
第十一届中国软件杯 大学生软件设计大赛 设计与说明文档

队伍名称： 泥头车队

队伍编号： 62012196

参赛题目： A1-高性能民航旅客行程推荐系统

队伍队长： 王凯

联系电话： 18272023062

团队成员： 叶承荣 袁忠升 范启航

目录

一、作品概述	4
(1) 基本功能要求	4
(2) 非功能性要求	5
(3) 实现条件	5
二、软件结构说明	6
(1) 前端结构说明	6
(2) 后端结构说明	6
三、功能模块说明	7
(1) 输入信息模块	7
(2) 低价行程搜索模块	9
(3) 余座更新模块	11
四、数据说明	11
(1) 航班数据	11
(2) 票价数据	12
(3) 票价规则数据	13
(4) 航班余座数据	13
(5) 余座更新数据	14
五、主要类定义	14

六、主要数据结构与高效数据访问技术说明	17
(1) 航班数据结构 FlightSet	18
(2) 票价数据结构 PriceTable	19
(3) 票价规则数据结构 PriceRuleTable	20
(4) 航班余座数据结构 RemainSeatTable	21
七、低价搜索算法	21
(1) 单航段搜索模块	21
(2) 低价行程推荐模块	24
(3) 低价搜索算法	28
八、自研高性能高并发 HTTP 服务器	29
(1) HTTP 请求类 HttpRequest	29
(2) HTTP 响应类 HttpResponse	30
(3) Http 服务器类 HttpServer	31
九、源代码阅读指南	32
(1) 前端源代码阅读指南	32
(2) 后端源代码阅读指南	33
十、附录：API 接口说明	

一、作品概述

（1）基本功能要求

实现 B/S 结构的前后端系统，已部署至公有云平台，浏览器输入域名 hustairline.xyz，即可访问作品。作品满足所有基本功能要求。

1、前端浏览器网页

前端界面效果如图 1-1，实现了搜索请求输入、结果展示以及余座数据更新功能。

9月1日 ~ 2023年8月31日

乘客人数 0 最大结果数 0

代理人 全选 全不选

SHS XIC WUX SHE NNG DNH SZX AOG XUZ CGQ

航段

第 1 段

出发地 目的地 出发日期

重置数据

余座数据更新

低价行程搜索

Q 搜索

图 1-1 前端界面效果

2、低价行程推荐

支持多个（最多 8 个）航班联程、多代理人（后端支持任意数量，目前前端提供 10 个可选项）、多旅客（最多 8 个），支持设置最大结果数。

3、航班余座数据更新

使用外部模拟系统手动触发，提供 5 个数据更新文件选项。



图 1-2 余座更新功能

(2) 非功能性要求

作品满足全部非功能性要求，性能及阻塞部分详见软件测试文档。

- 1、单次搜索过程单线程处理；
- 2、云服务器为 2 核 CPU，搜索服务设置为 2 个线程；并且在搜索并发数为 2 并且计算量足够时，CPU idle 值小于 20%，作品能够充分利用硬件资源。
- 3、搜索过程全程无阻塞，前后端在搜索过程中均无网络传输、文件访问、多线程等操作。
- 4、航班余座更新时，所有搜索服务进程/线程同时生效，且搜索服务不中断不阻塞；
- 5、同一台服务器内所有搜索服务进程/线程共享独一份数据，共享索引。

(3) 实现条件

本软件作品开发基于 linux 平台，使用 Clion 和 github 实现协同开发，后端搜索引擎使用 C++，前端使用 Vue 框架。作品提供了独立的 Web 服务，服务器使用 ubuntu 系统，并且部署到了公有云平台（浏览器访问域名 hustairline.xyz）。无第三方数据库，使用自研数据访问技术，自研高性能高并发 HTTP 服务器。编译器使用 gcc。

二、软件结构说明

（1）前端结构说明

前端使用 Vue 框架编写，具有高效率，实时响应的特点，能够实时对输入数据进行响应。界面美观，实现了所有功能，同时采用渐进式框架，用户体验更佳，便于维护与功能拓展。

前端又分为用户输入、查询结果和随机重置数据三大部分。结构如图 2-1：

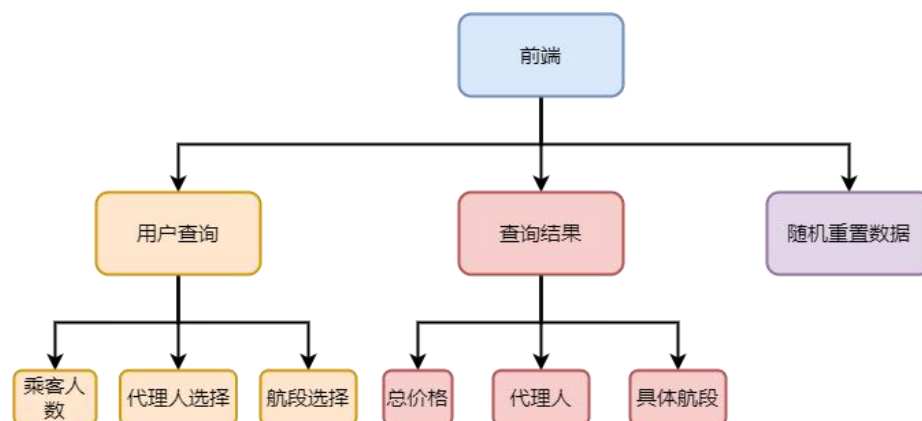


图 2-1 前端结构

（2）后端结构说明

后端部分也分为三大块，分别是数据录入与处理、数据查询和行程推荐。

1、数据录入与处理

将航班数据 flight.txt、票价数据 price.txt、票价规则数据 priceRule.txt 和余座数据 FlightSeats.txt 读入到程序当中，并且建立相应的目录索引，经过一定处理之后生成 SET、PT、PRT、RST 四个数据库。

2、数据查询

SET、PT、PRT、RST 四个数据库均提供了高性能查询接口。输入航班承运人、起始城市、终点城市等信息，能够查询到航班 flight、票价 price、代理人 agency、

附加费率 surcharge、余座 seats 等信息，其中余座数据库 RST 还提供数据更新 update 接口。

3、行程推荐

后端主要功能，支持多个（最多 8 个）航班联程、多代理人、多旅客（最多 8 个）的低价行程推荐，详见低价搜索算法说明部分。

三、功能模块说明

本软件功能模块分成：输入信息模块、低价行程搜索模块、余座更新模块。

三个模块都提供了外部接口，即通过前端界面实现调用，分布如图 3-1。其中低价行程搜索模块依赖于输入信息模块中用户输入的各项信息，余座更新模块可以随时调用，没有依赖关系。



图 3-1 软件功能模块分布

(1) 输入信息模块

该模块功能是设置低价行程搜索功能所需的各项信息，分别为：

- 1.乘客人数：初始值为 1，可随意调整；

2.最大结果数：搜索结果数的上限，取值范围 1~40；

3.代理人：初始为空，可多选。点击指定代理人按钮，蓝色表示选中，白色表示未选；

4.行程：每段行程包括出发地、目的地和出发日期三个信息，初始置为空。用户可通过点击图 2-1 中黑色“+”增加一段行程，还可通过点击黑色“×”删除该段行程；

用户输入全部以选项形式实现，避免不必要的输入判断。

根据赛题标准以及软件设计，低价行程搜索模块对用户输入有一些要求，具体如下：

- 1) 乘客人数支持 1~8，其余会弹出提示“乘客人数错误”或“查询失败”；
- 2) 最大结果数至少为 1；
- 3) 至少选中 1 个代理人，空代理人会提示“代理商不能为空”；
- 4) 至少 1 段行程（不超过 8 段）并且信息需完整，否则会提示“航程数量不能为 0”或者“输入数据错误”；
- 5) 行程的出发日期应随序号递增，相邻的出发日期可以相同，但是不能存在前面的某段行程晚于后面的某段行程。

The screenshot displays a flight search form with the following components:

- 乘客人数 (Passenger Count):** A text input field containing the value "4".
- 最大结果数 (Maximum Results):** A text input field containing the value "8".
- 代理人 (Agent):** A section with two buttons: "全选" (Select All) in green and "全不选" (Deselect All) in red. Below these is a row of airport codes: SHS, XIC, WUX, SHE, NNG, DNH, SZX, AOG, XUZ, and CGQ. The codes SHS, XIC, SHE, NNG, and AOG are highlighted in blue.
- 航段 (Flight Segments):** A section containing three segment input boxes, each with a label on the left and a close button (X) on the right.
 - 第 1 段 (Segment 1):** Includes "出发地" (Origin) with a dropdown menu showing "百色(AEB)", "目的地" (Destination) with a dropdown menu showing "包头(BAV)", and "出发日期" (Departure Date) with a date picker showing "2022-09-01".
 - 第 2 段 (Segment 2):** Includes "出发地" (Origin) with a dropdown menu showing "包头(BAV)", "目的地" (Destination) with a dropdown menu showing "北京首都(PEK)", and "出发日期" (Departure Date) with a date picker showing "2022-09-02".
 - 第 3 段 (Segment 3):** Includes "出发地" (Origin) with a dropdown menu showing "北京首都(PEK)", "目的地" (Destination) with a dropdown menu showing "长沙(CSX)", and "出发日期" (Departure Date) with a date picker showing "2022-09-03".
- Search Button:** An orange button at the bottom center with a magnifying glass icon and the text "搜索" (Search).

图 3-2 输入样例

如图 3-2 输入样例,乘客人数设置为 4,最大结果数为 8,代理人允许为“SHS、XIC、SHE、NNG、SZX、AOG”,有三段行程:

第一段 2022-09-01, 从百色到包头;

第二段 2022-09-02, 从包头到北京首都;

第三段 2022-09-03, 从北京首都到长沙。

注意行程除了时间递增的要求外,没有其他任何要求(当然我们的数据只支持搜索 2022-09-01 到 2023-08-31 之间的航班,详见数据说明部分)。

(2) 低价行程搜索模块

在信息输入满足要求的情况下,点击橙色的“搜索”按钮,即可调用后台的低价行程搜索功能,每次点击都会重新执行。如果查询不到结果会提示“查询失败”,

能够查询到会有搜索结果展示。

以图 3-2 的输入为例，搜索结果如图 3-3。

搜索到 8 条结果			
结果: 1	代理人: SZX001	总价格: 32360	
航班号: MF2308 座位1: 经济舱 座位2: 经济舱 座位3: 经济舱 座位4: 经济舱	2022-09-01 19:30 百色(AEB) → 2022-09-01 20:45 包头(BAV)	剩余座位信息: 头等舱: A 商务舱: 4 经济舱: A	价格: 6080
航班号: CA4782 座位1: 经济舱 座位2: 经济舱 座位3: 经济舱 座位4: 经济舱	2022-09-02 14:15 包头(BAV) → 2022-09-02 16:15 北京首都(PEK)	剩余座位信息: 头等舱: 4 商务舱: A 经济舱: 7	价格: 14960
航班号: HU2734 座位1: 经济舱 座位2: 经济舱 座位3: 经济舱 座位4: 经济舱	2022-09-03 14:15 北京首都(PEK) → 2022-09-03 18:00 长沙(CSX)	剩余座位信息: 头等舱: A 商务舱: A 经济舱: A	价格: 11320
结果: 2	代理人: XIC001	总价格: 32480	
航班号: MF2308 座位1: 经济舱 座位2: 经济舱 座位3: 经济舱 座位4: 经济舱	2022-09-01 19:30 百色(AEB) → 2022-09-01 20:45 包头(BAV)	剩余座位信息: 头等舱: A 商务舱: 4 经济舱: A	价格: 6120
航班号: CA4782 座位1: 经济舱 座位2: 经济舱 座位3: 经济舱 座位4: 经济舱	2022-09-02 14:15 包头(BAV) → 2022-09-02 16:15 北京首都(PEK)	剩余座位信息: 头等舱: 4 商务舱: A 经济舱: 7	价格: 15000
航班号: HU2734 座位1: 经济舱 座位2: 经济舱 座位3: 经济舱 座位4: 经济舱	2022-09-03 14:15 北京首都(PEK) → 2022-09-03 18:00 长沙(CSX)	剩余座位信息: 头等舱: A 商务舱: A 经济舱: A	价格: 11360

图 3-3 部分搜索结果

前端会显示查询的结果总数，查询结果分为总价格、代理人和具体航段三部分，并按照总价格排序。

1、总价格：当前方案的总价格。

2、代理人：当前方案的支持的出票代理人。

3、具体航段：每个方案中包含具体航段的信息，显示航班号、乘客座位表、行程信息（时间、地点）、航班余座数据以及总票价信息。

（3）余座更新模块

点击“重置数据”按钮，即调用余座数据更新功能，前端会提供 5 个更新文件来支持手动更新。执行后主要在航班余座上有较大不同，并且尤其在经济舱余座变化较大时搜索结果会有较大的变化。

四、数据说明

（1）航班数据

生成的 Flight.txt 文件，包含未来一年内国内所有航班信息，文件共 1862933 条航班信息，大小为 85.2MB。

样例：FM;3493;20230530030000;20230530073000;FUG;BHY;

各个部分用分号分开，各个分区依次表示航班承运人（carrier）、航班号 (FlightNo)、起飞时间、到达时间、起始城市三字码和终点城市三字码。

航班承运人共 23 个：CA、CZ、PN、MU、MF、SC、EU、FM、ZH、X2、3Q、XQ、3U、WU、G4、HU、FJ、GJ、JD、HO、9C、GS、G5。

城市三字码共 119 个：AQG、AOG、AVA、AEB、BSD、BAV、BHY、PEK、BFU、CGQ、CGD、CSX、CZX、CTU、CIF、CKG、DLU、DLC、

DNH、 ENH、 FUG、 HMI、 HGH、 HZG、 HFE、 HEK、 HNY、
TXN、 HET、 HUZ、 JMU、 KNC、 JGN、 JIL、 TNA、 JDZ、 JNG、 JNZ、
JIU、 CHW、 JZH、 KHG、 KRY、 KRL、 KMG、 LHW、 LXA、 LYG、
LJG、 LYI、 LHN、 LZH、 LYA、 LUZ、 LZO、 LUM、 NZH、 MIG、 MDG、
KHN、 NAO、 NKG、 NNG、 NTG、 NNY、 NGB、 PZI、 TAO、 IQN、
SHP、 NDG、 JUZ、 SYX、 SHA、 PVG、 SWA、 SHS、 SHE、 SZX、 SJW、
SZV、 TYN、 TSN、 TNH、 TGO、 TEN、 WEF、 WEH、 WNZ、 WUH、
WHU、 HLH、 URC、 WUX、 WUS、 WUZ、 XMN、 XIY、 SIA、 XIC、
XIL、 XNN、 XUZ、 ENY、 YNZ、 YNT、 YBP、 YIH、 YIN、 YIW、 LLF、
DYG、 ZHA、 ZAT、 CGO、 HJJ、 ZUH、 ZYI。

其余数据的航班承运人、城市三字码都为上述所示。

航班信息时间从 2022 年 9 月 1 日至 2023 年 8 月 31 日，后面的小时、分钟、秒的信息按一定规则生成，小时不会超过 24h，分钟是 15 的倍数，秒数为 00，到达时间总大于起飞时间，且不会超过 4 小时。

起始终点城市之间都是有航班的，且往返情况不尽相同，各个城市之间一天的航班个数在 1 到 15 个之间不等，航班承运人随机选取，航班号在 1000 和 9999 之间。

(2) 票价数据

生成 price.txt 文件，包含各个航班承运人在各个城市之间的三种舱位的票价，文件一共 161483 行，大小为 5.21MB。

样例：G5;ZUH;ZYI;F;3250;C;2940;Y;2540;

从左往右依次表示航班承运人、起始城市、终点城市、头等舱票价、公务舱票价、经济舱票价。

经济舱票价在 600 元到 3500 元不等，商务舱在经济舱的基础上多出 30 到 50 元不等，头等舱在经济舱的基础上多出 50 元到 90 元不等。

(3) 票价规则数据

生成 priceRule.txt 文件，包含各个航班承运人在各个城市之间的票价规则数据，文件一共 1937796 行，大小为 135MB。

样例：01174459;ZH;JDZ;TAO;JD;XUZ001,NNG001,AOG001,SHS001;17;

从左往右依次表示规则序列号、航班承运人、起始城市、终点城市、下一个航班承运人、航班代理人、票价起伏比率（surcharge）。

规则序列号从 12345 开始，一共八位数字。

如果起始城市、终止城市为空表示任意城市。

由于文件信息量比较大，本应当为有向图，即起始城市和终点城市顺序不能交换，变成了无向图，即规则和起始、终点城市顺序无关，节省内存。

航班代理人由城市三字码加上 001 构成，为了前端的可视性，代理人信息设计的少一些，只有 SHS001、XIC001、WUX001、SHE001、NNG001、DNH001、SZX001、AOG001、XUZ001、CGQ001 共 10 个航班代理人。

每一条规则的代理人个数在 4 到 9 个之间。

票价起伏比率取值为-1 到 100，-1 表示不出票，10 表示在原有票价基础上浮 10%，最终的票价为原有票价乘以 110%再减去个位数字。

(4) 航班余座数据

生成 flight.txt 的同时生成 FlightSeats.txt，一条数据包含航班承运人、航班号、起始城市、终点城市、起飞时间、抵达时间、头等舱余座、商务舱余座、经济舱余座共 9 项信息。文件共 1862933 条余座信息，数目上与 flight.txt 相等，大小为 95.9MB。

样例：WU;3278;HMI;CZX;20230530164500;20230530203000;8;A;A;

从左往右依次是航班承运人、航班号、起飞城市三字码、目的城市三字码、起飞时间、抵达时间、头等舱余座、商务舱余座、经济舱余座。

三项余座数据使用随机数生成，概率分布为[5,5,5,5,5,5,5,5,5,50]即 0 到 9 概率为 5%，A（10 座以上）概率为 50%。

（5）余座更新数据

数据设计与航班余座数据一致，一条数据包含航班承运人、航班号、起始城市、终点城市、起飞时间、抵达时间、头等舱余座、商务舱余座、经济舱余座共 9 项信息。

共有五个数据更新文件：Update1.txt，Update2.txt，Update3.txt，Update4.txt，Update5.txt。平均 45 万条更新数据，并且在生成时利用随机步长的方式模拟了赛题中对于更新频率的要求（即近两个月的航班更新频繁，往后的航班更新频率低）。三项余座数据同样使用随机数生成。

五、主要类定义

1. Time

- i. 属性：时间类，主要用于航班信息中的起飞和抵达时间
- ii. 成员：包含年、月、日、小时、分钟
- iii. 方法：提供了各种时间相关操作，包括构造函数，求算在该年中的第几天，将字符串转成 Time 型，比较，判断两个时间是否满足航班衔接条件（间隔大于 2 小时）等

2. Flight

- i. 属性：记录航班信息，是 Net 中主要的存储数据类型

-
- ii. 成员：承运人、航班号、起飞和抵达时间、起飞和抵达城市三字码
 - iii. 方法：构造函数，获取各项信息的接口

3. Net

- i. 属性：某一天（日期）中所有航班形成的航班图。
- ii. 成员：存放了城市节点，邻接矩阵和当天的日期。
- iii. 方法：通过城市三字码搜索城市，初始化航班数据，添加航班，单航段搜索等。
- iv. 定义了一个 Mysplit 全局函数，用于数据读入功能中分割字符串。

4. FlightSet

- i. 属性：以天为单位存放每一天的 Net
- ii. 成员：存放了当天日期，未来的所有航班图
- iii. FlightSet 存放了未来一年中每一天的 Net 航班网，以日期间隔天数为索引，将 365 个 Net 存入到一个向量当中，即为航班数据的存储结构
- iv. 方法：初始化，构建航班数据库，低价行程推荐等

5. AnsElement

- i. 属性：Flight 的子类，比 Flight 多了一些附加信息。主要用于低价搜索算法中充当中间层，实现搜索数据传递。同时也用于 FlightAns 类中存储推荐方案中某一航段的信息
- ii. 成员：（附加信息）共有代理人 agc、余座数据（seatF、seatC、seatY）、乘客座位表 seatList、最低票价 price
- iii. 方法：信息获取接口、附加信息设置接口

6. FlightAns

-
- i. 属性：低价行程搜索结果
 - ii. 成员：航班联程信息 `flight`、总票价 `ticketPrice`、共有代理人 `agc`
 - iii. 方法：添加一段航班、判断当前方案是否满足时间衔接条件、判断当前方案中所有航班支持的代理人是否存在交集等

7. FlightRequest

- i. 属性：搜索请求，包含一段行程的各项信息，用于搜索功能
- ii. 成员：包含时间，起止城市，代理人，乘客数目，最大请求数目
- iii. 方法：信息获取接口

8. Price

- i. 属性：票价数据类，是票价数据库 `PT` 存储的主要数据类型
- ii. 成员：存放了承运人、起止城市和 `FCY` 三种舱位的票价
- iii. 方法：构建票价、信息获取接口

9. PriceTable

- i. 属性：存放所有票价信息
- ii. 成员：包含 `Price` 向量，航班承运人出现的顺序字典和每一个承运人的起始城市在 `Price` 向量中位置的索引
- iii. 方法：构建表（同时构建索引），打印索引，找到某一航班的票价

10. PriceRule

- i. 属性：票价规则数据类，是票价规则数据库 `PRT` 存储的主要类型
- ii. 成员：包含序号，承运人，起止城市，下一个承运人，代理人队列，额外百分比
- iii. 方法：构造函数、信息获取接口

11. PriceRuleTable

- i. 属性：存放所有票价规则
- ii. 成员：包含票价规则表，承运人索引和城市索引表
- iii. 方法：构建，打印索引，查找索引，查询代理人，查询 surcharge

11.RemainingSeat

- iv. 属性：余座数据类，表示一个航班的余座，是余座数据表的主要存储类型
- v. 成员：航班承运人，航班号，起飞地，目的地，起飞时间，到达时间，头等舱剩余座位，商务舱剩余座位以及经济舱剩余座位
- vi. 方法：构造函数，数据更新函数，信息获取接口

12. RemainSeatTable

- i. 属性：存放所有航班的余座数据
- ii. 成员：一个存有所有航班余座信息的哈希表
- iii. 方法：读取文件储存航班余座数据，搜索特定航班余座数据，更新所有航班余座数据，判断余座数据表是否为空等

六、主要数据结构与高效数据访问技术说明

后端主要涉及航班数据、票价数据、票价规则数据、航班余座数据四种数据，我们通过定义 C++ 类来设计这四种数据结构，分别为上面提到的 FlightSet、PriceTable、PriceRuleTable、RemainSeatTable，并通过这四种数据结构封装的初始化函数生成 SET、PT、PRT、RST 四个独立的数据库，并提供高性能的数据查询接口，即高效数据访问技术。

(1) 航班数据结构 FlightSet

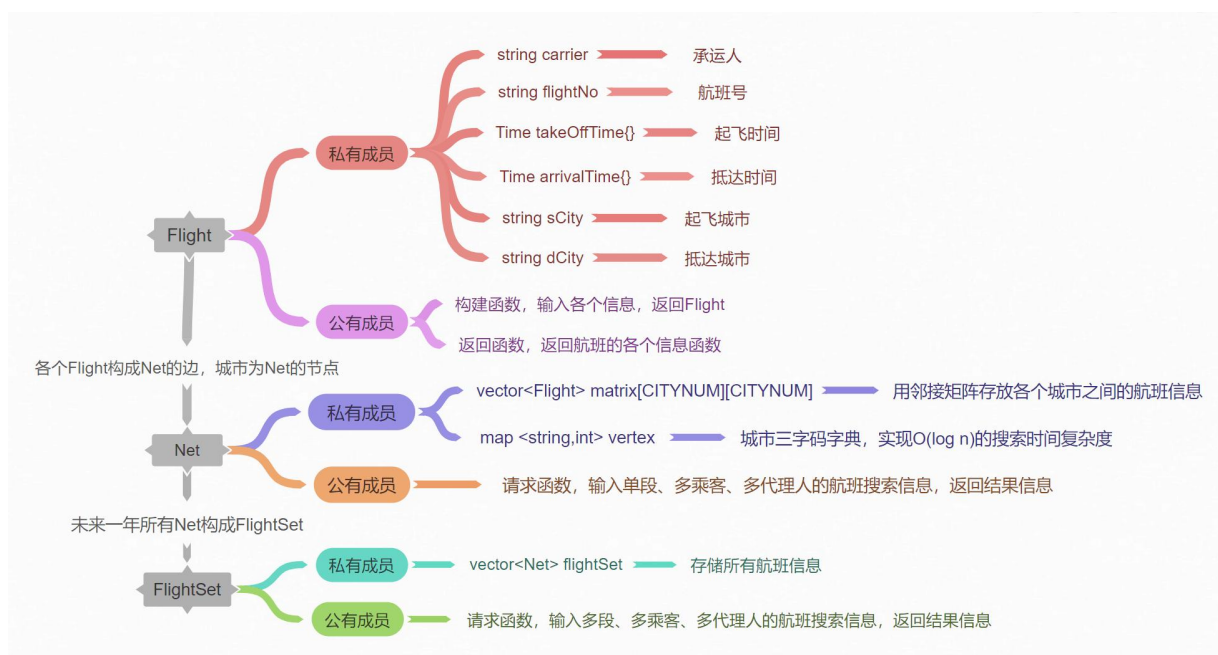


图 6-1 FlightSet 数据结构

1、核心存储结构：

FlightSet 使用数组存放一个个 Net；

Net 用二维数据加邻接矩阵实现存储某一天的所有航班；

2、高效数据访问技术：

FlightSet 以日期为索引定位相应的 Net，时间复杂度 $O(1)$ ；

Net 使用 C++封装的 map 查找城市三字码索引，时间复杂度 $O(\log N)$ ；即通过 $O(\log N)$ 的查找就能拿到当天从 A 到 B 的所有航班，

(2) 票价数据结构 PriceTable



图 6-2 PriceTable 数据结构

1、核心存储结构：

PriceTable 使用一个数据存放 Price 数据；

2、高效数据访问技术：

使用三级目录<carrier、sCity、dCity>（承运人、出发地、目的地）映射数组索引。查找票价时前两级使用 Hash 查找定位，最后一级采用遍历方式，设承运人数目 M，和城市数目 N，时间复杂度视作 $O(M*N)$ ，由于 M，N 规模很小，因此查询速度也非常快。

(3) 票价规则数据结构 PriceRuleTable

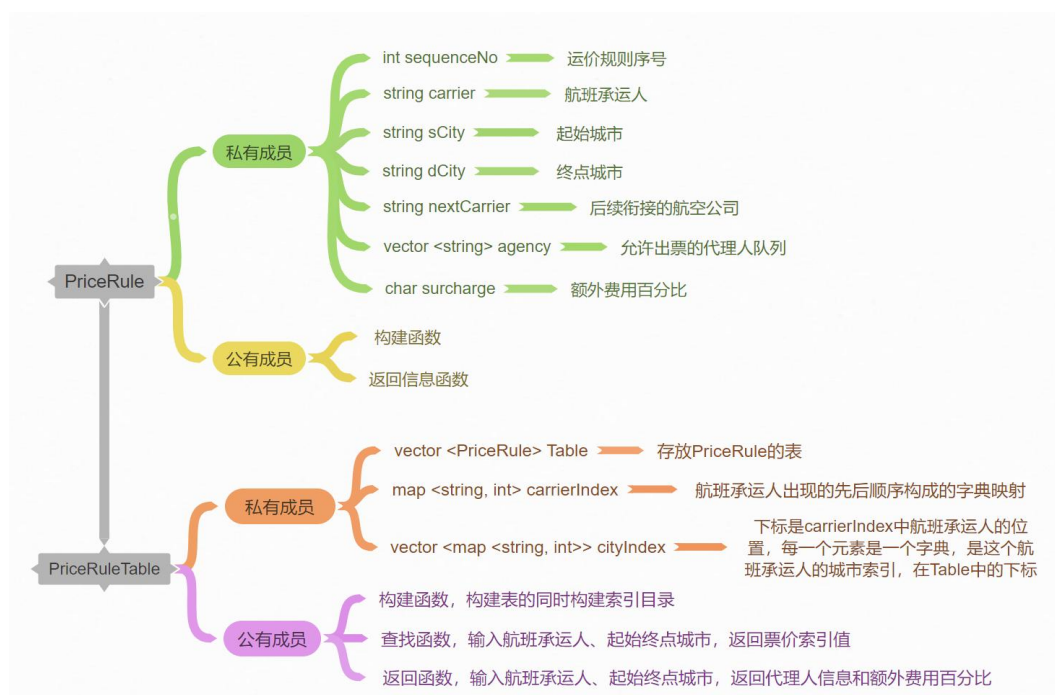


图 6-3 PriceRuleTable 数据结构

对比图 6-2、图 6-3 可以发现 PriceRuleTable 核心存储结构和数据访问方式与 PriceTable 完全相同。故不赘述。

(4) 航班余座数据结构 RemainSeatTable

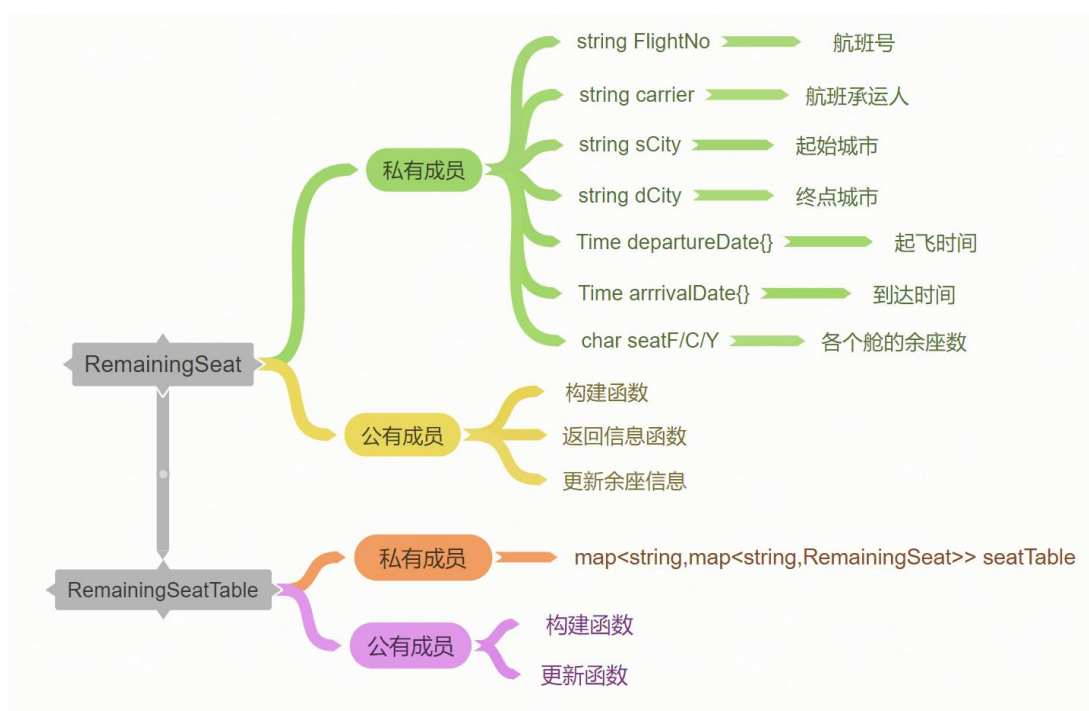


图 6-4 PriceRuleTable 数据结构

1、核心存储结构

使用双重 map 存储 RemainSeat 数据，第一层以时间字符串为索引，第二层以航班号 flightNo 为索引（某一时刻起飞、某一航班号对应的航班是唯一的）。

2、高效数据访问技术

双重 map，余座查询时间复杂度 $O(\log M * \log N)$ 。

七、低价搜索算法

本软件核心的低价搜索算法主要由两大模块构成：单航段搜索、低价行程推荐。分别对应 Net 和 FlightSet 中的 request 函数。

(1) 单航段搜索模块

1) 算法概述：

对于一段行程信息 req，搜索当天所有满足条件的航班，并对每一个满足条件的航班 f 记录下搜索过程中得到的重要数据：满足条件的共有代理人 common_agc、航班余座 f_seats、以最低票价为分配标准的乘客舱位表 seatList、相应的最低票价 ticketPrice。

该模块是低价搜索算法的基础模块，主要实现航班信息查询。

2) 算法详细流程如图 7-1、7-2、7-3

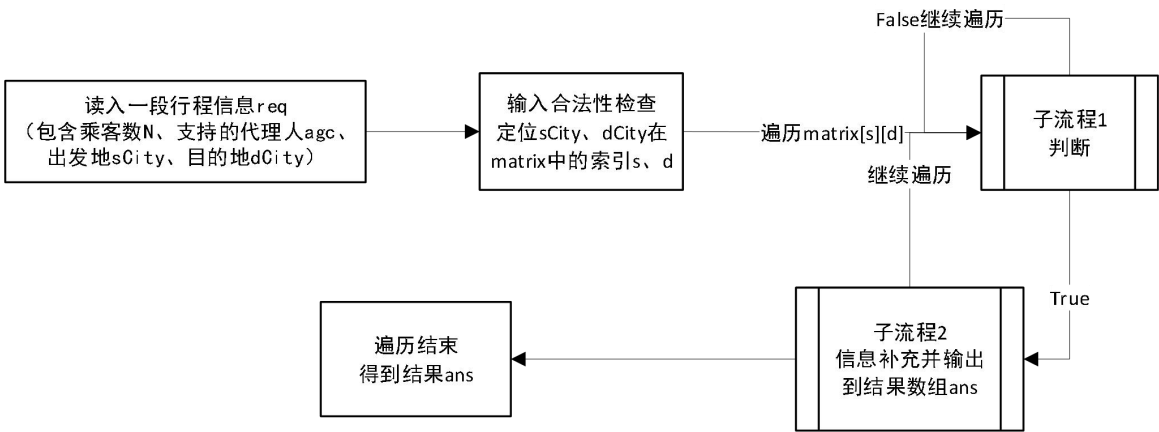


图 7-1 主流程

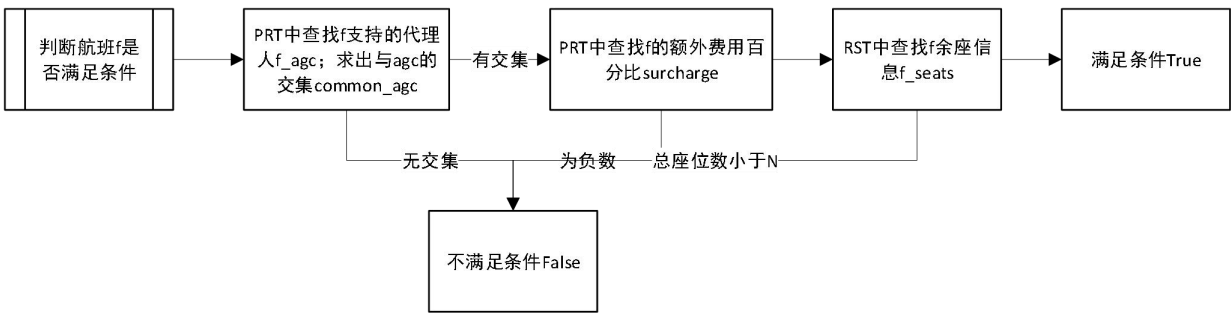


图 7-2 子流程 1

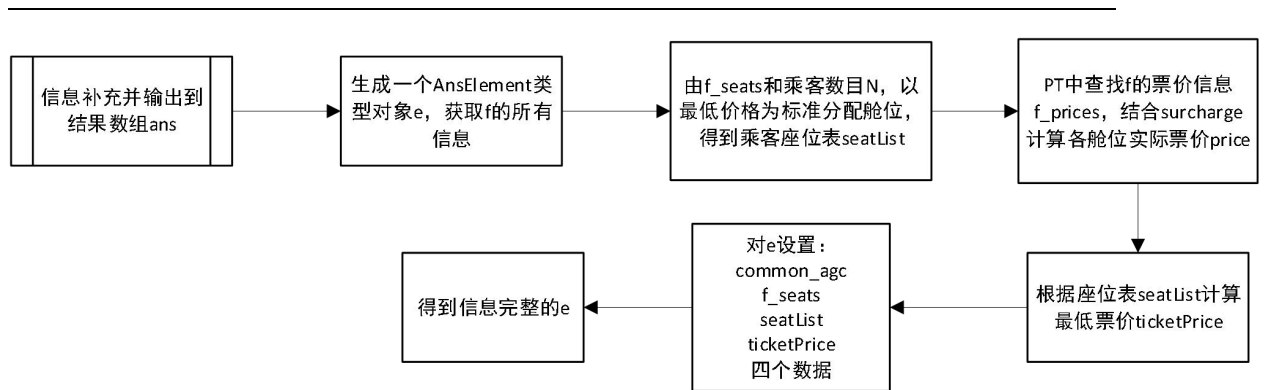


图 7-3 子流程 2

3) 流程图说明:

1、主流程图:

req 是类 FlightRequest 的一个对象, 存放了一段行程的各项信息;

matrix 是 flight 类型的二维数组加邻接矩阵结构, 是 Net 类的主要存储结构, 存放了一天的所有航班。matrix 索引使用 map, 可以实现 O(1)查找。

数组 ans 是 AnsElement 类型的对象数组;

AnsElement 是 Flight 的子类, 自定义了 agc(满足要求的代理人数组)、seats(三个舱位的余座数据)、passenger_seatList(乘客座位表)、price(票价)四个私有成员。AnsElement 类在低价搜索算法中实现了**中间层**的作用, 记录了在搜索过程中得到的、后续排序中必须的信息, 实现了两个模块的高效衔接。

2、子流程图 1:

f 是类 Flight 的一个对象;

PRT: 票价规则数据库; PT: 票价数据库; RST: 余座数据库; 三个数据表库均可实现高性能查询。

3、子流程图 2:

以最低价为标准是指优先经济舱、不够再选商务舱、头等舱;

实际票价计算公式:

$$a=f_price*(100+surcharge)/100; \text{ price}=a-a\%10;$$

最后一步设置四个附加数据, AnsElement 类中都封装了相应的接口。

3) 算法优势与创新特色:

结合 PRT、PT、RST 三个数据库的高性能搜索, 所有底层数据查询均在该模块完成;

使用 AnsElement 类作为中间层记录搜索过程中得到的重要信息(满足条件的代理人 common_agc、航班余座 f_seats、最低价的乘客舱位表 seatList、相应的最低票价 price)。实现了从单航段搜索到低价行程推荐的高效衔接。

(2) 低价行程推荐模块

1) 算法概述:

如果用户输入只有一段行程, 那么只要对单航段搜索得到的结果 tmp_flight[0]按照票价 ticketPrice 进行排序, 时间复杂度 $O(N\log N)$;

如果用户输入多段行程, 对于 tmp_flight 二维数组, 需要解决航班的组合问题, 同时拿到总票价 Price 最低的 k 个结果(至多)。此时核心问题就变成了以下算法难题:

给定 M 个长度不等 (N_i) 的序列, 从每个序列中各取一个数, 可以构成 $\prod N_i$ ($N_1*N_2*...*N_m$) 个和。求这些和中前 k 小的值。

对于以上算法难题, 首先需要对 M 个序列进行排序, 排序过程不可避免。因此现有解法的时间复杂度上限是 $O(MN\log N)$ 。

后续求前 k 小和, 常见解法是堆模拟搜索, 根据数学归纳法, 结合小根堆先求出前 2 个序列中任取一个数相加构成的前 k 小和, 把这 k 个和作为一

个序列，再与第 3 个序列求新的前 k 小和，依次类推，最终得到 M 个序列任取一个数相加构成的前 k 小和。时间复杂度为 $O(Mk \log k)$ 。

作品使用一个优化算法来提升搜索效率，能实现 $O(k \log k)$ 的时间复杂度，该算法同样是堆模拟搜索算法，它与常见解法最大的不同在于直接处理 M 个序列，而不是 2 个 2 个的拆分下来求解。算法核心在于它的状态扩展方案。

2) 优化算法：

优化算法同样是堆模拟搜索的应用，为便于描述，下文中将第 x 个序列中的第 y 个数记作 (x,y) 。

首先，最优解一定是将每个序列中最小的数（即从小到大排序后的第一个元素，下文默认所有序列已经进行排序）加起来。因此该方案即为堆中的初始状态。

下面思考状态的设计。考虑这样一种状态方案：用四元组 $(val, x, y, p=0/1)$ 代表一种状态，其中 val 代表当前状态中 M 个数的和， x 代表我们人为地从前 x 个序列中选出了一些数字，而第 $x+1$ 到第 M 个序列中每个序列被选取的都是第一个数字。例如初始状态记为 $(min, 1, 1, 0)$ 。

此时，状态中值为 0 或 1 的 p 看起来有些多余。事实上，该变量的值与我们采取的状态扩展策略有关。对于某个状态，我们有三种扩展方向：

1、当前状态中 $y < N_x$ (N_x 为序列 x 的长度)，在当前方案中去掉数字 (x,y) 并替换为 $(x,y+1)$ ，得到一种新的状态 $(NewVal, x, y+1, 0)$ 。

2、当前状态中 $x < M$ 且 $N(x+1) \geq 2$ ($N(x+1)$ 为序列 $x+1$ 的长度)，在当前方案中去掉数字 $(x+1, 1)$ 并替换为 $(x+1, 2)$ ，得到一种新的状态 $(NewVal, x+1, 2, 1)$ 。

3、当前状态中 $y=2$ 且 $x < M$ 且由途径 2 或 3 转移得到，在当前方案中去掉 (x,y) ，替换为 $(x, 1)$ ，并选取 $(x+1, 2)$ ，得到一种新的状态 $(NewVal, x+1, 2, 1)$ 。

由以上方案，我们不难看出， p 代表的就是当前状态是否由途径 2 或 3 转移得到。该变量的设置是为了进行下一步的转移。

堆模拟搜索的扩展过程中，答案的转移必须遵循“不重不漏”的原则。通过以上状态转移方式，我们可以避免状态的重复扩展，做到了“不重”，但是没有做到“不漏”。

考虑这样三个序列：10 100 、 15 50、 20 30

如果直接按照上述方案进行转移，我们会漏下一种方案，即选取 10、15 与 30。造成这一点的原因是，我们首先扩展出了选取 10、50 与 20 的状态。为了解决这个问题，需要进行一个操作——在进行第一次扩展前，将每个序列以“次小值减去最小值”作为关键字从小到大进行排序。这一步操作的目的是为了保证我们在进行扩展时先扩展出能令新方案的元素和较当前方案增量较小的方案。

剩下的工作已经很明显了——设置初始状态为(min,1,1,0)，我们每次取出堆顶状态进行扩展，重复 k 次即可。

3) 算法详细流程如图 7-4:

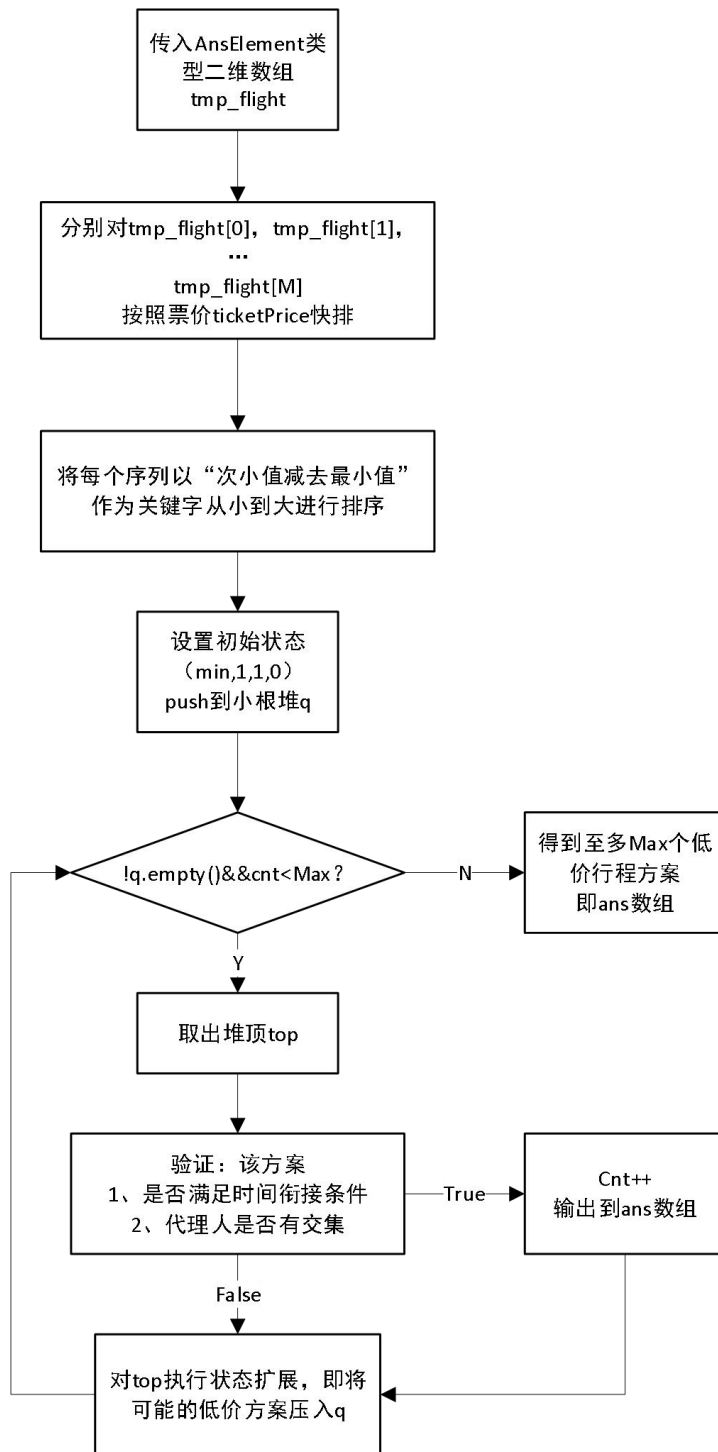


图 7-4 低价行程推荐流程图

4) 流程图说明:

tmp_flight 二维数组由调用单航段搜索功能得到;

状态量 (val,x,y,p) 由自定义结构体实现, 具体代码中还设置了 pos 等其

他辅助信息；

状态扩展方案参照 2) 最优算法中的描述；

ans 数组是 FlightAns 类型的对象数组，一个 FlightAns 对象存储了一个低价行程推荐方案的各项信息，其中还封装了检验时间衔接、求代理人交集等功能。

根据赛题要求，航班的时间衔接条件是抵达时间与下一个航班起飞时间间隔不少于 2 小时。

5) 算法优势与创新特色：

利用堆模拟搜索优化算法提高搜索效率。

(3) 低价搜索算法

1) 算法框架：

传入多段行程数据 (req 数组) -> 输入合法性检验 -> 对 req 每个元素调用单航段搜索功能 -> 得到 AnsElement 类型的二维数组 tmp_flight -> 调用低价行程推荐功能 -> 输出低价行程推荐结果。

2) 算法优势与创新特色：

时间复杂度 $O(MN\log N)$ ，性能强，结合高效数据访问，使用中间层实现信息的高效传递，利用堆模拟搜索优化算法提高搜索效率。

八、自研高性能高并发 HTTP 服务器

本团队自研高性能高并发 HTTP 服务器 `httplib`。主体部分为一个头文件 `httplib.h`，引入头文件即可使用，命名空间为 `httplib`。提供接口，可由用户自主设置路径及其匹配的响应函数。

使用自研 C++ 线程池 `ThreadPool.h` 实现高并发，提供接口，可由用户自主设置线程池中线程个数。

`httplib` 主要由 3 个类组成: `HttpRequest`、`HttpResponse`、`HttpServer`。

(1) HTTP 请求类 `HttpRequest`

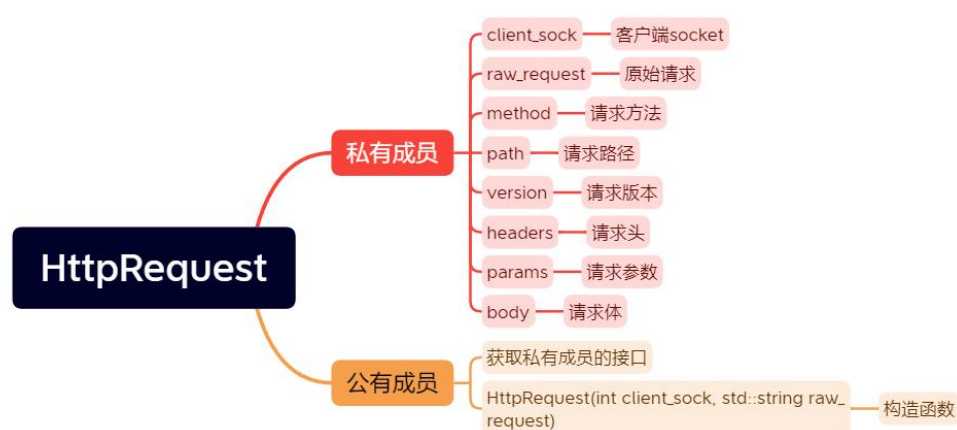


图 8-1 `HttpRequest` 数据结构

1) 功能:

用于解析存储 `Http` 请求。

2) 核心函数:

构造函数 `HttpRequest(int client_sock, std::string raw_request)`

传入 `client_sock` 和接收到的原始请求数据，对原始请求数据进行如下处理得到 `HttpRequest` 请求类。

1. 将原始请求数据按行分割。

2. 首行为状态行，将状态行按空格分割得到请求方法、请求路径、请求参数和请求版本

3. 第二行至/r/n 行为头部行，得到请求头信息

4. /r/n/r/n 后为请求体。

(2) HTTP 响应类 HttpResponse

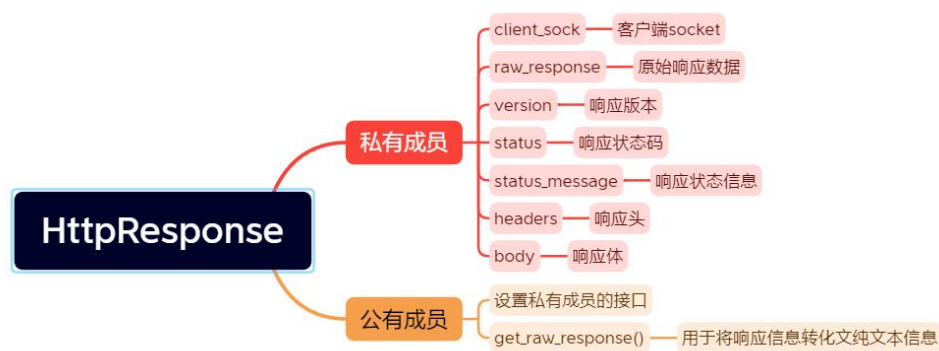


图 8-2 HttpResponse 数据结构

1) 功能：用于存储并构造响 HTTP 响应

2) 核心函数：

`get_raw_response()`

将 `HttpResponse` 类安装 HTTP 规范转化为可用于传输的纯文本信息。

1. 首行为 `version + " " + std::to_string(status_code) + " " + status_message`

2. 后面每行为响应头

3. 隔一个空行后为响应体。

(3) Http 服务器类 HttpServer

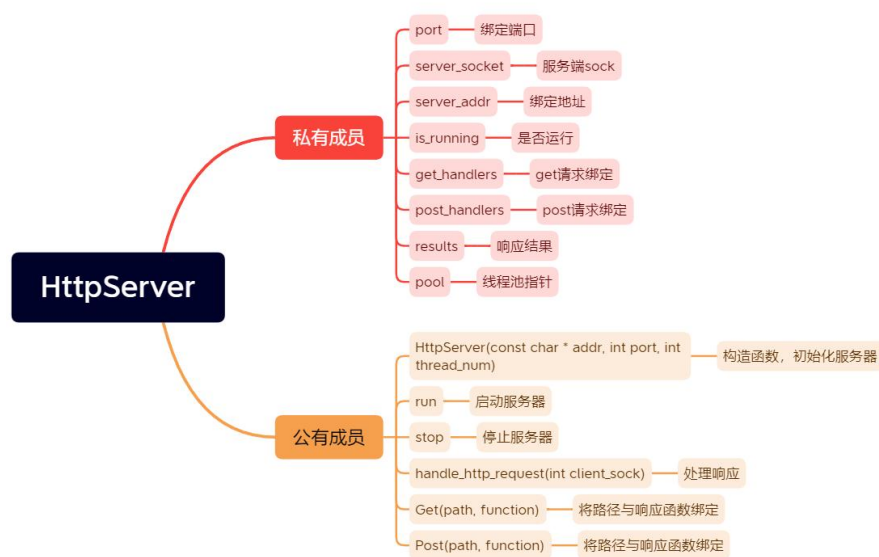


图 8-3 HttpServer 数据结构

1) 功能：用于监听指定地址及端口，开启线程池，对 http 请求进行响应。

2) 核心函数：

`HttpServer(const char *addr, int port, int thread_num) :`

初始化 Http 服务器，创建服务端 socket，将 socket 于地址进行绑定，监听 socket，同时初始化线程池

`run()`:

当 `is_running` 为 `true` 时，循环接收来自客户端的请求，等待客户端建立连接后，将客户端 socket 交给线程池进行响应

`handle_http_request(int client_sock):`

1. 创建缓存区，接收客户端请求
2. 初始化 Http 请求和 Http 响应

3. 根据请求的方法与路径匹配用户设置的响应函数并调用。
4. 将 Http 响应返回客户端。

九、源代码阅读指南

作品源代码主要分成前端源代码和后端源代码，分别对应文件夹 Travel_Recommend_front 和 Travel_Recommend_system，如图 9-1。


 Travel_Recommend_front	2022/7/4 9:16	文件夹
 Travel_Recommend_system	2022/7/4 9:16	文件夹

图 9-1 源代码文件夹

(1) 前端源代码阅读指南

源代码目录：

Travel_Recommend_front		
	.gitignore	// git 配置文件
	index.html	// index 页面
	package.json	// 包管理
	README.md	// README
	vite.config.js	// vite 配置
	yarn.lock	// yarn 配置
	└─.vscode	// 编辑器配置
	extensions.json	
	└─public	// 公共资源
	agents.json	// 代理人目录
	beian.webp	// 备案图标
	cityOptions.json	// 机场选项
	flight.svg	// 网页图标
	updateFiles.json	// 更新文件信息

└─src	// 源代码
App.vue	// app
main.js	// 主 js
└─assets	// 资源目录
└─css	// CSS 资源
global.css	
└─icons	// 图标
arrow-right.svg	
exchangeIcon.svg	
loading.gif	
plus.svg	
remove.svg	
search.svg	
team.svg	
update.svg	
└─components	// 组件
addSeg.vue	// 添加航段组件
Home.vue	// 主页面
loading.vue	// 加载页面
result.vue	// 结果组件
SearchAgent.vue	// 搜索代理人
SearchForm.vue	// 搜索页面
SearchNumber.vue	// 搜索人数、最大结果数
SearchSeg.vue	// 搜索航段
Update.vue	// 更新页面
└─router	
index.js	// 路由配置

(2) 后端源代码阅读指南

源代码目录：主体是定义的 C++ 类

Travel_Recommend_system

AnsElement.cpp

AnsElement.h	
CMakeLists.txt	
Flight.cpp	
Flight.h	
FlightAns.cpp	
FlightAns.h	
FlightRequest.cpp	
FlightRequest.h	
FlightSet.cpp	
FlightSet.h	//SET 数据库、低价行程推荐功能 request
FluctuationTable.cpp	
FluctuationTable.h	//票价波动表（不同代理人票价不同）
httplib.h	//自研 HTTP 服务器
main.cpp	//主函数：系统初始化、服务器启动
Net.cpp	
Net.h	//单航段搜索功能 request
Price.cpp	
Price.h	
PriceRule.cpp	
PriceRule.h	
PriceRuleTable.cpp	
PriceRuleTable.h	//PRT 数据库
PriceTable.cpp	
PriceTable.h	//PT 数据库
RemainingSeat.cpp	
RemainingSeat.h	
RemainSeatTable.cpp	
RemainSeatTable.h	//RST 数据库
ThreadPool.h	//线程池
Time.cpp	
Time.h	

接口列表

接口	说明
/api/query	查询
/api/update	用户登录接口

错误码列表

错误码	说明
200	正确
404	错误

接口详情

- 查询
 - 接口地址： /api/query
 - 返回格式： json
 - 请求方式： Post
 - 请求示例： <https://api.hustairline.xyz/api/query>
 - 接口备注： 通过查询条件， 请求获得航班方案。
 - 请求参数说明：

名称	类型	必填	说明
N	int	true	旅客总数
agency	list	true	允许的代理人
M	int	true	航段数
date	list	true	起飞日期
SCity	list	true	起飞城市
dCity	list	true	抵达城市
Max	int	true	最大结果数

- 请求示例 body

```
{
  "N": 3,
  "M": 2,
  "Max": 20,
  "agency": ["SHS001", "XIC001", "WUX001", "SHE001", "NNG001", "DNH001",
    "SZX001", "AOG001", "XUZ001", "CGQ001"],
  "date": ["20220921000000", "20220922000000"],
  "sCity": ["AOG", "AOG"],
  "dCity": ["BSD", "BHY"]
}
```

- 返回参数说明：

名称	类型	说明
meta	object	状态信息
msg	string	提示信息
status	int	状态码
data	object	具体数据
ansNum	int	方案数
ans	object	方案
agc	list	代理人
flight	object	具体航段
ticketPrice	int	总价格

- 随机重置数据

- 接口地址: /api/update
- 返回格式: json
- 请求方式: Get
- 请求示例: <https://api.hustairline.xyz/api/update?fileName=Update1.txt>
- 接口备注: 根据文件重置余座数据。
- 请求参数: fileName 文件名
- 返回参数说明:

名称	类型	说明
msg	string	提示信息
status	int	状态码

- 请求示例: <https://api.hustairline.xyz/api/update?fileName=Update1.txt>
- JSON返回示例:

```
{  
  "msg" : "update succeed",  
  "status" : 200  
}
```
