能力扩展类样题

数据说明

competition_training_dataset.csv（P8）
competition_testing（P8）
competition_training_dataset（P8）
model.py（P8）
demonstrated_answer_format.csv（P9）
input.md（P15）
test_script.py（P16）

以上涉及到的数据文件可在练习系统中查看，练习系统将在报名截止后开启，届时请关注 CCF 及 CACC 官网通知。

题目一

赛题名称： *Merkle Tree*
时间限制： 1.0 *s*
内存限制： 64.0 *MB*

1.1 题目描述

Merkle Tree 是密码学中一种非常重要的数据结构。它可以高效、安全的验证大量数据的完整性。本题需要实现二叉 *Merkle Tree* ，如图 1.1（来源：Wikipedia）。

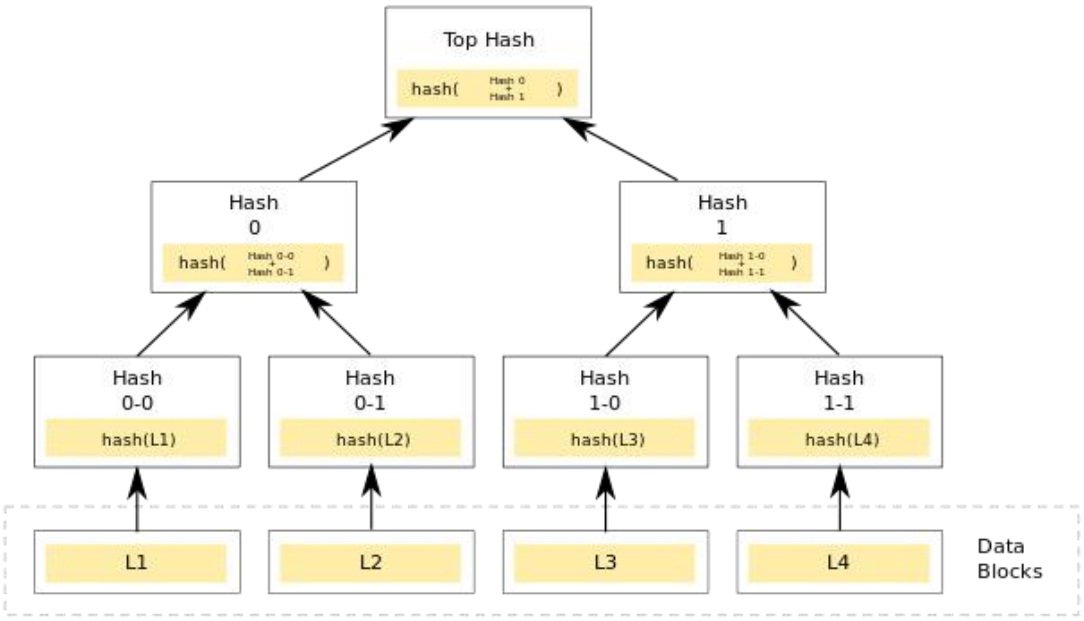

图 1.1

给定输入数据后，$Merkle\ Tree$ 首先使用哈希函数将数据映射到固定长度字符串，其满足以下性质：

1. 给定输入字符串，可以很快地计算输出字符串；
2. 给定输出字符串，很难找出对应的输入字符串；
3. 很难找到两个不同的输入字符串对应相同的输出字符串，即通常所说的哈希碰撞。

选手可直接调用$SHA256$ 哈希函数，其使用方式如下：

Python：

```python
import hashlib
def SHA256(input):
    return hashlib.sha256(input.encode('utf-8')).hexdigest()
```

Java：

```java
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class SHA256{
    public static String SHA256(String input) {
        try {
            MessageDigest digest =
MessageDigest.getInstance("SHA-256");
            byte[] hashBytes =
digest.digest(input.getBytes(StandardCharsets.UTF_8));

            StringBuilder hexString = new StringBuilder();
            for (byte b : hashBytes) {
                String hex = Integer.toHexString(0xff & b);
                if (hex.length() == 1) {
                    hexString.append('0');
                }
                hexString.append(hex);
            }

            return hexString.toString();
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
}
```

其中输入长度不限，输出长度为 32 字节。

Merkle Tree 每个叶节点存储一个数据块的哈希值，按照从左到右的顺序依次存储。

在本题中选手需要构造**平衡二叉 *Merkle Tree***：每个非根节点都有兄弟节点；且对于每个非叶节点，其左子树的叶节点个数大于等于右子树的叶节点个数，但最多只相差 1；

每个非叶节点存储两个子节点的存储内容拼接后的字符串的哈希值，拼接时左子节点的内容在前，右子节点的内容在后。

例如，对于只有三个数据块的输入[‘aaa’，‘bbb’，‘ccc’] 构造 $Merkle\ Tree$，则对应的哈希值分别为：

$Hash00 = SHA256(‘aaa’) =$

$9834876dcfb05cb167a5c24953eba58c4ac89b1adf57f28f2f9d09af107ee8f0$

$Hash01 = SHA256(‘bbb’) =$

$3e744b9dc39389baf0c5a0660589b8402f3dbb49b89b3e75f2c9355852a3c677$

$Hash0 = SHA256($

‘$9834876dcfb05cb167a5c24953eba58c4ac89b1adf57f28f2f9d09af107ee8f03e744b9dc39389baf0c5a0660589b8402f3dbb49b89b3e75f2c9355852a3c677’) =

$7607b2809ae92fffc220deb8af1e4c2878180c29e9f861d79efb0f8bb9961548$

$Hash1 = SHA256(‘ccc’) =$

$64daa44ad493ff28a96effab6e77f1732a3d97d83241581b37dbd70a7a4900fe$

$root = SHA256($

‘$7607b2809ae92fffc220deb8af1e4c2878180c29e9f861d79efb0f8bb996154864daa44ad493ff28a96effab6e77f1732a3d97d83241581b37dbd70a7a4900fe$’) =

$985f0137b0de9241e51dbbe5285d8d717ccc464f65d4213d40dea34c5db2d061$

你需要实现以下功能：

1. 树构造($tree\text{-}construction$)：给定任意数量的数据块，构造$Merkle\ Tree$并返回根节点的哈希值。
2. 数据证明($data\text{-}proof$)：在构造完 $Merkle\ Tree$ 后，输入任一原始数据块，输出其对应路径和路径上每个节点的兄弟节点的哈希值，哈希值之间用-相隔。接上例，当输入为‘bbb’时，输出的证明为$01, Hash1\text{-}Hash00$。如数据不在原始文件中，输出 nil 。
3. 数据获取($data\text{-}fetch$)：构造完 $Merkle\ Tree$ 后，输入一个路径，输出叶节点对应的数据块和功能 2 中的哈希序列作为证明。
4. 数据检验($data\text{-}verification$)：假设你从可信第三方获取了某个数据集的$Merkle\ Tree$ 根节点的哈希值 $root$，接下来从某个不可信的数据源获取了一个数据块 L，声称的存储路径和功能 2 中对应的哈希序列。验证该数据块是否确实在 $Merkle\ Tree$ 中，验证方式为：从叶节点开始沿路

径回溯，对每个节点计算其对应的哈希值，最终计算出根节点哈希值 $root'$，验证 $root' = root$。验证通过返回 $True$，不通过返回 $False$。根据$Merkle\ Tree$ 的构造方法可知：

(1) L 对应的叶节点存储的是数据块本身的哈希值，

(2) 每个非叶节点所需的两个子节点的哈希值之一通过上一步计算得到，另一个从哈希序列中获取，两个值的拼接方式为 0 节点在前，1 节点在后。

1.2 输入格式

从标准输入读入数据。

测试第一行给出数据块数量 n，以及功能测试数量 m。

之后 n 行，提供每个数据块的内容。

之后 m 行，每行给出需要完成的功能及所需输入，包括：

数据证明：该行形式为 $proof$ 数据块内容

数据获取：该行形式为 $fetch$ 路径二进制表述

数据检验：该行形式为 $verify$ 根节点哈希 数据块内容 存储路径 哈希序列（其中的数据项之间以空格间隔）。

1.3 输出格式

输出到标准输出。对每组测试数据的每个功能测试，输出对应内容。

1.4 样例输入 1

```
3 3
aaa
bbb
ccc
construct
proof aaa
fetch 01
```

1.5 样例输出 1

```
985f0137b0de9241e51dbbe5285d8d717ccc464f65d4213d40dea34c5db2d061
00 64daa44ad493ff28a96effab6e77f1732a3d97d83241581b37dbd70a7a4900fe-3
e744b9dc39389baf0c5a0660589b8402f3dbb49b89b3e75f2c9355852a3c677
bbb
 64daa44ad493ff28a96effab6e77f1732a3d97d83241581b37dbd70a7a4900fe-983
4876dcfb05cb167a5c24953eba58c4ac89b1adf57f28f2f9d09af107ee8f0
```

1.6 样例输入 2

```
8 5
```

```
aaa
bbb
ccc
ddd
efgab
xyzmn
000hyz
abce
proof efgab
proof eee
fetch 011
verify b54361df660d316d1c900617717d2fe97e88687e13187bd3c9f4589319ab5b
d0 abce  111 6614de6f1515fd0a053162f0a66484b8c63391c9de566165051c2b10
6683cfd7-f6b48b46298212898181c7636801abdc0cc9093459e1561f665a06915514
4196-2d0a0363c970a15c3f06806b6bb122543a14aae1141a5246b1fed4e3dff87e6d
verify b54361df660d316d1c900617717d2fe97e88687e13187bd3c9f4589319ab5b
d0 fffff 111 6614de6f1515fd0a053162f0a66484b8c63391c9de566165051c2b10
6683cfd7-f6b48b46298212898181c7636801abdc0cc9093459e1561f665a06915514
4196-2d0a0363c970a15c3f06806b6bb122543a14aae1141a5246b1fed4e3dff87e6d
```

1.7 样例输出 2

```
100 6614de6f1515fd0a053162f0a66484b8c63391c9de566165051c2b106683cfd7-
1a925d250af5db9589945e758137cdf8b637aa1e80879e8997bb767bae2d7a5c-5c1b
561c0b9f2120a93aa2d7571358f89d8f8496fb64ccc168c658e149d9e550
nil
ddd f00e59efbf875d7e32ad52fdd9e938ff99beacac67da5f57df7503d1a4c9b516-
7607b2809ae92fffc220deb8af1e4c2878180c29e9f861d79efb0f8bb9961548-64da
a44ad493ff28a96effab6e77f1732a3d97d83241581b37dbd70a7a4900fe
True
False
```

1.8 测试数据规模

共 20 组测试，前 4 组数据量为 5~16，功能测试 1-5 次；4-8 组数据量 16-100，功能测试 1-10 次；8-12 组数据量 100-500，功能测试 1-15 次；12-16 组数据量 500-1000，功能测试 1-20 次；最后 4 组数据量为 1000-2000，功能测试 1-25 次。

1.9 评分标准

满分 200 分：每组测试 10 分，所有功能测试均通过计 10 分，否则计 0 分。

题目二

赛题名称：揭示区块链账户身份

2.1 题目背景

区块链技术通过建立一个安全、去中心化和透明的交易记录框架，革新了各种行业。在区块链网络中，众多账户互动以验证和记录交易，这些交易可能代表了不同的实体和服务之间的关联，也可能表明一些恶意行为，如诈骗。通过交易，准确识别这些账户对于深入理解区块链动态、增强安全性和减少欺诈至关重要。

2.2 题目描述

本竞赛挑战要求参赛者通过利用交易数据和一部分先前标记的账户，预测特定区块链账户的角色。这一任务对于推进区块链理解、增强安全协议和改进区块链框架内的数据分析算法至关重要。

2.3 数据描述

本次竞赛的数据集包括：

交易特征：这是一个处理过的交易特征集，可在‘*competition_training_dataset.csv*’中找到，这些数据来自于已标记账户的交易数据。该数据集旨在促进分析和模型训练。相关特征的详细描述在**附录I** 中提供。

2.4 任务解析

参赛者需要开发预测模型，确定‘*competition_testing*’目录下的测试文件中的账户标签。关于所包含标签的详细描述在**附录Ⅱ** 中提供。参赛者可以使用多种不同的算法和技术基于‘*competition_training_dataset*’数据集训练模型。‘*model.py*’中包含了一种示例算法。

2.5 方法建议

鼓励参赛者使用各种机器学习技术和方法来解决这个问题。以下是一些建议的方法：

1. 监督学习：利用已标记的数据训练模型，能够有效地根据已知示例预测标签。这种方法可能包括使用分类算法、回归模型和集成方法来提高预测性能。
2. 特征工程：从现有数据中提取其他相关特征以增强模型输入。
3. 集成方法：堆叠、提升和装袋：使用集成学习方法结合多个模型以提高准确性和稳定性。像随机森林、梯度提升机和堆叠泛化等技术可以特别

有效。

4. 降维：特征选择和提取：应用如主成分分析（PCA）或其他降维技术，以提高模型训练效率并关注最有信息的特征。

2.6 提交要求

参赛者需提交两类文件：

1. 预测答案应遵循‘*demonstrated_answer_format.csv*’ 文件中展示的格式，提交三个不同文件，预测答案应命名为：

‘*prediction_easy_dataset.csv*’

‘*prediction_medium_dataset.csv*’

‘*prediction_hard_dataset.csv*’

2. 比赛中使用的相关代码文件，应包含适当的注释。

2.7 评分标准

本题目满分为 200 分，根据参赛者预测的准确性打分。

定义：准确性定义为正确预测标签的数量与每个数据集中需要标签的地址总数的比例。

具体评分：

• 简单数据集：该数据集上的准确性将占题目总得分的 40%。

• 中等数据集：该数据集上的准确性将占总得分的 30%。

• 困难数据集：具有更高的复杂性和难度，该数据集上的准确性也将占总得分的 30%。

评分示例：

• 假设一位参赛者在简单数据集上达到 80%的准确性，在中等数据集上达到 70%，在困难数据集上达到 60%：

简单数据集得分：$80\% \times 40\% = 32\%$的总分。

中等数据集得分：$70\% \times 30\% = 21\%$的总分。

困难数据集得分：$60\% \times 30\% = 18\%$的总分。

• 总得分：总分的 71%，即 $200 \times 71\% = 142$ 分。

附录I

$competition\_training\_dataset.csv$ 包含从以太坊交易和 ERC -20 代币交易中派生的特征，适用于各种按典型用途（如交易所、钱包等）分类的地址。这些特征在多个时间段内计算，包括交易金额、交易次数、与独特交易对手的互动以及交易之间的时间动态。

具体字段解释：

$Address$：以太坊地址。

$label$：地址的类别标签（例如，'dex' 代表去中心化交易所）。

$lifetime$：地址首次与最后一次交易之间的天数。

$timeframe$：统计数据计算的不同时间段，通常为'$three\_month$'（三个月）、'$half\_a\_year$'（半年）和'one_year'（一年）。

$active\_day$：在给定时间段内的活跃天数。

$Transaction\ Amount\ Features$：包括三个月、半年和一年时间内的交易量总额、收入和支出以太坊金额，及其比例和统计摘要（平均值、最小值、最大值、标准差）。

示例：

$amount\_total$：指定时间段内所有交易的以太坊总金额。

$amount\_mean$：时间段内每笔交易的平均金额。

$amount\_min$：时间段内最小交易金额。

$amount\_max$：时间段内最大交易金额。

$amount\_std\_dev$：时间段内交易金额的标准差。

$amount\_in\_total$：时间段内接收的总以太坊金额。

$amount\_out\_total$：时间段内发送的总以太坊金额。

$amount\_ratio\_in\_out\_total$：收入与支出以太坊总金额的比率。

附录 II

在本次竞赛中，参与者的任务是基于交易数据预测区块链账户的标签。以下是参与者可能遇到的最常见账户类别的简要描述：

1. dex：去中心化交易所允许用户直接相互交易加密货币，无需中介或中央机构。DEX平台通常使用智能合约自动化区块链上的交易，增强了安全性和用户控制。
2. $phishing$：在区块链的背景下，网络钓鱼是指旨在诱使个人透露私人信息（例如加密货币钱包的私钥）的恶意活动。攻击者通常创建欺诈性网站或发送模仿合法服务的欺骗性消息来窃取用户资金。
3. $lending$：区块链借贷平台允许用户借出或借入数字资产，通常用于获取利息。这些平台通常以去中心化方式运作，使用智能合约管理贷款、抵押品和还款。借贷对借贷双方来说都是一个吸引人的选项，因为它提供了流动性并有机会在没有传统中介的情况下赚取利息。
4. cex：中心化交易所(CEX)是由中心化机构运营的加密货币交易平台，该机构作为买卖双方之间的中介。用户将资金存入由交易所管理的账户，由交易所促成交易。尽管CEX平台（如$Coinbase$）对新手更友好，但通常被认为比去中心化交易所更容易受到黑客攻击。
5. $gambling$：区块链上的博彩平台使用户能够在各种机会游戏上投注加密货币，如彩票、扑克和赌场游戏。区块链技术确保了这些平台的透明度和公平性，因为结果通常可以通过智能合约验证。然而，博彩平台有时可能与高风险活动和潜在滥用有关。
6. $wallet$：区块链空间中的钱包是一种数字工具，允许用户存储、发送和接收加密货币。钱包可以是基于软件的（在线或移动设备上）或基于硬件的（物理设备）。
7. $payment$：支付指的是为了商品、服务或履行义务而在各方之间转移数字资产。区块链技术使支付变得快速、安全且无国界限制。

题目三

赛题名称：萝卜快跑

3.1 题目背景

萝卜快跑通过最新的无人驾驶技术为用户提供网约车服务。在本题中你将为萝卜快跑设计网约车智能调度方案。

3.2 题目描述

假设市中心的交通路网是一个边长 26 公里的方格图。城市里有 2 条环线，其中一环的 4 个顶点坐标为(9,4)，(9,18)，(21,18)和(21,4)；二环的 4 个顶点坐标为(3,1)，(3,22)，(24,22)，(24,1)；这 2 条环线将城市分隔为一环内、一环到二环间、和二环外共 3 大区域。萝卜快跑网约车的车库位置在（0，0）坐标；并且还有 3 个充电站，坐标位于(10,18)，(8,8)和(20,12)。

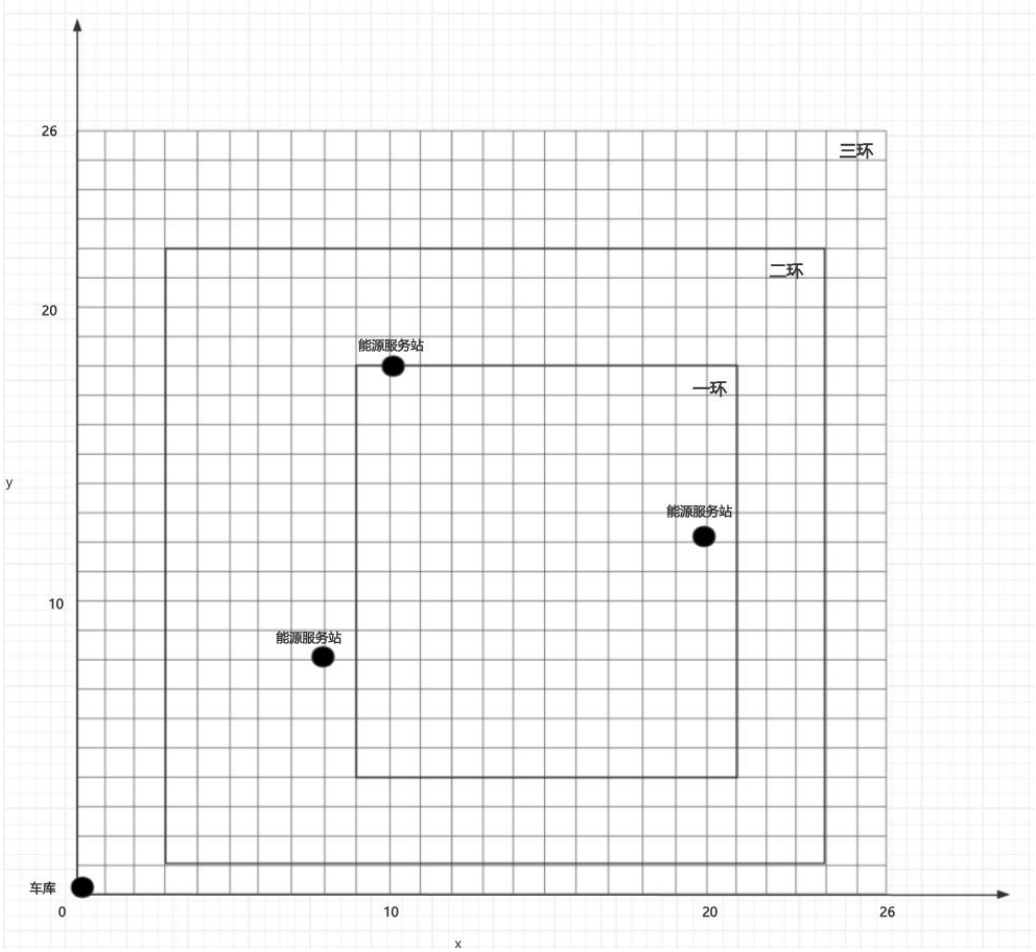

图 3.1

每辆车每天早上 6 点在车库充满电后出发，电量不足时可以去充电站充电，完成全部任务后要返回车库。假设"萝卜快跑"在武汉有 6 辆无人驾驶网约车，为用户提供服务。现在有 50 个用户提出即时用车需求。现实的订单是实时到达，

题目出于为选手进行简化的原因，将订单信息全部给出；并且订单均不考虑拼车的情况。

每个订单为 $\sigma_i = ([h_i, l_i], [m_i, n_i], s_i, e_i, p_i)$ ，表示用户从 s_i 地点上车，到 e_i 地点下车，共 p_i 人。用户期望的最佳服务时间窗口是 $[m_i, n_i]$ 。但如网约车因故出现早到或迟到等情况，网约车允许接受该订单的极限时间窗口是 $[m_i, n_i]$ ；如果网约车至少能够在这个极限时间窗口内到达，订单就可以接受，否则将拒绝。

网约车每分钟的订单收入为1.5元，但早到或迟到（即到达时间在 $[m_i, n_i]$ 内但在 $[h_i, l_i]$ 外）的情况需要惩罚费用。如提前到达造成的每分钟费用系数为0.1（例如停车费等）；迟到造成的每分钟费用系数为0.2（例如赔偿用户优惠券等）。惩罚费用为费用系数乘以惩罚时间。

id	$[h_i, l_i]$	$[m_i, n_i]$	s_i	e_i	p_i
1	[8:15,8:30]	[8:05,8:40]	(4,4)	(12,14)	2
2	[19:15,19:30]	[19:05,19:40]	(13,15)	(16,8)	1
3	[7:30,8:45]	[7:20,8:55]	(25,7)	(4,5)	3
4	[13:45,14:00]	[13:35,14:10]	(8,16)	(12,9)	1
5	[18:00,18:15]	[17:50,18:25]	(16,17)	(1,8)	1
6	[13:00,13:15]	[12:50,13:25]	(4,17)	(21,23)	4
7	[7:45,8:30]	[7:35,8:45]	(20,11)	(18,9)	2
8	[8:15,8:30]	[8:05,8:45]	(7,3)	(5,19)	3
9	[16:45,17:00]	[16:35,17:10]	(14,6)	(7,17)	1
10	[9:45,10:00]	[9:35,10:10]	(19,23)	(8,8)	4
11	[7:00,7:15]	[6:50,7:25]	(1,7)	(11,4)	2
12	[10:00,10:15]	[09:50,10:25]	(13,20)	(6,12)	2
13	[11:30,11:45]	[11:20,11:55]	(10,11)	(23,18)	1
14	[15:00,15:15]	[14:50,15:25]	(22,23)	(20,9)	4
15	[19:15,19:30]	[19:05,19:40]	(16,5)	(10,18)	2
16	[11:30,11:45]	[11:20,11:55]	(15,7)	(15,20)	3
17	[17:00,17:15]	[16:50,17:25]	(2,11)	(9,6)	1
18	[8:45,9:00]	[8:35,9:10]	(0,1)	(19,7)	1
19	[7:30,7:45]	[7:20,7:55]	(20,21)	(5,9)	1
20	[8:30,8:45]	[8:20,8:55]	(20,14)	(9,1)	1
21	[18:45,19:00]	[18:35,19:10]	(10,14)	(0,18)	2
22	[18:15,18:30]	[18:05,18:40]	(23,13)	(17,11)	2
23	[7:30,7:45]	[7:20,7:55]	(12,2)	(10,9)	4

24	[16:45,17:00]	[16:35,17:10]	(10,2)	(17,15)	1
25	[9:00,9:15]	[8:50,9:25]	(20,13)	(8,9)	2
26	[8:15,8:30]	[8:05,8:40]	(2,19)	(14,13)	3
27	[7:30,7:45]	[7:20,7:55]	(5,23)	(1,16)	4
28	[8:45,9:00]	[8:35,9:10]	(4,9)	(20,5)	2
29	[8:30,8:45]	[8:20,8:55]	(12,14)	(10,18)	1
30	[7:45,8:00]	[7:35,8:10]	(17,12)	(20,6)	2
31	[21:15,21:30]	[21:05,21:40]	(19,0)	(15,8)	2
32	[9:00,9:15]	[8:50,9:25]	(20,9)	(14,19)	1
33	[8:30,8:45]	[8:20,8:55]	(8,4)	(1,7)	2
34	[9:00,9:15]	[8:50,9:25]	(18,20)	(23,11)	3
35	[12:00,12:15]	[11:50,12:25]	(18,8)	(12,20)	1
36	[8:15,8:30]	[8:05,8:40]	(5,13)	(9,10)	2
37	[7:27,7:35]	[7:17,7:45]	(25,19)	(16,14)	3
38	[20:45,21:00]	[20:35,21:10]	(15,13)	(2,20)	2
39	[15:00,15:15]	[14:50,15:25]	(4,24)	(8,18)	1
40	[7:00,7:15]	[6:50,7:25]	(4,2)	(9,5)	4
41	[15:45,16:00]	[15:35,16:10]	(7,11)	(15,15)	1
42	[18:15,18:30]	[18:05,18:40]	(22,3)	(16,18)	1
43	[8:15,8:30]	[8:05,8:40]	(7,24)	(10,9)	2
44	[8:15,8:30]	[8:05,8:40]	(22,0)	(16,3)	1
45	[15:15,15:35]	[15:05,15:45]	(7,19)	(10,15)	1
46	[18:15,18:30]	[18:05,18:40]	(25,20)	(22,17)	3
47	[7:45,8:00]	[7:35,8:10]	(10,21)	(15,17)	2
48	[10:30,10:40]	[10:20,10:50]	(6,3)	(9,18)	4
49	[8:45,9:00]	[8:35,9:10]	(7,10)	(16,11)	1
50	[11:45,12:00]	[11:35,12:10]	(17,7)	(24,12)	1

表 3.1

请根据订单需求、电量消耗（见下）与充电判断，设计一个智能调度系统，让“萝卜快跑”无人驾驶网约车的总收益最大。

3.3 其他假设

车辆需沿方格图网格线行走

行驶速度为$V = v_base/D$

时段	00-05	05-10	10-17	17-21	21-00
平均行驶时速（km/h）	60	30	45	30	50

表 3.2 不同时间段的车辆平均行驶时速（v_base）

道路	一环以内（含一环）	一环 ～ 二环（含二环）	二环以外
拥堵度	D = 2	D = 1.5	D = 1

表 3.3 不同道路拥堵度（D）

时段	00-07	07-10	10-15	15-18	18-21	21-00
电价【元/($kW \cdot h$)】	0.35	0.85	1.23	0.85	1.23	0.85

表 3.4 不同时间段的电价

每辆车的电池容量为 50 $kW \cdot h$ ，充电功率为 10 kW

平均每小时耗电为$E0$=5 $kW \cdot h$ 。受承载人数 P 影响，实际耗电为

$$E = e_0 \times (1 + k_1 \times (p - 2)/3)$$

$$k_1 = 0.33$$

3.4 输入数据

输入数据在$input.md$。

3.5 输出结果

最大运营收益路径，包含：

(1) 总运营收益；

(2) 每辆车的行驶路径、接单、和充电记录（按行驶顺序排列）。

输出样例：

程序应返回两个变量 $income$ 和 $vehicles$ 。其中 $income$ 是总运营收益；$vehicles$ 应包含 6 个表项，分别描述每辆车的行驶路径、接单信息等，其中接单 id 为 0 表示这段行程不属于任何订单。

注意：最终只需提交程序脚本，不需要直接提供输出结果。

3.6 示例

第一辆车 $vehicles1$ 的 3 条行驶记录：

(1) 第 1 条为从起点到达接单位置，因早到产生订单惩罚 0.5；

(2) 第 2 条为从接单起点到达接单终点，订单行驶收益 30.10；

(3) 第 3 条为去往充电站，并充电 0.5 小时，充了 $5kW$ ，花费 17；

注意：该样例只提示输出格式，不反映实际运行数据。

```
income = 24.41
vehicles = [
```

```
    # vehicle1，第一辆车的行驶路径，按行驶顺序依次排序：
        pd.DataFrame({
            '出发时间': ['06:00:00', '08:55:00', '09:32:00'],
            '出发位置': [(0,0), (20,9), (14,19)],
            '到达位置': [(20,9), (14,19), (10,18)],
            '接单 id': [0, 32, 0],
            '到达时间': ['08:55:00', '09:32:00', '09:45:00'],
            '订单收益': [0, 30.10, 0],
            '订单惩罚': [0.5, 0, 0],
            '充电时长': [0, 0, 0.5], #单位小时
            '充电量': [ 0, 0, 5],
            '充电费用': [0, 0, 17]
        }),
    # vehicle2
    …
    # vehicle3
    …
    # vehicle4
    …
    # vehicle5
    …
    # vehicle6
    …
]
```

3.7 评分标准

本题共 200 分。

(1) 首先校验运行路径和收益的正确性，即执行选手提交的程序获取输出，对输出执行 *test_script.py* 验证是否通过。

(2) 注意需针对所有车辆路径都运行正确才通过，比如到达时间符合可接受时间窗口、单个订单有且仅有一次车辆服务、一旦接单必须完成且中途不得充电等。若通过则进行收益排名， *income* 收益越多，排名越靠前。如某个路径出现运行错误则不参与排名也不得分。

(3) 在运行正确且 *income* 相同的情况下，程序运行时间越短排名越靠前。程序运行时间由各语言的框架计算，选手不需要自己统计。

(4) 在运行正确的前提下将收益与参考收益做比较，根据近似比打分。例如参考收益为 300，选手达到的收益为 240，则得分为 $200 \times 240/300 = 160$，得分超过 200 的均记为 200。