Mo Felonda

577 HW6

Algorithm Design

Input: An array A[1...,n] in which each element represents that day's BDC to USD exchange rate (total of n days), the initial dollars k, USD to BDC transaction fee d, and BDC to USD transaction fee b.

Output: The maximum amount of dollars you can have at the end of the n^{th} day maxDollars.

```
procedure MAX-INVESTMENT(A[], k,d,b)
1. maxDollars ← k
2. badgercoin ← 0
3. for i=1...n do
4.     currentTotal ← maxDollars
5.     maxDollars ← max(currentTotal, (badgercoin - b) * A[i])
6.     badgercoin ← max(badgercoin, (currentTotal - d) ÷ A[i])
7. return maxDollars
```

Proof: Base Case: n=1, or there is only one day to exchange currency. Since there is only one day, the maximum number of dollars obtained cannot exceed the initial dollars. During the first iteration of the for loop, line 5 will always find the max to be currentTotal which is at this point equal to maxDollars. Thus, the original value of k is returned, which is correct since it is the max. Logically we can understand that there is no reason to trade with only one day as the only possibility is ending up with more Badgercoin and less dollars.

Induction Hypothesis: Assume that for $n \geq 1$, maxDollars and badgercoin (lines 5 and 6) are calculated correctly.

Inductive Step: For each day in which trading is possible, there are three possible actions that can be taken (do nothing, BDC→USD, USD→BDC). When $i \leq n$ we enter the for loop and compute the maximum between the current maximum and the amount that could be made by selling our Badgercoin, both of which we know to be correct by our induction hypothesis. Thus, the correct maximum profit is calculated on the i^{th} day.

∴ we know that, by extension, the correct maximum dollars is returned on the n^{th} day. ■

Mo Felonda

577 HW6 Part II

Correctness: We know that the maximum dollars obtained is returned on the n^{th} day. Consider that in an optimal solution, the dollars obtained is less than our maximum. We know that logically, more money could have been obtained if on a given day more dollars were obtainable by trading Badgercoin or the current amount of dollars was higher with no trading. In other words, on at least one day, the correct maximum was not chosen. Therefore, by contradiction, MAX-INVESTMENT is an optimal solution for returning the max dollars obtainable by the n^{th} day. □

Space Complexity: As we only need currentTotal, maxDollars, and badgercoin to compute the max dollars obtainable, we only need a constant amount of memory space. Thus, the space complexity is $O(1)$.

Time Complexity: Throughout our procedure, we only: do assignments (constant time), compute maximums (constant time) and iterate through our array from 1 to n ($O(n)$). Thus, the time complexity is $O(n)$.