Exercise 3.4 *Investing in the stock market* **(2 Points)**.

You have 100 CHF and you are considering investing it in the stock market. You heard from your friends that a particular stock is promising but you are not sure. You decided to analyze the performance of this stock during the recent past.

You have a friend that had invested in this stock for n consecutive days in the recent past. You asked your friend about how much money she invested in this stock and how much her total investment worth was progressing every day. You gathered this information in two arrays $A = (A_1, \ldots, A_n)$ and $I = (I_1, \ldots, I_n)$. Here, A_i represents the additional amount of money that your friend invested on the i-th day. More precisely, if your friend bought on the i-th day then A_i would be positive, and if she sold on the i-th day, then A_i would be negative. The value of I_i is the total worth of her investment in the stock on the i-th day. Here is an illustrating example with $n = 4$:

i	1	2	3	4
A_i	150	150	200	-100
I_i	150	350	500	400

In the above example, your friend had invested for 4 days. She invested $A_1 = 150$ CHF on the first day. Since she had not invested anything prior to that day, we must have $I_1 = A_1 = 150$ CHF. On the second day, she invested an additional $A_2 = 150$ CHF and the total worth of her investment was $I_2 = 350$ CHF. This means that right before buying the additional 150 CHF, her total investment in the stock was worth $350 - 150 = 200$ CHF, which is an indication of an increase in the price of the stock from the first to the second day. On the last day of investment ($i = 4$), she sold 100 CHF worth of her investment, and the remaining total worth of her investment after the selling operation was 400 CHF.

a) Let $1 \leq i \leq n - 1$. Suppose that you had invested 1 CHF on the i-th day and sold the entire investment on the $(i+1)$-th day. Show that you would have got $\frac{I_{i+1} - A_{i+1}}{I_i}$ CHF in return.

Solution: On day i, your friend had a total investment of I_i. On the other hand, on day $i+1$, right before the additional investment of A_{i+1}, the total investment was worth $I_{i+1} - A_{i+1}$. Therefore, if you had invested 1 CHF on day i, this investment would be worth $\frac{I_{i+1} - A_{i+1}}{I_i} \times 1\text{ CHF} = \frac{I_{i+1} - A_{i+1}}{I_i}$ CHF on day $i+1$.

↳ In I_{i+1} ist auch A_{i+1} enthalten ⇒ Wert des Investments an Tag $i+1$ ohne neu investiertes/abgehobenes Geld: $I_{i+1} - A_{i+1}$

- Wie können wir die Performance eines Investments über einen Tag messen?

  ↳ $\frac{\text{Wert Tag 2}}{\text{Wert Tag 1}} = \frac{I_{i+1} - A_{i+1}}{I_i}$

- Da nur 1 CHF investiert: Mit 1 CHF multiplizieren.

b) Let $1 \leq i \leq j \leq n$. Suppose that you had invested 100 CHF on the i-th day and sold the entire investment on the j-th day. Show that your profit is equal to

$$100 \cdot \prod_{k=i}^{j-1} \frac{I_{k+1} - A_{k+1}}{I_k} - 100.$$

Note that in the above equation, we adopt the convention that if $i = j$, then $\prod_{k=i}^{j-1} \frac{I_{k+1} - A_{k+1}}{I_k} = 1$.

Von a) wissen wir noch, dass unser Return of Invest bei einem Tag Halte-Dauer bei $\left(\frac{I_{i+1} - A_{i+1}}{I_i}\right) \cdot$ (INVESTMENT) CHF liegt.

Anstatt den Verlauf über $(j-i)$ Tage anzugucken können wir also auch einfach Tag 1 zu Tag 2, Tag 2 zu Tag 3, ... betrachten und die Tages-Änderungen multiplizieren.

Insgesamt gibt es damit ein Return of Invest von:

$$\left(\prod_{k=i}^{j-1} \left(\frac{I_{k+1} - A_{k+1}}{I_k}\right)\right) \cdot 100$$

Da wir uns aber für den Profit interessieren, müssen wir noch unser Anfangsinvest abziehen:

$$100 \cdot \prod_{k=i}^{j-1} \left(\frac{I_{k+1} - A_{k+1}}{I_k}\right) - 100$$

You are interested in finding the maximum profit that you could have made in a single buy-sell operation by investing 100 CHF. Here, you would buy 100 CHF worth of the stock on some day i where $1 \le i \le n$ and then sell the entire investment another day j where $i \le j \le n$.

We first assume that all $A_1, \dots, A_n$ and $I_1, \dots, I_n$ are positive, and that $I_k > A_k$ for every k.

c) Describe how you can use the maximum subarray-sum algorithm that you learned in class in order to devise an algorithm that computes the maximum profit in $O(n)$ time. You can assume that arithmetic operations (such as addition, subtraction, multiplication and division) as well as logarithms and exponentials are elementary. This means that the computation of log and exp take one unit of time each.

Hint: *You can use the fact that the logarithm is a strictly increasing function that turns products into sums.*

Aus b) wissen wir. Max. Profit $:= \max_{1\le i\le j\le n} \left\{ 100 \cdot \prod_{k=i}^{j-1} \left(\frac{I_{k+1} - A_{k+1}}{I_k} \right) - 100 \right\}$

$$= 100 \cdot \left(\max_{1\le i\le j\le n} \prod_{k=i}^{j-1} \left(\frac{I_{k+1} - A_{k+1}}{I_k} \right) \right) - 100$$

$\Rightarrow$ Also müssen wir nur $\max_{1\le i\le j\le n} \prod_{k=i}^{j-1} \left(\frac{I_{k+1} - A_{k+1}}{I_k} \right)$ berechnen, um Profit zu berechnen, da Rest unveränderlich

Trick: Wenn wir neues Array G konstruieren, können wir bekannten Algo verwenden (MSS – Ist aber nur für ein Array def.)

- Wir konstruieren $G(1, \dots, n-1)$ mit $G_k = \frac{I_{k+1} - A_{k+1}}{I_k}$
- Aus Vorgaben folgt: $\forall k \in \{1, \dots, n-1\}: G_k > 0$

$\rightarrow$ Also nur $\max_{1\le i\le j\le n} \prod_{k=i}^{j-1} G_k$ berechnen.

Problem: Tagesänderung ist Multiplikation, wir kennen aber nur Algo. für Addition

Trick: $n = e^{\ln(n)}$ und e, ln sind streng mon. wachsend

$$\max_{1\le i\le j\le n} \prod_{k=i}^{j-1} G_k = \exp\left(\ln\left(\max_{1\le i\le j\le n} \prod_{k=i}^{j-1} G_k\right)\right)$$

$$= \exp\left(\max_{1\le i\le j\le n} \ln\left(\prod_{k=i}^{j-1} G_k\right)\right)$$

$\log(a \cdot b \cdot \ldots \cdot c) = \log(a) + \log(b) + \ldots + \log(c)$

$$= \exp\left(\max_{1\le i\le j\le n} \sum_{k=i}^{j-1} \ln(G_k)\right)$$

Array of logs of G_k $\Rightarrow$ Can use MSS

Algorithm 10 Computation of Maximum Profit

procedure MAXPROFIT(A, I)
  for $1 \leq k \leq n-1$ **do**
    $G_k \leftarrow (I_{k+1} - A_{k+1})/I_k$
  MaxProdG $\leftarrow$ MaxSubarrayProduct(G)
  MaxProfit $\leftarrow$ 100 $\cdot$ MaxProdG $-$ 100
  return MaxProfit

procedure MAXSUBARRAYPRODUCT(G)
  for $1 \leq k \leq n-1$ **do**
    $L_k \leftarrow \log(G_k)$
  MaxSumL $\leftarrow$ MaxSubarraySum(L)
  MaxProdG $\leftarrow$ exp(MaxSumL)
  return G

d) Now assume that the logarithm and exponential operations are expensive so that we would like to avoid using them. Explain how you can modify the maximum subarray sum algorithm in order to solve the problem using only elementary arithmetic operations such as addition, subtraction, multiplication and division. The running time of the algorithm should remain in $O(n)$.

Use same Idea as MSS but with product and ≥ 1

Algorithm 11 Computation of maximum subarray-product

procedure MAXSUBARRAYPRODUCT(G)
  $P \leftarrow G_1$
  MaxProd $\leftarrow \max\{1, P\}$
  for $2 \leq j \leq n-1$ **do**
    $P \leftarrow \max\{G_j, G_j \cdot P\}$
    MaxProd $\leftarrow \max\{\text{MaxProd}, P\}$
  return MaxProd

It is easy to see that the above algorithm runs in $O(n)$ time.