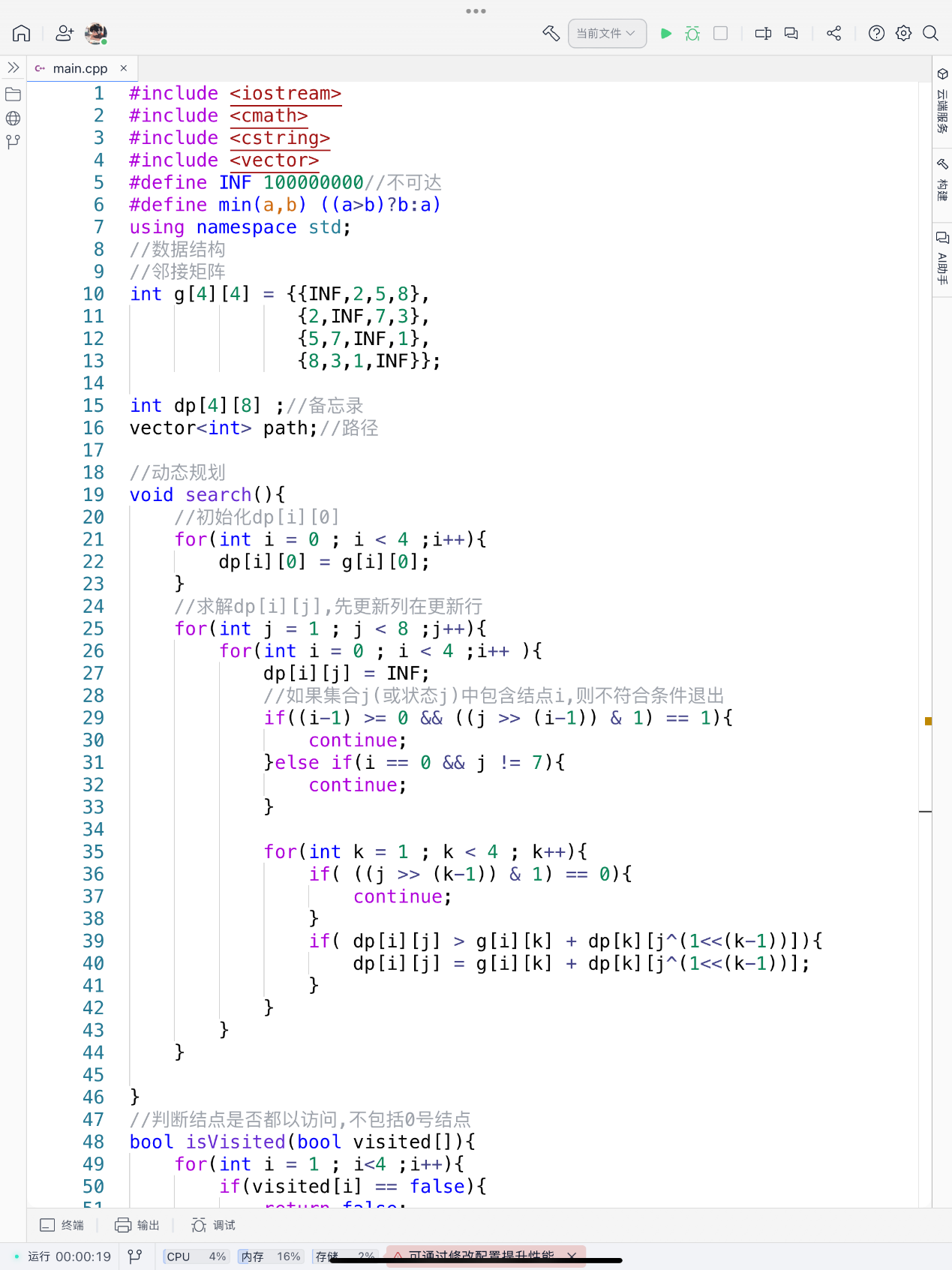
动态规划法求解TSP问题的思路是，存在一个数据结构d{a，{b,c,d}}，求解该问题的公式为：

d{a,{b,c,d}}=min{Cab+d{b,{c,d}}

可解释为，当前问题的最优解，是当前城市到剩余城市的某一个的路程+除被选择城市外的剩余城市的最优路径。

## 6.2 数据结构

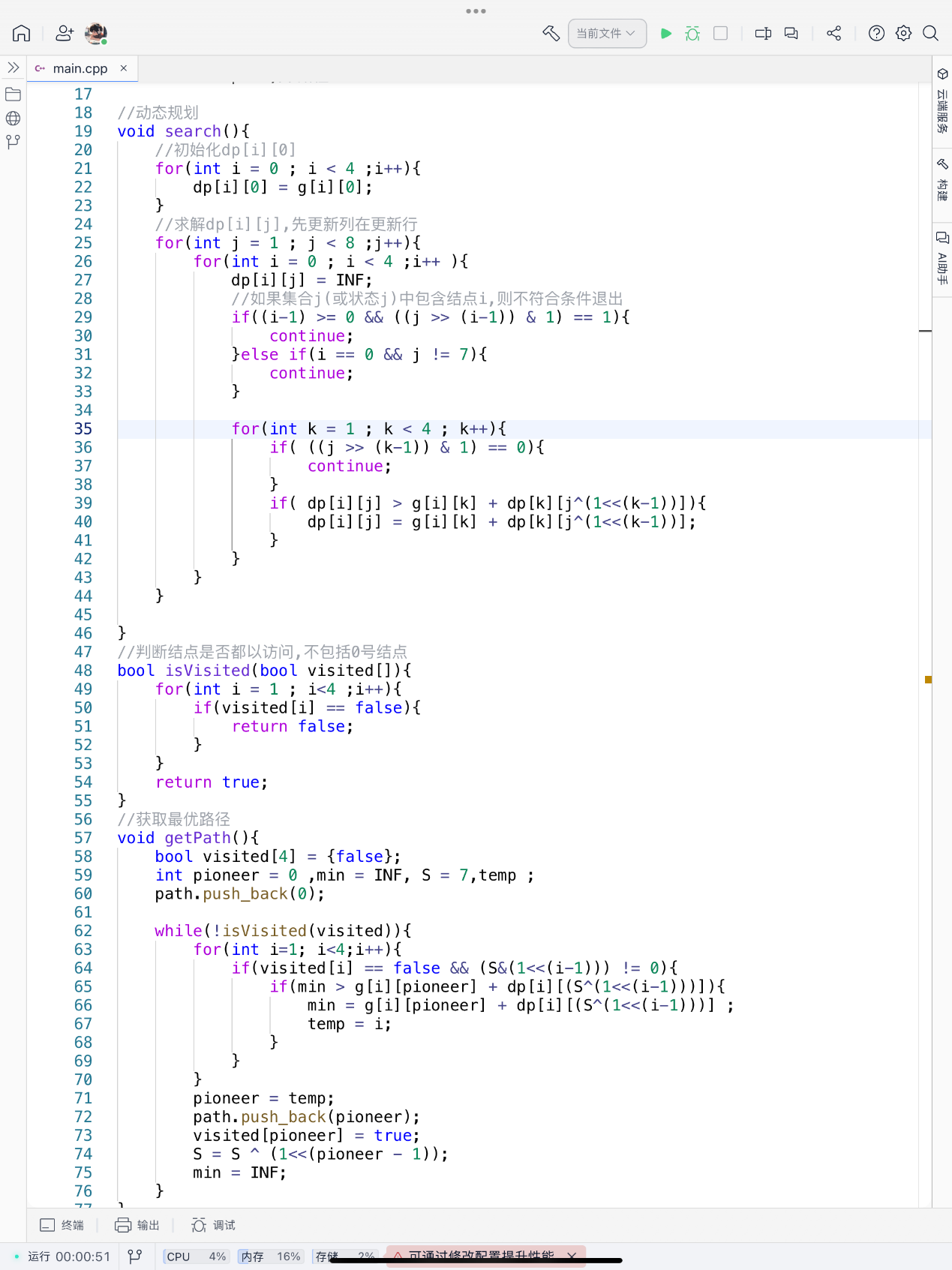
+



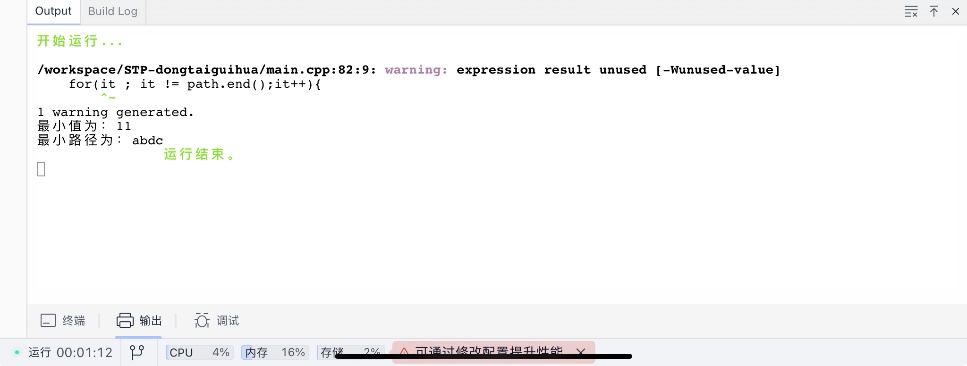
主要的数据结构有：

* 邻接矩阵：他存放的是城市与城市之间的路径长度，元素的下标表示城市
* 备忘录：存放子问题的最优解
* 结果列表：存放已经访问的城市列表

## 6.3 实现函数



## 6.4 输出分析



根据备忘录反向使用递归函数，将的到如上图所示的最优解。

# 7 TSP-分治法

## 7.1 问题分析

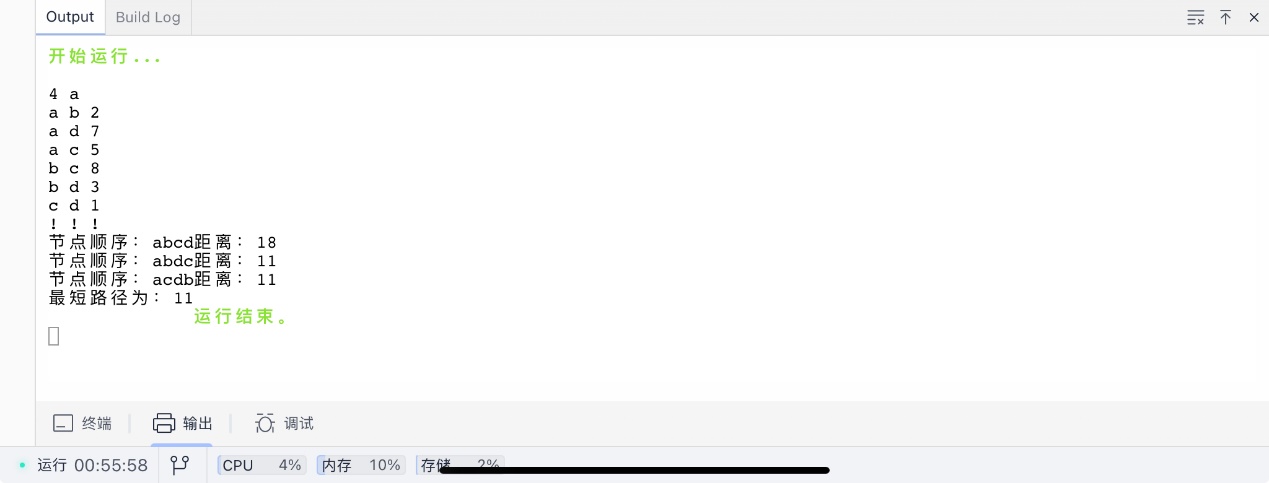
采用分治法的主要难点是如何确定子问题，本题的子问题，如果只考虑路径长短，则很难抽象出子问题。换个思路，如果确定了遍历城市的次序，则确定了路径长度。故而本题可以转化为节点的排列问题（但不是全排列）。利用分治法可以求解集合的排列问题，同时加入剪枝操作，可以减小系统开销的同时，求解最优排列。

## 7.2 实现函数



算法设计思想是：从剩余城市中，选一个城市，放到队列前，排列剩下的城市，更新当前代价。当发现当前代价大于最优代价时，剪枝。

## 7.3 输出分析



输出按递减顺序排列，说明剪枝在求解排列问题时起到了作用。

# 8 感悟

经过本次实验，我成功复现用穷举法、贪心选择、动态规划、回溯法、分支限界法和分治法解决TSP问题。算法这门课，需要我在今后的学习中继续提高。本次实验解决了课堂中，不能及时复现代码的问题。在尝试自己写代码的过程中，逐渐体会了各个经典算法的巧妙之处。同时，亲自编写代码能够发现课堂上领会不到的细节问题，这使我对算法的理解更加深入了。