COS 445 - Strategy Design 5

Due online Monday, April 28th at 11:59 pm

Instructions:

- **You may not take late days on the Strategy Designs**. If it helps, think of the Strategy Designs as being due on Friday, except we have given everyone three free late days.
- You should aim to work in a team of two, but you are allowed to work alone or in a team of three. Your team should submit a *single* writeup, using the team feature on codePost. You should also submit a *single* code solution, using the team feature on TigerFile.
- Your goal in this assignment, and all strategy designs, is to **maximize your absolute payoff**. Your goal is *not* to outperform other submissions. Any justification you provide should explain why your strategy maximizes your absolute payoff (and only this kind of justification will contribute to your writeup score). Your code score is computed solely based on how you perform in comparison to "if your solution were replaced by a course staff submission" (that is, you are not being compared to a strategy that you play against, so it does you no good to harm the strategies you play against).[1]
- You are allowed to engage with other teams over Ed or in person (but this is neither encouraged nor discouraged). If this is part of your strategy, you should discuss what you did and why you did it in your writeup. You are allowed to coordinate with other teams, or trick other teams. You are *not* allowed to promise other teams favors (e.g. monetary rewards) or threaten punishment outside the scope of this assignment. For example, you are allowed to promise "if your code does X, our code will do Y." You are not allowed to promise "if your code does X, I will buy you a cookie." If this is part of your strategy, your justification should explain why it will help you *on this assignment*.
- Please reference the course collaboration policy here.
- Please reference the following document for further detail on how these assignments are evaluated: GradesForStrategy.pdf.
- This assignment is open-ended, **please ask questions on Ed to clarify expectations as needed**.

Reminder!

Please read the instructions at GradesForStrategy.pdf to better understand how the strategy design assignments are graded (which in turn should clarify how to answer the prompts).

[1] This claim is *slightly* inaccurate in order to save runtime while computing code scores. You are free to read the full details **here**, and to ask for clarification on Ed, but I'm comfortable advising that the best way to optimize your code score is to just optimize your own payoff and not to overthink subtleties in precisely how the code scores are computed.

Cheating the System (50 points)

Now bankrupt as the result of a failed startup,[2] your best friend has gone off the grid and resorted to using Bitcoin for all transactions. They barely have enough funds to survive, and as such are hoping to cheat the system by successfully double-spending the same coins. Your friend remembered that you were taking 445, and asked for your help profiting as much as possible.

In this challenge you'll play the following game:

Setup:

- One basic unit is a *spend* operation. All spend operations in this game will be from a player to a central bank. You can think of a spend operation as trying to "cash out" some number of BTC owned by that player from the bank.
  - That is, every spend operation has only two arguments, a player i, and a quantity, x. This denotes that the player is trying to trade x BTC from a ledger for x USD.
- The second basic unit is a *block*, B. In this game, a block contains the following arguments:
  - A round $r(B)$ that B was created in.
  - A player $i(B)$ which is the ID of the player who created block B.
  - A pointer to a previous block, $P(B)$.
  - A height, $h(B)$, defined to be $h(P(B)) + 1$.
  - A set of spends, $S(B)$.
  - A *bribe*, $b(B)$. This is a payment from the miner $i(B)$ to any block B' with $P(B') = B$.
  - Player i's *balance*, $c_i(B)$, in the ledger defined by block B is computed as follows.[3]
    - Every player i has an initial balance of 100 BTC.
    - For every spend(i, x) operation in the ledger defined by B, player i's balance decreases by x (that is, all spent BTC are moved from player i to the bank).
    - For every block created by player i in the ledger defined by B, player i's balance increases by 12.5 (that is, they get a block reward of 12.5 for every block they create).[4]
    - For every block B' created by player i in the ledger defined by $P(B)$, player i's balance decreases by $b(B')$ (that is, they lose b BTC because they are paying this bribe).[5]
    - For every block B' created by player i in the ledger, player i's balance increases by $b(P(B'))$ (that is, they gain b BTC because they are paid a bribe).[6]

[2] The advertising went great, but unfortunately there isn't a thick market for gerrymandering consulting outside of census years.

[3] Recall that the ledger defined by block B is the set of spends in B or any blocks B' along the path from B to the root.

[4] This assignment was first created in 2019, when Bitcoin's block reward was 12.5 BTC.

[5] Note the precise choice of which bribes have been processed in the balance $c_i(B)$: all bribes in the ledger defined by $P(B)$ are processed (because we know who receives the bribe already), but the bribe in B itself is *not* yet processed to compute balances in B (because the ledger defined by B does not know who receives it).

[6] If $P(B') = \perp$, then $b(\perp) = 0$.

- At all times, we will refer to *the* longest-chain as the block of greatest height, tie-breaking in favor of blocks created in earlier timesteps (i.e. the block of maximum $h(B)$, tie-breaking in favor of smallest $r(B)$). We will say a block B is *the unique* longest chain if there is no other block of height $\geq h(B)$.

Mining:

- Every round, a uniformly random player i will be selected to mine. That player will create a new block, B, and it will be broadcast to all other miners.
- That player will choose a previous block $P(B)$ (see `getBlockToMine()`).
- That player will also choose a bribe $b(B)$ to offer (see `getAmountToBribe()`). If $b(B) > c_i(P(B))$, then the bribe will be reset to 0 (if you try to bribe more than your balance, your bribe will be updated to 0 — also note that you cannot bribe using the block reward or bribes you would earn this round, per definition of $c_i(P(B))$).
- Both of the above decisions will be made *without* knowing what spend operations are being attempted this round by other players.
- The block will always include all spends created during that round. If any spends try to overdraw their balance, they will be adjusted to draw exactly their balance. That is, $S(B)$ will be set by the environment, not by the miner. The height and balances are computed automatically, and also need not be set by the miner.
- **At all times, every player will also be told the number of rounds until they are next selected to mine**. When this number is zero, the player is mining during this round. If a player is never mining again, this number is -1.
- To be clear, the only two choices you have to make as a miner is where to mine and how much to bribe. You make this decision without knowing what spends will be made this round.

Spending:

- Every round, every player can generate a single spend transaction, by reporting a number $x \geq 0$ (see `getAmountToSpend()`). This decision will be made without knowing the decisions of the miner during this round (unless you are that miner).
- This spend will be included in whatever block is mined this round. If the player does not have enough remaining balance to spend the full x, x will be updated to the remaining balance. To be extra clear about what is the remaining balance: if you are not mining this round, and the miner creates block B with $P(B) = B'$, then your remaining balance at the start of the round is just $c_i(B')$, and you can spend up to this much.[7] If you *are* mining this round, then your remaining balance at the start of the round is again $c_i(B')$. You must reserve $b(B)$ for your bribe, so the maximum you can spend is $c_i(B') - b(B)$ (that is, you cannot spend your new block reward or received bribe in block B itself). Again, if you submit a spend more than this, it will just be decreased to $c_i(B') - b(B)$.

[7] Note that if you do not know $c_i(B')$, the spend will be adjusted to the maximum of your balance and your desired spend, so the transaction will not fizzle just because you aren't sure where the block is being created.

- **Important:** A spend placed in block B is *executed* as soon as there is some chain containing B which is *the unique* longest chain. That is, the bank will not actually give you your USD for a spend included in block B until it is in a unique longest chain. But once this happens, you have your USD and it does not matter what happens afterwards.[8]

Payoffs:

- The game will last for $T = 10000$ rounds. Let B denote *the* longest chain after round T.
- At the end of T rounds, player i will get payoff x for every *executed* spend of size x.
- You will also get payoff $c_i(B)$. That is, the bank will cash out all remaining BTC on *the* longest chain.

You strategy should implement the four functions below. The first three are your strategy. The fourth is a helper function, which you are free to copy/paste from the sample strategies.

Code it up according to the specifications below, and write a brief justification. You will implement the Miner interface provided in `Miner.java`, which requires the following methods, as documented in `Miner.java`.

- `public double getAmountToBribe()` returns your bribe for the block you mine (only applicable if you're selected to mine this round).
- `public double getAmountToSpend()` returns your spend for this round.
- `public BlockChain getBlockToMine()` returns the block you wish to mine on top of (only applicable if you're selected to mine this round).
- `public void refreshNetwork(BlockChain[] spendableBlocks, int myIndex, int roundsRemaining)` lets you know which blocks are available to mine on top of, what your mining index is (to get your stake at a given block), and how many rounds until you're chosen to mine (-1 if you're never chosen to mine again).

Additionally, we've provided the usual construction of testing code: `Network.java` and a makefile to run it against varied strategies. We've also provided the following sample strategies:

- `Miner_longest`: Mines on the longest chain, tie-breaking in favor of the earliest chain, and spends a small amount of their stake from the longest chain every turn.
- `Miner_bribable`: Mines on the chain with the biggest bribe, tie-breaking in favor of the longest chain.
- `Miner_aggressive`: Mines on the second longest chain, tie-breaking in favor of the chain with the most stake, and attempts large double spends when eligible to mine next turn.

Note: Some of these strategies contain snippets of code you may find useful, for example, choosing the longest chain to mine on top of, etc. You are allowed to copy this code.

Your file must follow the naming convention `Miner_netID1.java`, where `netID1` is the Net ID of the primary submitter. All advice from previous homeworks applies again to this homework, especially that you should consider the performance of your strategy against the strategies and distribution thereof which you expect your peers to play.

[8] For example, if B is built on top of any existing longest-chain, B will immediately become the unique longest chain, and all spends are immediately executed. If B is not built on top of an existing longest chain, these spends may get executed later, but are not immediately executed.

Your writeup should provide an overview of the main ideas in your code (remember that we also have your code — so you don't need to provide pseudocode or a step-by-step description of your algorithm), and justify why you think it will perform well, in addition to concrete answers to parts **a** and **b**.

Penalties may be given for code which does not compile, throws exceptions, or violates assertions. Remember to test your code with settings besides the default settings of our testing environment. Extra credit may be awarded for reporting substantive bugs in our testing code.

Also submit a single PDF file, containing answers to the following three prompts. Recall that your grade for part c is the maximum of your grade on the writeup and your grade for your strategy's performance (any penalties are subtracted after taking the maximum).

Part a (10 points)

Evaluate the payoff for player one, assuming the game ends after $T = 5$ rounds when the following sequence of events occur. Recall that each player starts with 100 BTC, and that the block reward is 12.5 per block. Provide a brief justification for any calculations.

1. Round One:
   - Player one is selected to mine, points to the genesis/empty/null block, and includes a bribe of 5. Call this block B_1
   - Player one includes a spend of 20.
2. Round Two:
   - Player two is selected to mine, points to B_1, and includes a bribe of 0. Call this B_2.
   - Player one includes a spend of 40.
3. Round Three:
   - Player three is selected to mine, points to B_1, and includes a bribe of 3. Call this B_3.
   - Player one includes a spend of 10.
4. Round Four:
   - Player two is selected to mine, points to B_3, and includes a bribe of 0. Call this B_4.
   - Player one includes a spend of 20.
5. Round Five:
   - Player one is selected to mine, points to B_2, and includes a bribe of 0. Call this B_5.
   - Player one includes a spend of 25.

Part b (10 points)

Assume that all other players are using the following strategy:

- Always mine on top of a block of greatest height. Tie-break first in favor of any block which offers a bribe of at least 5 more than any other block of this height, then in favor of the earliest-announced block.

- Always set bribes to zero, always spend 0.

What is your best response to this strategy? Briefly describe why it is your best response. For simplicity of exposition, you may assume that you are never selected to mine in back-to-back rounds.

Part c (15 points)

Provide a brief justification for your strategy. Focus on convincing the grader that it is a good strategy, by explaining the main ideas and why you chose this strategy. You should aim to keep this under one page. This will not be strictly enforced, but the grader may choose not to read beyond one page. You should not think of this merely as a documentation explaining only *what* your code does. Instead, try to imagine that it's purpose is to convince your friend *why* they should adopt your strategy.

Note: For all Strategy Designs, it is a good idea to explicitly reason about how you believe your classmates will behave. For this Strategy Design in particular (because your utility is heavily dependent on what your classmates do), it may be a better-than-usual idea to explicitly reason about how you believe your classmates will behave. Note that you are allowed/encouraged to discuss your strategy publicly/privately with other teams (but you should still **not** share code).

Part d (15 points)

Your friend decides to turn your brilliant money machine into a startup, recruiting some recent Princeton grads to start Tiger Spend and significantly scaling up the operation. The Bank quickly catches wind and cancels their Bitcoin operation, but not before Tiger Spend nets a cool $5.76 billion. The Bank appeals to the Bitcoin community: *clearly* the Bitcoin protocol did not function as The Bank expected, and Tiger Spend's behavior clearly constitutes an exploit. Moreover, it's quite easy to undo the exploit: simply ignore the longest-chain where the coins were double-spent, and everyone go back to mining on top of the chain where the coins were originally spent. As such, The Bank formally asks the Bitcoin community to undo the exploit.

People in the community disagree loudly about whether they should do this for The Bank. One group thinks Bitcoin should stick to the protocol verbatim, always. Their argument is often referred to as "code is law," and they think it's a core principle that no subjective decision-making enters the picture. The other group takes what is called the "social consensus" view, which posits the core principle of blockchain is really that decisions reflect the widespread agreement among network participants and so they should be able to override protocol with sufficient agreement.

1. Should the Bitcoin community *ever* undo a double-spend? If so, should they undo *this* double-spend? What if the double-spend were carried out by human rights activists (who use cryptocurrency to circumvent dictators' authoritarian control over the local money supply) instead of Tiger Spend? Make ethical arguments, building on claims made by the two opposed groups.

2. Write down the best argument that could be made for the group that thinks "code is law," even if the double-spend were instead carried out by North Korea's Lazarus group (who use cryptocurrency to circumvent international sanctions). Again, make ethical arguments.

3. Let's say the "social consensus" view prevailed on this chain, and the group decided to use "majority rule" to settle the question of whether to undo the exploit.[9] Would majority rule be best for Tiger Spend, the human rights activists, or the Lazarus group? Give one suggestion of how your friend should change their strategy from part C if they know majority rule would be used to make the final decision.

You should also aim to keep the total length of your entire answer to part d under one page.[10]

[9] "Majority rule" is in quotes because "majority of what" is not obvious. In your answer, you can pick anything reasonable (i.e. "majority of computational power," "majority of Bitcoin owners," "majority of loud voices on Bitcoin forums," etc.).

[10] Hopefully this also suggests the appropriate amount of justification – your justification should be *clear*, include *concrete statements* and be *comparably rigorous to part c*, but ***does not*** *need to be as* ***thorough*** *as part c*.