EECS 336 Homework 1

Devon Buckingham

January 18, 2018

Problem 2

Correctness

We begin with three lemmas to prove the correctness of the algorithm.

Lemma 1. *After the k^{th} iteration of the for loop in algorithm* $\boldsymbol{A}$*,* $1 \leq k \leq n$*, the variable S gives the cumulative sum of all trades up to and including trade k (i.e., $\sum_{i=1}^{k} a_i$).*

Proof. We prove by induction.

Base Case: Before the first iteration of the for loop, S is set to 0. The first iteration sets S to $S + a_1 = 0 + a_1 = a_1$.

Inductive step: Assume the lemma holds for $k = r$. Denote the value of S at the end of iteration r as S_r (i.e., $\sum_{i=1}^{r} a_i$, by the inductive hypothesis). Iteration $r + 1$ sets $S = S_r + a_{r+1} = \sum_{i=1}^{r} a_i + a_{r+1} = \sum_{i=1}^{r+1} a_i$. □

Lemma 2. *After the k^{th} iteration of the for loop in algorithm* $\boldsymbol{A}$*,* $1 \leq k \leq n$*, the variable MaxS contains the largest value of S yet observed (i.e., the maximal sum of subsequence terms, over the empty subsequence and all subsequences of consecutive trades that start at trade 1 and go up to some $j \leq k$.)*

Proof. We prove by induction.

Base Case: Before the first iteration of the for loop, $MaxS$ is set to zero. During the first iteration (but prior to updating $MaxS$), S is updated from 0 to a_1, as shown in Lemma 1. Then, $MaxS$ is set to the minimum of the current $MaxS$ (0) and the updated value of S (a_1). So, if a_1 is positive, MaxS is set to a_1, which is indeed the maximal subsequence sum/maximal value of S observed so far. If a_1 is not positive, then $MaxS$ remains at zero, which is also the maximal subsequence sum observed so far (the sum of the empty sequence), or equivalently, the maximal value of S observed so far (the value S held before entering the for loop).

Inductive step: Assume lemma 2 holds for $k = r$. At iteration $r + 1$, S will be set to $\sum_{i=1}^{r+1} a_i$, by lemma 1. $MaxS$ will then be set to the maximum of this new S and the current $MaxS$, which by the inductive hypothesis is equal to the maximal observed value of S up to iteration r. So, by taking the maximum of these two numbers, $MaxS$ will then be set to the new maximal value of S, up to iteration $r + 1$. □

Lemma 3. *Let S_q be the value of S after iteration q and $MaxS_q$ be the value of $MaxS$ after iteration q, or 0 if $q = 0$. After iteration k, $MinValue$ will be equal to $\min_q(S_q - MaxS_{q-1})$, $q \leq k$.*

Proof. We prove by induction.

Base Case: In the first iteration, prior to the assignment to $MinValue$, S is updated to a_1 as described in lemma 1. This quantity corresponds to S_1. At the assignment to $MinValue$, $MaxS$ has not yet been updated, so is still equal to zero. This quantity corresponds to $MaxS_0$. The difference of these two quantities is a_1. This is already the current value of $MinValue$, so $MinValue$ will be $\min(a_1, a_!) = a_1 = S_1 - MaxS_0$. So, the lemma holds in the base case.

Inductive Step: Assume the lemma holds for $k = r$. In iteration $r + 1$, the algorithm will set $MinValue$ equal to the minimum of its current value (which is correct for iteration r by the inductive assumption) and $S_{r+1} - MaxS_r = S_{r+1} - MaxS_{(r+1)-1}$. So, $MinValue$ will be set to the new minimal value of $(S_q - MaxS_{q-1})$, $q \leq (r + 1)$. □

We now investigate the quantity that the algorithm is intended to find. This value is given as

$$\min_{s \leq t} \sum_{i=s}^{t} a_i$$

where s and t are integers from 1 to n. This expression can be rewritten in the following manner:

$$\min_{s \leq t} \sum_{i=s}^{t} a_i \tag{1}$$

$$\min_{t} \min_{s \leq t} \sum_{i=s}^{t} a_i \tag{2}$$

$$\min_{t} \min_{s \leq t} (\sum_{i=1}^{t} a_i - \sum_{i=1}^{s-1} a_i) \tag{3}$$

$$\min_{t} (\sum_{i=1}^{t} a_i - \max_{s \leq t} \sum_{i=1}^{s-1} a_i) \tag{4}$$

Now, consider only the portion of expression (4) that is in parens, and assume t is fixed. Notice that the first summation corresponds to the variable S after iteration t (which the algorithm computes correctly, by lemma 1). Similarly, the second summation corresponds to the value of $MaxS$ after iteration $t-1$, which, by lemma 2, is also computed correctly. So, the entire quantity in parens corresponds to $S - MaxS$, as is computed in iteration t of the algorithm. By lemma 3, the algorithm keeps the minimum value this quantity has obtained in $MinValue$. So, at termination, $MinValue$ will be equal to expression (4).

Running Time

The operations outside the for loop are constant-time assignments. The operations inside the for loop are constant-time arithmetic and assignments. The for loop iterates from 1 to n, performing these constant-time operations. So, the algorithm as a whole is $O(n)$.