

# Documentation

Lucas Guesser Targino da Silva

January 12, 2023



# Contents

<b>1</b>	<b>Share Market</b>	<b>5</b>
1.1	Basic Definitions . . . . .	5
1.2	Problem Definitions . . . . .	5
1.2.1	Definitions . . . . .	5
1.3	Problem Description . . . . .	8
1.4	Solution - Dynamic Programming . . . . .	8
1.4.1	Initial State . . . . .	9
1.4.2	Optimal Substructure . . . . .	9
1.5	Solution - Simple . . . . .	9
<b>2</b>	<b>Sum of the Range</b>	<b>11</b>
2.1	Problem Definition . . . . .	11
2.1.1	Input . . . . .	11
2.1.2	Output . . . . .	11
2.2	Example . . . . .	11
2.3	Solution Naive . . . . .	12
2.4	Solution Optimized . . . . .	12



# Chapter 1

## Share Market

### 1.1 Basic Definitions

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

$$\mathbb{N}_v = \{n \in \mathbb{N} : n \geq v\} \quad (1.1)$$

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

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

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

$$sq(n) = [[n]] \times [[n]] \quad (1.3)$$

### 1.2 Problem Definitions

You are given an array in which the  $i$ th 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:

$$p = \langle p_0, \dots, p_{n-1} \rangle \in \mathbb{R}_+^n \quad (1.4)$$

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

$$\omega = \langle b(\omega), s(\omega) \rangle \in sq(n) \quad , b < s \quad (1.5)$$

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

**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:

$$p_b \leq p_{b+1} \leq \dots \leq p_{s-1} \leq p_s \quad (1.6)$$

**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  $\omega$  is **Maximal**, when it is Ascending and it satisfies the conditions:

$$p_s - p_b \geq p_{s+1} - p_b \quad , \text{ if } s + 1 < n \quad (1.7)$$

$$p_s - p_b \geq p_s - p_{b-1} \quad , \text{ if } b - 1 > 0 \quad (1.8)$$

**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:

$$p_s \geq p_{s+1} \quad , \text{ if } s + 1 < n \quad (1.9)$$

$$p_b \leq p_{b-1} \quad , \text{ if } b - 1 > 0 \quad (1.10)$$

*Proof.* It comes directly from the inequalities of the Definition 7.  $\square$

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

$$s(\omega) < b(\omega') \vee s(\omega') < b(\omega) \quad (1.11)$$

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

$$\forall \omega \forall \omega' (\omega \in S \wedge \omega' \in S \rightarrow \omega, \omega' \text{ are independent}) \quad (1.12)$$

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.  $\forall \omega (\omega \in S \rightarrow \omega \text{ is an Operation Pair})$
2.  $S$  is independent

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

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

**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:

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

**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.:

$$\rho(S^*) = \max_{S \in \Omega_n} [\rho(S)] \rightarrow \forall \omega (\omega \in S^* \rightarrow \omega \text{ is Maximal}) \quad (1.15)$$

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

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

**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

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

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

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

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  $p_s \geq p_{s+1}$ ,  $s + 1 < n$ , 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  $\omega$ , and denote by  $i \triangleright \omega$ , when  $b \leq i \leq s$ .

**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:

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

i.e. the indices of all ascending pairs of  $p$  are included in  $\omega$ .

**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.

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

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  $\omega'$ , then join  $\omega$  and  $\omega'$  to get an Operation Set with cost greater than  $S^*$ . If that is not the case, extend  $\omega$  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  $S' = S^* \uplus \{(i, i+1)\}$ , 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.  $\square$

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

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

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

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.  $\forall \omega (\omega \in S^* \rightarrow \omega \text{ is Maximal})$
2.  $S^*$  is Great

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

**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  $(s_l == i - 1) \wedge (p_{s_l} \leq p_i)$ 
  - (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  $(s_l < i - 1) \wedge (p_{i-1} \leq p_i)$ 
  - (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-Algorithm

---

- 1:  $\Delta p \leftarrow [\langle p_{i+1} - p_i \rangle \text{ for } i \in \{0, \dots, n - 2\}]$
  - 2:  $\Delta p_{>} \leftarrow \text{filter}(\Delta p, (>= 0))$
  - 3:  $r \leftarrow \text{sum}(\Delta p_{>})$
  - 4: **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

$$n = 6 \tag{2.2}$$

$$m = 3 \tag{2.3}$$

$$v = \langle 1, -2, 3, 10, -8, 0 \rangle \tag{2.4}$$

$$q = \langle \langle 0, 2 \rangle, \langle 1, 4 \rangle, \langle 3, 3 \rangle \rangle \tag{2.5}$$

$$a = \langle 2, 3, 10 \rangle = \langle 1 - 2 + 3, -2 + 3 + 10 - 8, 10 \rangle \tag{2.6}$$

## 2.3 Solution Naive

---

### Algorithm 2 Naive

---

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

```

1:  $a = \text{zeros}(m)$ 
2: for  $k \in \{0, \dots, m-1\}$  do
3:    $\langle i, j \rangle \leftarrow Q[k]$ 
4:    $a[k] \leftarrow \text{sum}(\langle v[i], \dots, v[j] \rangle)$ 
5: return  $a$ 

```

---

## 2.4 Solution Optimized

Notice that:

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

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

---

### Algorithm 3 Opt

---

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

```

1:  $\Delta s \leftarrow \text{zeros}(n+1)$ 
2: for  $i \in \{0, \dots, n-1\}$  do
3:    $\Delta s[i+1] \leftarrow \Delta s[i] + v[i]$ 
4:  $a = \text{zeros}(m)$ 
5: for  $k \in \{0, \dots, m-1\}$  do
6:    $\langle i, j \rangle \leftarrow Q[k]$ 
7:    $a[k] = \Delta s[j+1] - \Delta s[i]$ 
8: return  $a$ 

```

---