数据结构与算法 I 实验报告

实验 2：区块链 (1)

龚舒凯 2022202790 应用经济-数据科学实验班

https://github.com/GONGSHUKAI

2023 年 10 月 26 日

区块链 (1)

1 需求分析

问题描述：区块链，就是一组区块构成的“链”，也就是链表。下面关于区块和交易的定义来自比特币系统，根据任务需要作了简化。在比特币系统中，采用的是 UTXO 模型，即每个交易将使用前面某个交易的输出 (output) 来作为当前交易的输入 (input)。而当前交易的输出将作为未来某个交易的输入。区块链中各要素的定义如下：

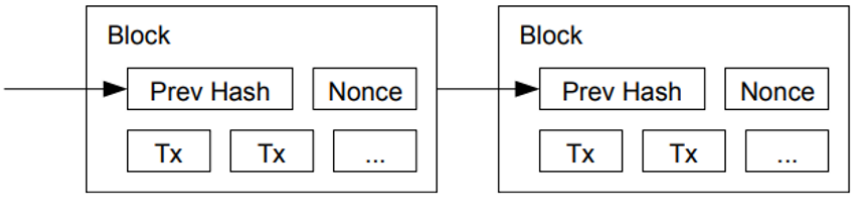

图 1: 区块链示意图

- **区块 (Block)**：区块为区块链中的基本单位，其内涵成员如下：

成员名	类型	说明
height	整数	当前块的高度，一条链上每个区块的 Height 均不相同。
hash	字符串	本区块的哈希值。
prevHash	字符串	前一个区块的哈希值，本实验中可以置空
merkleRoot	字符串	本区块中所有交易的默克尔树根，本实验中可以置空
nonce	整数	神秘数，本实验可以忽略
transactions	数组	一组交易 (transaction) 的集合

- **交易 (Transaction)**：

成员名	类型	说明
txid	字符串	交易的编号，具有唯一性
input_count	整数	inputs 的数量，本实验可以忽略
output_count	整数	outpus 的数量，本实验可以忽略
inputs	数组	一组 input 的集合，表示当前交易的输入所用到的输出
outputs	数组	一组 output 的集合，表示当前交易的输出，可能作为后续交易的输入
is_coinbase	整数	表示是否为 coinbase 交易（1 为 coinbase 交易，0 为非 coinbase 交易）

- **输入 (Input)**：

成员名	类型	说明
pre_block	整数	该 input 所引用的 output 所在区块的高度;
prevTxID	整数	该 input 所引用的 output 所在交易的 txID
prevTxOutIndex	整数	该 input 所引用的 output 位于所在交易 output 集合中的索引
scriptSig	字符串	脚本和签名，本实验中可以置空

- **输出 (Output):**

成员名	类型	说明
txid	字符串	该 output 所属的交易
index	整数	该 output 在所属交易中的索引值
value	整数	该 output 的价值
script	字符串	脚本，本实验可以置空

基本要求:

1. 读入区块相关数据文件 demo.zip(包括 blocks.csv,transaction.csv,inputs.csv,outputs.csv 四个文件)，生成区块数据变量，将所有区块按高度顺序组织成链表。
2. 检验区块链中交易 (Transactions) 的合法性：根据如下三条规则判断区块中交易合法与否:
   (a) 每个 input 所使用的 output 能够找到。(例外：coinbase= 1 的交易可以没有 input，只有 output。该类交易是合法的，其中的 output 可能被后续的 transaction 所引用。)
   (b) 每个 input 所使用的 output 没有被之前的交易用过，且引用的 output 必须来自合法的交易。
   (c) 该交易所有 input 所引用的 output 的价值 (value) 之和大于等于该交易所有 output 的价值 (value) 之和。

输出形式:

1. 打印区块总数、合法交易总数、不合法的交易总数。
2. 从键盘输入区块高度 (Height)，输出该区块内容。
3. 从键盘输入交易号 txid，输出该交易内容。

2 具体实现

2.1 区块链的定义

根据之前对区块结构的分析，将区块中的 Block, Transaction, Input, Output 定义如下：

```
#define MAXTRANS 100//一个块内最高有 100 条交易信息
#define MAXINPUT 100//一条交易信息中最高有 100 个输入
#define MAXOUTPUT 100//一条交易信息中最高有 100 个输出

typedef struct output{
    string txid;//该 output 所属的交易
    unsigned long long index;//该 output 在所属交易中的索引值
    unsigned long long value;//该 output 的价值 (数据已乘 10^8, 避免浮点误差)
    string script;//脚本

    int IsUse = NotUsed;
}output;

typedef struct input{
    unsigned long long pre_block;//该 input 所引用的 output 所在区块的高度
    string prevTxID;//该 input 所引用的 output 所在交易的 txID
    unsigned long long prevTxOutIndex;//该 input 所引用的 output 位于所在交易 output 集合中的索引
    string scriptSig;//脚本和签名
}input;

typedef struct transaction{
    string txid;//交易的编号, 具有唯一性
    unsigned long long input_count;//inputs 的数量
    unsigned long long output_count;//outputs 的数量
    input inputs[MAXINPUT];//一组 input 的集合, 表示当前交易的输入所用到的输出
    output outputs[MAXOUTPUT];//一组 output 的集合, 表示当前交易的输出
    int is_coinbase;//1 为 coinbase 交易, 0 为非 coinbase 交易

    int valid = yes;
}transaction;

typedef struct block{
    unsigned long long height;//当前块的高度, 一条链上每个区块的 Height 均不相同
    string hash;//本区块的哈希值
    string prevHash;//前一个区块的哈希值
    string merkleRoot;//本区块中所有交易的默克尔树根
    unsigned long long nonce;//神秘数
```

```
38      transaction transactions[MAXTRANS];//一组 transaction 的集合
39      struct block *next;
40  }block;
```

容易看出，Block 与 Block 之间通过指针链接。Block 内的成员 transaction 又有 txid, inputs, outputs, coinbase 等成员，每个 transaction 里的 inputs, outputs 又有成员 prevTxID, script 等等。因此，这样定义下的区块链是一个复杂的、相互连接嵌套的数据容器。

2.2 输入输出流 (I/O Stream)

区块的信息储存在四个数据表格中：`blocks.csv,transaction.csv,inputs.csv,outputs.csv`。在使用结构体定义完区块各组成部分后，需要将数据有序的填入。这里分别使用 4 个函数来读取 4 个文件：

```
block* FileToBlock();
void FileToTransaction(block *currentBlock);
void FileToInput(block *currentBlock);
void FileToOutput(block *currentBlock);
```

- `block* FileToBlock();`一行行的读取 block.csv，以区块高度 Height 为依据创造一个个区块并读入数据。
- `void FileToTransaction(block *currentBlock);`一行行的读取 transaction.csv，对于每行的交易 (transactions[i])，先找到该交易对应的区块 Height 找到应存入的区块 currentBlock，再将数据存入 currentBlock。
- `void FileToInput(block *currentBlock);`一行行的读取 inputs.csv，对于每行输入 (inputs[i])，先找到该输入对应的区块 currentBlock，再找到使用该输入的交易，然后将数据存入该交易的 inputs 数组。
- `void FileToOutput(block *currentBlock);`一行行的读入 outputs.csv，对于每行输出 (outputs[i])，先找到该输出对应的区块 currentBlock，再找到给出该输出的交易，然后将数据存入该交易的 outputs 数组。

读入数据时，

1. 调用库`fstream`中的`ifstream file(const char [])`打开 csv 文件。
2. 首先用`getline(file, line)`一行行的读取 csv 文件，将当前行的数据读入到一个字符串`line`中。
3. 由于 csv 文件在实际储存时不同数据间用逗号分隔，因此调用库`sstream`创建一个`stringstream ss`保存`line`
4. 然后使用逗号作为分隔符，从`ss`中提取每个字段的内容，并将其存储到字符串变量`cell`中，将一个个`cell`存入区块及其结构中。
5. 重复上述过程，一行行遍历 csv 文件直到到文件末尾结束。

具体代码实现见附录。

2.3 交易合法性判断

在判断区块中所有交易的合法性时，需要使用以下函数：

```
void CheckValidTransaction(block *firstblock);
bool IOCheck(block *firstblock, block *currentblock, transaction t);
transaction* FindPrevOutput(string My_txid, block *firstblock, block *endblock);
```

- `void CheckValidTransaction(block *firstblock);`从第一个区块开始，检查整个区块链中所有的不合法交易。
- `bool IOCheck(block *firstblock, block *currentblock, transaction t);`从第一个区块 (firstblock) 开始一直到当前区块 (currentBlock)，检查交易`t`用到的输入输出是否合法。检验合法性的三条规则即：
  1. 每个 input 所使用的 output 能够找到。(例外：coinbase= 1 的交易可以没有 input，只有 output。该类交易是合法的，其中的 output 可能被后续的 transaction 所引用。)
  2. 每个 input 所使用的 output 没有被之前的交易用过，且引用的 output 必须来自合法的交易。
  3. 该交易所有 input 所引用的 output 的价值 (value) 之和大于等于该交易所有 output 的价值 (value) 之和。
- `transaction* FindPrevOutput(string My_txid, block *firstblock, block *endblock);`从第一个区块 (firstblock) 开始一直到当前区块 (currentBlock)，寻找`My_txid`对应的交易。用到这个函数是因为执行上述 1, 2, 3 判断首先均需要定位到“当前 input 所用的 output“。

交易合法性判断的流程如下，

1. 首先调用`void CheckValidTransaction(block *firstblock);`，遍历每一个区块中的每一条交易
2. 对于区块 i 的第 j 条交易，先执行特判，如果该交易的 coinbase= 1 则该交易一定是合法交易
3. 如果该交易的 coinbase= 0,则调用`bool IOCheck(block *firstblock, block *currentblock, transaction t);`，检查该交易的输入输出合法性：
   (a) 如果交易不合法，则标记区块 i 的第 j 条交易为“不合法”(`block->transactions[i].valid = no;`)，不合法交易计数器`invalid++;`，继续判断区块 i 的第 $j+1$ 条交易的合法情况。
   (b) 如果交易合法，则合法交易计数器`valid++;`，继续判断区块 i 的第 $j+1$ 条交易的合法情况。

3 使用说明

这里我们读取 2009 年比特币全部区块数据。首先，程序需要一段时间读取四个 csv 文件。读取完毕后，将显示

```
Time spent reading data: 35.8814s
Block count: 32490
不合法交易数: 218
合法交易数: 32491
```

接着显示

```
Please input block's height:
```

此时输入一个区块的 Height(以输入 114 为例)，将显示该区块的信息：

```
Block Height: 114
Block Hash: 000000005d7dd40c24c4b6334812f48d4386c7756fb9006d6c74300837c91ebd
Block prevHash: 0000000019176838de40606d70738084f2fbc48a50548eeeac3ceb857677c6d
Block merkleRoot: ca7b0295546806c9b6631f8bc89ed0b55434ee5f8ce6198a918af18c6da9237c
Block nonce: 2979771436

Block Transaction 0 txid: ca7b0295546806c9b6631f8bc89ed0b55434ee5f8ce6198a918af18c6da9237c
Block Transaction 0 input count: 1
Block Transaction 0 output count: 1
Block Transaction 0 Coinbase: 1
Transaction 0 in Block: 114
Transaction 0 input count: 1
Transaction 0 output count: 1
Transaction 0 Coinbase: 1
Transaction 0 output 0 txid: ca7b0295546806c9b6631f8bc89ed0b55434ee5f8ce6198a918af18c6da9237c
Transaction 0 output 0 index: 0
Transaction 0 output 0 value: 500000
Transaction 0 output 0 script: 4104cd10ce592a9c4918948a5b4e92e9702e6c36eed1a627594f67066792b3a227cedb
```

接着显示

```
Please input transaction's txid:
```

此时输入一个交易的 txid(以输入 d9df26d62a3ce4855f3282462ba5581b23dc51ca3595de810115c0e7176722d3 为例)，将显示该交易所属的区块，该交易共有几个 input，几个 output，coinbase 是什么：

```
Transaction 0 in Block: 514
Transaction 0 input count: 1
Transaction 0 output count: 1
Transaction 0 Coinbase: 1
```

4 附录

运行代码见https://github.com/GONGSHUKAI/Data_Structure/tree/main/Lab_Code/Lab_2/Oct.8_Lab

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <ctime>
#define yes 100
#define no -100
#define NotUsed 200
#define Used -200
#define TransactionExist 999

#define MAXTRANS 100//一个块内最高有 100 条交易信息
#define MAXINPUT 100//一条交易信息中最高有 100 个输入
#define MAXOUTPUT 100//一条交易信息中最高有 100 个输出
using namespace std;

typedef struct output{
    string txid;//该 output 所属的交易
    unsigned long long index;//该 output 在所属交易中的索引值
    unsigned long long value;//该 output 的价值 (数据已乘 10^8, 避免浮点误差)
    string script;//脚本

    int IsUse = NotUsed;
}output;

typedef struct input{
    unsigned long long pre_block;//该 input 所引用的 output 所在区块的高度
    string prevTxID;//该 input 所引用的 output 所在交易的 txID
    unsigned long long prevTxOutIndex;//该 input 所引用的 output 位于所在交易 output 集合中的索引
    string scriptSig;//脚本和签名
}input;

typedef struct transaction{
    string txid;//交易的编号, 具有唯一性
    unsigned long long input_count;//inputs 的数量
    unsigned long long output_count;//outputs 的数量
    input inputs[MAXINPUT];//一组 input 的集合, 表示当前交易的输入所用到的输出
    output outputs[MAXOUTPUT];//一组 output 的集合, 表示当前交易的输出
    int is_coinbase;//1 为 coinbase 交易, 0 为非 coinbase 交易
```

```
40
41      int valid = yes;
42  }transaction;
43
44  typedef struct block{
45      unsigned long long height;//当前块的高度，一条链上每个区块的 Height 均不相同
46      string hash;//本区块的哈希值
47      string prevHash;//前一个区块的哈希值
48      string merkleRoot;//本区块中所有交易的默克尔树根
49      unsigned long long nonce;//神秘数
50      transaction transactions[MAXTRANS];//一组 transaction 的集合
51
52      struct block *next;
53  }block;
54
55  block* FileToBlock();
56  void FileToTransaction(block *currentBlock);
57  void FileToInput(block *currentBlock);
58  void FileToOutput(block *currentBlock);
59  int BlocksLength(block *firstblock);
60  void BlockInfo(int height, block *firstblock);
61  int TransactionInfo(string txid, block *firstblock, block *endblock);
62  block* InitBlockChain();
63  void CheckValidTransaction(block *firstblock);
64  bool IOCheck(block *firstblock, block *currentblock, transaction t);
65  transaction* FindPrevOutput(string My_txid, block *firstblock, block *endblock);
66
67  block* FileToBlock() {
68      block *firstBlock = new block;
69      ifstream file("2009data/blocks.csv"); //打开 CSV 文件
70      if (!file) {
71          cout << " 无法打开文件" << endl;
72          return nullptr;
73      }
74      string line;
75      getline(file, line); // 读取第一行标题（忽略）
76      /*getline 函数返回一个布尔值，表示是否成功读取一行内容。
77      如果读取成功，则返回 true;
78      如果已到达文件末尾或发生错误，则返回 false。*/
79      block *currentBlock = nullptr;
80      block *prevBlock = nullptr;
81      while (getline(file, line)) {
```

```
82          stringstream ss(line);//创建一个 stringstream 对象，字符串 ss 作为初始输入。
83          string cell;
84          int column = 0;
85          block *newBlock = new block();// 创建新的 block
86
87          while (getline(ss, cell, ',')){
88          //使用逗号作为分隔符，从 ss 中提取每个字段的内容，并将其存储到字符串变量 cell 中
89              switch (column) {
90                  case 0:
91                      newBlock->height = stoull(cell);//stoi 函数将字符串 cell 转换为 unsigned long long
92                      break;
93                  case 1:
94                      newBlock->hash = cell;
95                      break;
96                  case 2:
97                      newBlock->prevHash = cell;
98                      break;
99                  case 3:
100                     newBlock->merkleRoot = cell;
101                     break;
102                 case 4:
103                     newBlock->nonce = stoull(cell);
104                     break;
105                 default://当没有匹配到任何 case 标签时执行的代码块。
106                     break;
107             }
108             column++;
109         }
110         // 如果是第一个 block，则将其设置为第一个 block
111         if (prevBlock == nullptr) {
112             firstBlock = newBlock;
113             currentBlock = newBlock;
114         } else {
115             prevBlock->next = newBlock; // 将前一个 block 的 next 指针指向新的 block
116             newBlock->next = nullptr;
117             currentBlock = newBlock;
118         }
119         prevBlock = currentBlock;
120     }
121     file.close(); // 关闭文件
122     return firstBlock;
123 }
```

```
124
125 void FileToTransaction(block *currentBlock){
126     ifstream file("2009data/transactions.csv");
127     if (!file) {
128         cout << " 无法打开文件" << endl;
129         return;
130     }
131     string line;
132     getline(file, line);
133     int transIndex = 0;
134
135     while (getline(file, line)) {
136         stringstream ss(line);
137         string cell;
138         int column = 0;
139         int transHeight = 0;
140         transaction currentTrans;
141         currentTrans.valid = yes;
142         while (getline(ss, cell, ',')) {//扫描一行的数据，以逗号为分隔符
143             switch (column) {
144                 case 0:
145                     transHeight = stoull(cell);
146                     break;
147                 case 1:
148                     currentTrans.txid = cell;
149                     break;
150                 case 2:
151                     currentTrans.is_coinbase = stoi(cell);
152                     break;
153                 case 3:
154                     currentTrans.input_count = stoull(cell);
155                     break;
156                 case 4:
157                     currentTrans.output_count = stoull(cell);
158                     break;
159                 default:
160                     break;
161             }
162             column++;
163         }
164         if (transHeight == currentBlock->height){
165             currentBlock->transactions[transIndex] = currentTrans;
```

```
166             transIndex++;
167         }
168         else{
169             while (currentBlock->height != transHeight){
170                 currentBlock = currentBlock->next;
171             }
172             transIndex = 0;
173             currentBlock->transactions[transIndex] = currentTrans;
174             transIndex++;
175         }
176     }
177     file.close();
178 }
179 
180 void FileToInput(block *currentBlock){
181     ifstream file("2009data/inputs.csv");
182     if (!file) {
183         cout << " 无法打开文件" << endl;
184         return;
185     }
186     string line;
187     getline(file, line);
188     int transIndex = 0;
189     int inputsIndex = 0;
190     while (getline(file, line)) {
191         stringstream ss(line);
192         string cell;
193         int column = 0;
194         int inputHeight = 0;
195         string input_txid;
196         input currentInput;
197         while (getline(ss, cell, ',')) {//扫描一行的数据，以逗号为分隔符
198             switch (column) {
199                 case 0:
200                     inputHeight = stoull(cell);
201                     break;
202                 case 1:
203                     input_txid = cell;
204                     break;
205                 case 2:
206                     currentInput.pre_block = stoi(cell);
207                     break;
```

```
208                 case 3:
209                     currentInput.prevTxID = cell;
210                     break;
211                 case 4:
212                     currentInput.prevTxOutIndex = stoull(cell);
213                     break;
214                 case 5:
215                     currentInput.scriptSig = cell;
216                 default:
217                     break;
218             }
219             column++;
220         }
221         if (inputHeight == currentBlock->height && input_txid == currentBlock->transactions[transInde
222             currentBlock->transactions[transIndex].inputs[inputsIndex] = currentInput;
223             inputsIndex++;
224         }
225         else{
226             while (currentBlock->height != inputHeight){
227                 currentBlock = currentBlock->next;
228             }
229             while (currentBlock->transactions[transIndex].txid != input_txid && currentBlock->transac
230                 transIndex++;
231             }
232             inputsIndex = 0;
233             currentBlock->transactions[transIndex].inputs[inputsIndex] = currentInput;
234             inputsIndex++;
235         }
236     }
237     file.close();
238 }
239
240 void FileToOutput(block *currentBlock){
241     ifstream file("2009data/outputs.csv");
242     if (!file) {
243         cout << " 无法打开文件" << endl;
244         return;
245     }
246     string line;
247     getline(file, line);
248     int transIndex = 0;
249     int outputsIndex = 0;
```

```
250         while (getline(file, line)) {
251             stringstream ss(line);
252             string cell;
253             int column = 0;
254             int outputHeight = 0;
255             string output_txid;
256             output currentOutput;
257             currentOutput.IsUse = NotUsed;
258             while (getline(ss, cell, ',')) {//扫描一行的数据，以逗号为分隔符
259                 switch (column) {
260                     case 0:
261                         outputHeight = stoull(cell);
262                         break;
263                     case 1:
264                         currentOutput.txid = cell;
265                         break;
266                     case 2:
267                         currentOutput.index = stoi(cell);
268                         break;
269                     case 3:
270                         currentOutput.value = stoull(cell);
271                         break;
272                     case 4:
273                         currentOutput.script = cell;
274                         break;
275                     default:
276                         break;
277                 }
278                 column++;
279             }
280             if (outputHeight == currentBlock->height && currentOutput.txid == currentBlock->transactions[
281                 currentBlock->transactions[transIndex].outputs[outputsIndex] = currentOutput;
282                 outputsIndex++;
283             }
284             else{
285                 while (currentBlock->height != outputHeight){
286                     currentBlock = currentBlock->next;
287                 }
288                 while (currentBlock->transactions[transIndex].txid != currentOutput.txid && currentBlock-
289                     transIndex++;
290                 }
291                 outputsIndex = 0;
```

```
292                 currentBlock->transactions[transIndex].outputs[outputsIndex] = currentOutput;
293                 outputsIndex++;
294             }
295         }
296         file.close();
297     }
298 
299     void BlockInfo(int height, block *firstblock){
300         //输入区块高度，输出该区块内容
301         block *p = firstblock;
302         while (height != p->height && p != nullptr){
303             p = p->next;
304         }//找到 height 值对应的 block
305         cout << "Block Height: " << p->height << endl;
306         cout << "Block Hash: " << p->hash << endl;
307         cout << "Block prevHash: " << p->prevHash << endl;
308         cout << "Block merkleRoot: " << p->merkleRoot << endl;
309         cout << "Block nonce: " << p->nonce << endl;
310         int i = 0;
311         cout << endl;
312         while (p->transactions[i].txid != ""){
313             cout << "Block Transaction " << i << " txid: " << (p->transactions[i]).txid << endl;
314             cout << "Block Transaction " << i << " input count: " << (p->transactions[i]).input_count <<
315             cout << "Block Transaction " << i << " output count: " << (p->transactions[i]).output_count <
316             cout << "Block Transaction " << i << " Coinbase: " << (p->transactions[i]).is_coinbase << end
317             TransactionInfo(p->transactions[i].txid, firstblock, nullptr);
318             cout << endl;
319             i++;
320         }
321         cout << endl;
322     }
323 
324     int BlocksLength(block *firstblock){
325         block *p = firstblock;
326         int len = 0;
327         while (p != nullptr){
328             len++;
329             p = p->next;
330         }
331         return len;
332     }
333 
```

```
334 int TransactionInfo(string My_txid, block *firstblock, block *endblock){
335     block *p = firstblock;
336     int find = 0;//找到交易信息则 find = 1, 否则 find = 0
337     int i = 0;
338     int j = 0;
339     int k = 0;
340     while (p != endblock){
341         while (p->transactions[i].txid != ""){
342             if (p->transactions[i].txid == My_txid){
343                 cout << "Transaction "<< i <<" in Block: " << p->height << endl;
344                 cout << "Transaction "<< i <<" input count: " << (p->transactions[i]).input_count <<
345                 cout << "Transaction "<< i <<" output count: " << (p->transactions[i]).output_count <
346                 cout << "Transaction "<< i <<" Coinbase: " << (p->transactions[i]).is_coinbase << end
347                 while (p->transactions[i].inputs[j].scriptSig != ""){
348                     cout << "Transaction "<< i <<" input " << j << " pre_block: " << (p->transactions
349                     cout << "Transaction "<< i <<" input " << j << " prevTxID: " << (p->transactions[
350                     cout << "Transaction "<< i <<" input " << j << " prevTxOutIndex: " << (p->transac
351                     cout << "Transaction "<< i <<" input " << j << " scriptSig: " << (p->transactions
352                     j++;
353                 }
354                 while (p->transactions[i].outputs[k].script != ""){
355                     cout << "Transaction "<< i << " output " << k << " txid: " << (p->transactions[i]
356                     cout << "Transaction "<< i << " output " << k << " index: " << (p->transactions[i
357                     cout << "Transaction "<< i << " output " << k << " value: " << (p->transactions[i
358                     cout << "Transaction "<< i << " output " << k << " script: " << (p->transactions[
359                     k++;
360                 }
361                 find = 1;//找到这条交易记录
362                 break;
363             }
364             i++;
365         }
366         if (find == 1) break;
367         else{
368             i = 0;
369             p = p->next;
370         }
371     }
372     if (find == 1) return find;
373     else{
374         cout << "Transaction Not Found!" << endl;
375         return find;
```

```
376         }
377     }
378
379     block* InitBlockChain(){
380         block* firstBlock = FileToBlock();
381         FileToTransaction(firstBlock);
382         FileToInput(firstBlock);
383         FileToOutput(firstBlock);
384         return firstBlock;
385     }
386
387     void CheckValidTransaction(block *firstblock){
388         block *p = firstblock;
389         int i = 0;
390
391         int valid = 0;
392         int invalid = 0;
393         //规则 1: 每个 input 所使用的 output 能够找到。
394         //规则 2: 每个 input 所使用的 output 没有被之前的交易用过。
395         //规则 3: 该交易所有 input 所引用的 output 的价值（value）之和大于等于该交易所有 output 的价值（valu
396
397         //a、有一类特殊交易，其 is_coinbase 字段为 true，该类交易的特点是没有 input，只有 output。该类交易是
398         //b、每一个 output 只能被使用一次，即便还有剩余的 value 没有被使用。
399         //c、如果某个交易是非法的，那么引用了该交易作为 input 的交易也同样是非法的（非法交易不会被包括在区块内
400         while (p != nullptr){
401             while (p->transactions[i].txid != ""){
402                 //规则 1: 其 is_coinbase 字段为 true，该类交易的特点是没有 input，只有 output。该类交易是合法
403                 if (p->transactions[i].is_coinbase == 1){
404                     valid++;
405                     i++;
406                     continue;
407                 }
408                 //规则 2: 如果每个 input 所使用的 output 找不到，交易不合法；
409                 //规则 3: 每个 input 所使用的 output 被之前的交易用过，交易不合法
410                 //规则 4: 如果引用的 output 来自不合法的交易，交易不合法
411                 //规则 5: 所引用的 output 的 value 之和 < 该所有 output 的 value 之和，交易不合法
412                 else if (IOCheck(firstblock, p, p->transactions[i]) == false){
413                     p->transactions[i].valid = no;
414                     invalid++;
415                     i++;
416                     continue;
417                 }
```

```
418             else{//
419                 valid++;
420                 i++;
421                 continue;
422             }
423         }
424         i = 0;
425         p = p->next;
426     }
427 
428     cout << " 不合法交易数: " << invalid << endl;
429     cout << " 合法交易数: " << valid << endl;
430 }
431 
432 bool IOCheck(block *firstblock, block *currentblock, transaction t){
433     int i = 0;
434     while (t.inputs[i].scriptSig != ""){
435         transaction *PrevOutput = FindPrevOutput(t.inputs[i].prevTxID, firstblock, currentblock);
436         if (PrevOutput != nullptr){
437             if (PrevOutput->outputs[t.inputs[i].prevTxOutIndex].IsUse == NotUsed &&
438                 PrevOutput->valid == yes){
439                 int sum_in = 0;
440                 int sum_out = 0;
441                 for (int j = 0 ; j < t.input_count ; j++) sum_in += PrevOutput->outputs[t.inputs[j].p
442                 for (int j = 0 ; j < t.output_count ; j++) sum_out += t.outputs[j].value;
443                 if (sum_in >= sum_out){
444                     PrevOutput->outputs[t.inputs[i].prevTxOutIndex].IsUse = Used;
445                     return true;
446                 }
447                 else return false;
448             }
449         }
450         i++;
451     }
452     return false;
453 }
454 
455 transaction* FindPrevOutput(string My_txid, block *firstblock, block *endblock){
456     //根据 PrevTxid 找到当前 input 所用的 output 的 Txid, 再根据当前 prevTxOutIndex 找到所用的 output
457     block *p = firstblock;
458     int i = 0;
459     while (p != endblock){
```

```
460             while (p->transactions[i].txid != ""){
461                 if (p->transactions[i].txid == My_txid) return &p->transactions[i];
462                 else i++;
463             }
464             i = 0;
465             p = p->next;
466         }
467         return nullptr;
468     }
469
470     int main(){
471         clock_t startTime = clock();//计时开始
472         block* firstBlock = InitBlockChain();
473         clock_t endTime = clock();//计时结束
474         cout << "Time spent reading data: " <<(double)(endTime - startTime) / CLOCKS_PER_SEC << "s" << en
475
476         cout << "Block count: " << BlocksLength(firstBlock) << endl;
477         CheckValidTransaction(firstBlock);
478         cout << endl;
479         int height;
480         cout << "Please input block's height: " << endl;
481         cin >> height;
482         cout << endl;
483         BlockInfo(height, firstBlock);
484
485         string txid;
486         cout << "Please input transaction's txid: " << endl;
487         cin >> txid;
488         cout << endl;
489         TransactionInfo(txid, firstBlock, nullptr);
490
491         return 0;
492     }
```