PROJECT REPORT
Group 45
Topic : Bitcoin

Team Members:

1. Rishabh Agrawal
2. Mihir Rawat
3. Hitesh Goel
4. Anjali Singh
5. Ishanya Sethi

GitHub Repo link :
https://github.com/rishabh770/Bitcoin

Contributions of all members:

1) Rishabh Agrawal (2020102038)

- Wrote functions for user registration, user deletion, find user, transaction history, and transaction.
- Also added a function that changes value of coin according to situations.
- Wrote Readme file of project.
- Contributed in debugging and error handling of the programme.

2) Hitesh Goel (2020115003):

- Coded the bitcoin.h file (all basic structures and function declarations) along with Ishanya.
- Implemented 4 major functions:

a) createBlock() :
Creates a block to hold 50 transactions
maintains a blockchain in the form of a linked list.
InitBlock() is called in this function. emptyBlock()
function is used to create the first block in the
blockchain.
b) Attack() :
Randomly generates a number and attacks a block in the blockchain if the block corresponding to that number by modifying it's Nonce.

c) Validate() :
Checks the validity of the blockchain by traversing the chain
and checking if the prev_block_hash matches with the hash value of the previous block.

d) updateBlockArray() :
It is an array that helps us access any block in the blockchain in O(1) time.

- Helped in debugging of the functions.
- Compiled the project report.

3) Ishanya Sethi (2020102014)

- Coded the bitcoin.h file along with Hitesh.(which contains all the structs and function declarations).
- Coded the Hash Function.
- Helped a bit in debugging.

4) Mihir Rawat (2020113005)

- Worked on initializing all variables and data types including the userlist and blockchain in main().

- Wrote the code for the mine function.
- Set up the loop and switch case to let the user choose what action they want to perform.
- Added the printhelp() feature.
- Worked on attack function and validate function.
- Debugged code and resolved errors in attack(), validate(), colors.c, main.c, createblock() and trans.c.

5) Anjali Singh (2020102004)

- Wrote code for color functions in color.c file and the color.h file.
- Made the printhelp() function.
- Worked on unregister() function and transaction() function in the main file (in the input-output statements).
- Helped a bit in debugging the code in main.c and bitcoin.c files.

Data structures used and Complexity of all functions:

1) main():

Time complexity: O(q); where q is the number of queries.

Considering short command lengths.

Firstly, the main function initializes all the global variables and structures so that they can be used in other functions. It also seeds the pseudo random number generator.

Then, it prints the help page to show the available commands.

Inside the gameloop, a switch-case statement is used to reduce the number of string comparisons to a maximum of 1.

Inside each case, the required input is taken and passed onto the corresponding functions. The output is also printed.

2) mine():

time complexity: O(1)

Randomly decides weather the mine was a success or not. If it was a success then the user is rewarded with some bitcoins.

3) printhelp():

time complexity: O(1)

Prints the help page when command `h` or `help` is entered or when an invalid command is entered.

4) User registration:

Data Structures used: Dynamic array, Struct.
Algorithm used and its complexity: hashing is used to store details in an array and its time complexity is of O(1).

5) Transaction:

Data Structures used: Linked list, Struct.
Algorithm used and its complexity: Transaction details of a user is stored in linked list and time complexity to add a transaction in that linked list is O(1) because it added at starting node.

6) Change value of bitcoin:

Data Structures used: Struct.
Algorithm used and its complexity: A struct is used to keep track of all the values that affect value of coin, time complexity is O(k) where k is a constant.

7) Finduser()

No such data structure is used here.And time complexity of this function is O (1)

8) void initBlockArray() :

The data structure used is a frequency table.
Time complexity = O(n) ; n = 51.

9) void updateBlockArray(Block *Bl):

It adds the value of nonce and pointer to the block corresponding to it's index.
Time complexity: Function executes in constant time.

10) Block createBlock(Transact T, int block_num):

The data structure used is a linked list. We amake use of Hash() function to calculate hash value of the block.
The **initBlock()** function is called in createBlock() to update the values of the block.
Time complexity: It executes in O(1) time.
There is also the **emptyBlock()** function which creates the first block in the blockchain. It also executes in constant time.

11) int Attack():

We use the frequency table PtrBlock[].
Time complexity:The function executes in constant time.

12) int Validate():

We traverse the linked list of blocks to check the validity of blockchain.
Time complexity = O(n*q) time, where 'n = length of blockchain (max = 50)' and 'q = 500'.
It changes the value of Nonce to modify the hash value, so that the blockchain becomes valid again.

13) ElementType Hash(Block B):

It takes the Block variables (previous block hash , nonce , block number , sender's transaction id , receiver's transaction id , transaction amount) and returns a key . It is designed to

reduce the number of collisions.

The data structure used is a doubly linked list (Block is a typedef of struct Blockchain pointer).
The hash value is being calculated in a constant time.The while loop will run for 50 transactions max resulting the hash_value to be calculated in constant time.

14) All color functions:

(Black(bold/standard), White, Red, Cyan, Blue, Green, Yellow and the reset function)
Time Complexity = O(1) since the function contains printf statements.
Reset function is used to change the color to default/ the color which was being used before.