Documentation

Lucas Guesser Targino da Silva

February 9, 2023

Contents

Todo list

Chapter 1

Share Market

1.1 Basic Definitions

Definition 1 (Natural)**.** Given $v \in \mathbb{N}$, we define:

$$\mathbb{N}_v = \{n \in \mathbb{N} : n \geqslant v\} \tag{1.1}$$

Definition 2 (Range)**.** Given $n \in \mathbb{N}_2$, we define the **Range of** n as:

$$[[n]] = \{0, \dots, n-1\} = \{i \in \mathbb{N} : i < n\} \tag{1.2}$$

Definition 3 (Square)**.** Given $n \in \mathbb{N}_2$, we define the **Square of** n as:

$$sq(n) = [[n]] \times [[n]] \tag{1.3}$$

1.2 Problem Definitions

You are given an array in which the ith element is the price of a given stock on the day i. You are permitted to complete at most 1 transaction (i.e. buy once and sell once). What is the maximum profit you can gain?

Notice that you cannot sell a stock before buying it.

1.2.1 Definitions

Definition 4 (Price Tuple)**.** Given $n \in \mathbb{N}_2$, a **Price Tuple** p is a positive real tuple with n elements:

$$p = \langle p_0, \dots, p_{n-1} \rangle \in \mathbb{R}^n_+ \tag{1.4}$$

Definition 5 (Operation Pair)**.** Given $n \in \mathbb{N}_2$, an **Operation Pair** ω is a pair:

$$\omega = \langle b\left(\omega\right), s\left(\omega\right) \rangle \in sq(n) \quad , b < s \tag{1.5}$$

when the context is clear enough, we will simply write $\omega = \langle b\left(\omega\right), s\left(\omega\right) \rangle = \langle b, s \rangle$.

Definition 6 (Ascending Operation Pair)**.** Given $n \in \mathbb{N}_2$ and a Price Tuple p, an Operation Pair $\omega = \langle b, s \rangle$ is said to be **Ascending** when:

$$p_b \leqslant p_{b+1} \leqslant \cdots \leqslant p_{s-1} \leqslant p_s \tag{1.6}$$

Definition 7 (Maximal Operation Pair)**.** Given $n \in \mathbb{N}_2$ and a Price Tuple p, we say that an Ascending Operation Pair $\omega = \langle b, s \rangle$ is a **Maximal Operation Pair**, or simply that ω is **Maximal**, when it is Ascending and it satisfies the conditions:

$$p_s - p_b \geqslant p_{s+1} - p_b \quad , \text{if } s+1 < n \tag{1.7}$$
$$p_s - p_b \geqslant p_s - p_{b-1} \quad , \text{if } b-1 > 0 \tag{1.8}$$

Theorem 1. *Given $n \in \mathbb{N}_2$ and a Price Tuple p, an Operation Pair $\omega = \langle b, s \rangle$ is Maximal if and only if the following conditions are satisfied:*

$$p_s \geqslant p_{s+1} \quad , \textit{if } s+1 < n \tag{1.9}$$
$$p_b \leqslant p_{b-1} \quad , \textit{if } b-1 > 0 \tag{1.10}$$

Proof. It comes directly from the inequalities of the Definition 7. □

Definition 8 (Independent Operation Pairs)**.** Given two Operation Pairs ω, ω', we say that ω, ω' are **Independent** when:

$$s(\omega) < b(\omega') \vee s(\omega') < b(\omega) \tag{1.11}$$

moreover, given a set of Operation Pairs $S = \{\omega_0, \ldots, \omega_{m-1}\}$, we say that S is independent when it satisfies:

$$\forall \omega \forall \omega' \left(\omega \in S \wedge \omega' \in S \to \omega, \omega' \text{ are independent}\right) \tag{1.12}$$

i.e. all Operation Pairs $\omega, \omega' \in S$ are pairwise Independent.

Definition 9 (Operation Set)**.** Given $n \in \mathbb{N}_2$, an **Operation Set** $S \subseteq sq(n)$ is a set which satisfies:

1. $\forall \omega \left(\omega \in S \to \omega \text{ is an Operation Pair}\right)$
2. S is independent

Moreover, we denote the set of all possible Operations Set by:

$$\Omega_n = \{S \subseteq sq(n) : S \text{ is an Operation Set}\} \tag{1.13}$$

Definition 10. Given an $n \in \mathbb{N}_2$, a Price Tuple p, and an Operation Set $S \in \Omega_n$, the price $\rho(S)$ of S is:

$$\rho(S) = \sum_{\omega \in S} p_{s(\omega)} - p_{b(\omega)} \tag{1.14}$$

Lemma 1. *Given an $n \in \mathbb{N}_2$ and a Price Tuple p, if a Operation Set $S^* \in \Omega_n$ is has optimal price, then all Operation Pairs of S^* are Maximal, i.e.:*

$$\rho(S^*) = \max_{S \in \Omega_n} \left[\rho(S)\right] \to \forall \omega \left(\omega \in S^* \to \omega \textit{ is Maximal}\right) \tag{1.15}$$

Proof. Suppose $S^* \in \Omega_n$ is a Operation Set with optimal price. Suppose by contradiction that $\exists \omega \left(\omega \in S^* \wedge \omega \text{ is not Maximal}\right)$, and let $\omega = \langle b, s \rangle$ be such Operation Pair. By Theorem 1, one of the following cases must be true:

1. $p_s < p_{s+1} \quad$, if $s+1 < n$
2. $p_b > p_{b-1} \quad$, if $b-1 > 0$

Case 1

Suppose that $p_s < p_{s+1}$, $s+1 < n$.

Case 1.1

Suppose in addition that $\omega' = \langle b, s+1\rangle$ is independent of all $\omega \in S^* \setminus \{\omega\}$. Let $S' = (S^* \setminus \{\omega\}) \cup \{\omega'\}$. Notice that

$$\rho(\omega) = p_{s(\omega)} - p_{b(\omega)} < p_{s(\omega')} - p_{b(\omega')} = \rho(\omega') \tag{1.16}$$

Therefore $\rho(S^*) < \rho(S')$ (because S^* and S' differ only in the elements above), contradicting the optimality of S^*.

Case 1.2

Suppose in addition that $\exists \omega'(\omega' \in S^* \wedge b(\omega') = s+1)$, and let $\omega' = \langle s+1, s'\rangle$ be such Operation Pair. Let $\omega'' = \langle b, s'\rangle$ and $S'' = (S^* \cup \omega'') \setminus \{\omega, \omega'\}$. Notice that

$$\begin{aligned} \rho(\omega) + \rho(\omega') = \\ (p_{s(\omega)} - p_{b(\omega)}) + (p_{s(\omega')} - p_{b(\omega')}) = \\ (p_s - p_b) + (p_{s'} - p_{s+1}) = \\ (p_{s'} - p_b) + (p_s - p_{s+1}) < \\ p_{s'} - p_b = \\ \rho(\omega'') \end{aligned} \tag{1.17}$$

Therefore $\rho(S^*) < \rho(S'')$ (because S^* and S' differ only in the elements above), contradicting the optimality of S^*.

Case 1 - Conclusion

The hypothesis $p_s < p_{s+1}$, $s+1 < n$ leads to a contradiction. Therefore, if S^* is optimal, then $p_s \geqslant p_{s+1}$, $s+1 < n$, as we wanted to prove.

Case 2

Suppose that $p_b > p_{b-1}, b-1 > 0$. The proof of this case is similar to the proof of the Case 1.2.1, but this uses b and $b-1$ instead of s and $s-1$. $\square$

Definition 11. Given an $n \in \mathbb{N}_2$, let $\omega = \langle b, s\rangle \in sq(n)$ be an Operation Pair. We say that i is included in ω, and denote by $i \triangleright \omega$, when $b \leqslant i \leqslant s$.

Definition 12. Given an $n \in \mathbb{N}_2$ and a Price Tuple p, we say that an Operation Set $S \in \Omega_n$ is Great when:

$$\forall i \big((i \in [[n-1]] \wedge p_i < p_{i+1}) \rightarrow (\exists \omega\, (\omega \in S \wedge i \triangleright \omega \wedge (i+1) \triangleright \omega)) \big) \tag{1.18}$$

i.e. the indices of all ascending pairs of p are included in ω.

Lemma 2. *Given an $n \in \mathbb{N}_2$ and a Price Tuple p, if a Operation Set $S^* \in \Omega_n$ is has optimal price, then S^* is Great.*

Proof. Suppose by contradiction that S^* is not Great, i.e.

$$\exists i \big((i \in [[n-1]] \wedge p_i < p_{i+1}) \wedge (\nexists \omega\, (\omega \in S \wedge i \triangleright \omega \wedge (i+1) \triangleright \omega)) \big) \tag{1.19}$$

and let i be such value.

Case 1

Suppose that $i \triangleright \omega \wedge \neg(i+1 \triangleright \omega)$ for some $\omega \in S^*$. If $i+1 \triangleright \omega'$ for some ω', then join ω and ω' to get an Operation Set with cost greater than S^*. If that is not the case, extend ω to include $i+1$ to get an Operation Set with cost greater than S^*. In all cases, the original hypothesis contradicts the optimality of S^*.

Case 2

Suppose that $\neg(i \triangleright \omega) \wedge i+1 \triangleright \omega$ for some $\omega \in S^*$. This case is similar to the previous one, except that the Operation Pair has to be extended backwards instead of forwards.

Case 3

Suppose that $\forall \omega(\omega \in S^* \rightarrow (\neg(i \triangleright \omega) \wedge \neg(i+1 \triangleright \omega)))$. Create a new Operation Set $S' = S^* \Cup \{\langle i, i+1 \rangle\}$, which has greater cost, contradicting the optimality of S^*.

Conclusion

The cases above cover all possible cases. Therefore, the lemma has been proven by contradiction. □

Lemma 3. *Given an $n \in \mathbb{N}_2$ and a Price Tuple p, if an Operation Set $S^* \in \Omega_n$ satisfy:*

1. $\forall \omega \, (\omega \in S^* \rightarrow \omega$ *is Maximal*$)$
2. S^* *is Great*

then S has optimal price, i.e $\rho(S) = \max\limits_{S' \in \Omega_n} [\rho(S')]$.

It is tiresome to write this proof so I won't.

Theorem 2. *Given an $n \in \mathbb{N}_2$ and a Price Tuple p, an Operation Set $S^* \in \Omega_n$ has optimal price if and only if it satisfies:*

1. $\forall \omega \, (\omega \in S^* \rightarrow \omega$ *is Maximal*$)$
2. S^* *is Great*

Proof. Lemmas 1 and 2 prove the $\Rightarrow$ part. Lemma 3 proves the $\Leftarrow$. □

1.3 Problem Description

Input a Price Tuple p

Output a value $z \in \mathbb{R}$

Goal $\max z$

1.4 Solution - Dynamic Programming

Given that you buy on a day i, while the value does not decrease, you keep it. If it will drop the next day, you sell it.

1.4.1 Initial State

Find the first pair $\langle b, s\rangle$ for which the price increases, i.e. the first pair of consecutive indices for which $p_s - p_b > 0$.

1.4.2 Optimal Substructure

Let:

1. $\langle b_l, s_l\rangle$ be the last operation;
2. p_{s_l} be the price of the last sell;
3. i the index of the current day;
4. p_i the stock price of the current day;
5. p_{i-1} the stock price of the previous day;

Cases:

1. if $(s_l == i-1) \wedge (p_{s_l} \leqslant p_i)$
   (a) replace $\langle b_l, s_l\rangle$ by $\langle b_l, i\rangle$
   (b) rationale: if the stock price is increasing, you keep it;
2. if $(s_l < i-1) \wedge (p_{i-1} \leqslant p_i)$
   (a) add $\langle i-1, i\rangle$
   (b) rationale: if you have no stock and it will increase, you buy and sell it;
3. the others are cases in which the stock price drops, and there is nothing to do;

Complexity: $\mathcal{O}(n)$

1.5 Solution - Simple

Algorithm 1 Simple-Algoruithm

$\Delta p \leftarrow [\langle p_{i+1} - p_i\rangle \ for \ i \in \{0, \ldots, n-2\}]$
$\Delta p_{>} \leftarrow filter(\Delta p, (>= 0))$
$r \leftarrow sum(\Delta p_{>})$
return r

The filter takes care of removing the drops on the price, while the sum of the differences computes the gains.

Complexity: $\mathcal{O}(n)$

Chapter 2

Sum of the Range

2.1 Problem Definition

2.1.1 Input

1. two natural numbers $m, n \in \mathbb{N}$
2. an array of values $v \in \mathbb{R}^n$
3. an set of queries $Q = \{\langle i, j \rangle : i, j < n\}^m$

2.1.2 Output

The output $a : Q \rightarrow \mathbb{R}^m$ is the answer function of all queries Q. The answer $a(q)$ to a query $q = \langle i, j \rangle$ is given by:

$$a(q) = \sum_{k=i}^{j} v[k] \tag{2.1}$$

2.2 Example

$$\begin{aligned}
n &= 6 && (2.2)\\
m &= 3 && (2.3)\\
v &= \langle 1, -2, 3, 10, -8, 0 \rangle && (2.4)\\
q &= \langle \langle 0, 2 \rangle, \langle 1, 4 \rangle, \langle 3, 3 \rangle \rangle && (2.5)\\
a &= \langle 2, 3, 10 \rangle = \langle 1 - 2 + 3, -2 + 3 + 10 - 8, 10 \rangle && (2.6)
\end{aligned}$$

2.3 Solution Naive

Algorithm 2 Naive

Require: $m \in \mathbb{N}, n \in \mathbb{N}, v \in \mathbb{R}^n, Q = \{\langle i, j\rangle : i, j < n\}^m$

```
a = zeros(m)
for k ∈ {0, ..., m − 1} do
    ⟨i, j⟩ ← Q[k]
    a[k] ← sum(⟨v[i], ..., v[j]⟩)
return a
```

2.4 Solution Optimized

Notice that:

$$a(\langle i, j\rangle) = \begin{cases} a(\langle 0, j\rangle) - a(\langle 0, i-1\rangle) & , \text{ if } i > 0 \\ a(\langle 0, j\rangle) & , \text{ if } j = 0 \end{cases} \tag{2.7}$$

The algorithm is then: compute all values $a(\langle 0, j\rangle), \forall j \in \{0, \ldots, n-1\}$ and then answer all queries using the formula above.

Algorithm 3 Opt

Require: $m \in \mathbb{N}, n \in \mathbb{N}, v \in \mathbb{R}^n, Q = \{\langle i, j\rangle : i, j < n\}^m$

```
Δs ← zeros(n + 1)
for i ∈ {0, ..., n − 1} do
    Δs[i + 1] ← Δs[i] + v[i]
a = zeros(m)
for k ∈ {0, ..., m − 1} do
    ⟨i, j⟩ ← Q[k]
    a[k] = Δs[j + 1] − Δs[i]
return a
```

Chapter 3

Longest Increasing Subsequence

3.1 Basic Definitions

Definition 13 (Sequence)**.** A **Sequence** is a function f from the subset $I \subseteq \mathbb{N}$ of the Natural Numbers into a Codomain Cd:

$$f : I \to Cd \tag{3.1}$$

Denote by $\mathcal{S}(I, Cd)$ the set of all sequences of I into Cd:

$$\mathcal{S}(I, Cd) = \{f : I \to Cd\} \tag{3.2}$$

Definition 14 (Successor)**.** Let $I \subseteq \mathbb{N}$ be a set. The successor is a bijective function:

$$\sigma_I : I \backslash \max(I) \to I \backslash \min(I) \tag{3.3}$$

$$i \mapsto \min\{j \in I : j > i\} \tag{3.4}$$

Definition 15 (Increasing Sequence)**.** Given sets $I \subseteq \mathbb{N}$ and Cd, a sequence $f \in \mathcal{S}(I, Cd)$ is said to be **increasing** when the values of the sequence increase, i.e.:

$$f \text{ is an increasing sequence} \quad \leftrightarrow \quad \forall i \left(i \in I \backslash \max(I) \to f(i) \leqslant f(\sigma_I(i))\right) \tag{3.5}$$

Definition 16 (Length of a Sequence)**.** Given sets $I \subseteq \mathbb{N}$ and Cd, and a sequence $f \in \mathcal{S}(I, Cd)$, the length of the sequence f, denoted by $\mathfrak{L}(f)$ is the cardinality (number of elements) of its domain I:

$$\mathfrak{L}(f) = |I| \tag{3.6}$$

Definition 17 (Subsequence)**.** Let $f \in \mathcal{S}(I, Cd)$ be a sequence from $I \subseteq \mathbb{N}$ into a Codomain Cd. A sequence $g \in \mathcal{S}(I', Cd')$ is called a subsequence of f, and denoted by $g \preceq f$, when $I' \subseteq I$ and $Cd' \subseteq Cd$:

$$\forall f \left(f \in \mathcal{S}(I, Cd) \to \forall g \left(g \in \mathcal{S}(I', Cd') \to \left(g \preceq f \leftrightarrow (I' \subseteq I \wedge Cd' \subseteq Cd)\right)\right)\right) \tag{3.7}$$

Moreover, denote by:

1. $\mathfrak{s}(f)$ the set of all subsequences of the sequence f;
2. $\mathfrak{d}(g) \subseteq I$ the domain of a subsequence $g \preceq f$;

Theorem 3. *Every sequence of n elements can be indexed using $\{0, \ldots, n-1\}$.*

Write this theorem ap priately and prove it.

3.1.1 Examples

We will write the sequence using ordered pairs. For example, a sequence v with $I = \{0, 1, 3, 10\}$ and $Cd = \mathbb{R}$ could be:

$$v = \{\langle 0, 4.7\rangle, \langle 1, -8.8\rangle, \langle 3, -5.4\rangle, \langle 10, 2.3\rangle\} \tag{3.8}$$

The elements of the sequence will be written using common function notation:

$$\begin{aligned} v(0) &= 4.7 \\ v(1) &= -8.8 \\ v(3) &= -5.4 \\ v(10) &= 2.3 \end{aligned}$$

In general, we are interested in sequences of n elements for which $I = \{0, \ldots, n-1\}$. In such cases, we will write the sequence simply as a tuple:

$$\begin{aligned} v &= \{\langle 0, 4.7\rangle, \langle 1, -8.8\rangle, \langle 2, -5.4\rangle, \langle 3, 2.3\rangle\} \\ &\equiv \langle 4.7, -8.8, -5.4, 2.3\rangle \end{aligned} \tag{3.9}$$

3.2 Problem Definition

3.2.1 Input

1. a natural number $n \in \mathbb{N}$;

2. A sequence $v \in \mathcal{S}(\{0, \ldots, n-1\}, \mathbb{R})$[1]

3.2.2 Output

Define $\mathcal{I}(v) = \{s \in \mathfrak{s}(v) : s \text{ is increasing}\}$ the set of all increasing subsequences of v. An output is any element $s \in \mathcal{I}(v)$.

3.2.3 Goal

Find the longest increasing subsequence of v:

$$s^* = \underset{s \in \mathcal{I}(v)}{\arg\max}\ \mathfrak{L}(s) \tag{3.10}$$

[1]Notice that here we are indexing using the naturals $\{0, \ldots, n-1\}$ and not a general $I \subseteq \mathbb{N}$. This is because of the Theorem 3.

3.3 Naive Algorithm

Algorithm 4 Naive

Require: $n \in \mathbb{N}, v \in \mathbb{R}^n$

1: $S \leftarrow \text{generate\_all\_subsequences}(v)$
2: $S' \leftarrow \text{filter}(\text{is\_increasing\_sequence}, s)$
3: $s^* \leftarrow \underset{s \in S'}{\arg\max}\ \mathfrak{L}(s)$
4: **return** s^*

3.4 Recursive Algorithm

Define:

1. $B(i)$: the set of indices lower than i for which the correspondent element in the sequence is smaller than $v(i)$. This is going to be used to compose the subproblems.

$$B(i, v) = \{j \in \mathfrak{d}(v) : j < i \ \wedge \ v(j) < v(i)\} \tag{3.11}$$

2. $L(i)$: the longest increasing subsequence ending at the index i, so that its last element is $v(i)$. It can be defined recursively by the Equation 3.12. In it, $\oplus$ is the concatenation of two sequences and when $B = \varnothing$, the output of the $\arg\max$ is an empty sequence.

$$L(i, v) = \underset{j \in B(i,v)}{\arg\max}\ \mathfrak{L}(L(j)) \oplus \langle v(i) \rangle \tag{3.12}$$

Algorithm 5 Recursive

Require: $n \in \mathbb{N}, v \in \mathbb{R}^n$

1: $S \leftarrow \langle L(i) \ : \ i \in \mathfrak{d}(v) \rangle$
2: $s^* \leftarrow \underset{s \in S}{\arg\max}\ \mathfrak{L}(s)$
3: **return** s^*