# COL351

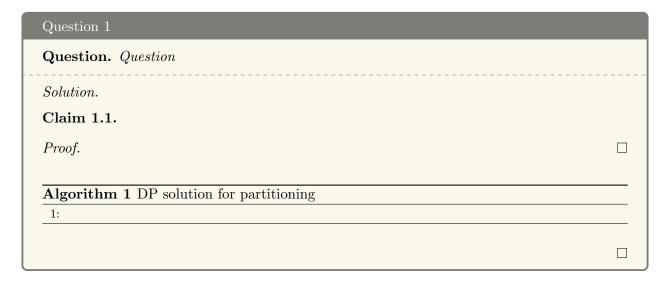
# Assignment 3

### Mallika Prabhakar (2019CS50440) Sayam Sethi (2019CS10399)

## ${\bf September}\ 2021$

## Contents

1	Que	estic	n	1																					2
2	Que	estic	n	2																					3
3	Que	estic	n	3																					5
	3.1	3.1										 													5
	3.2																								5
	3.3	0.0																							5
	3.4	3.4										 													6
	3.5	3.5																						•	6
4	Que																								7
	4.1																								7
	4.2	4.2										 													7
	4.3	4.3										 													7



#### Question 2

Question. The total net force on particle j, by Coulomb's Law, is equal to

$$F_{j} = \sum_{i < j} \frac{Cq_{i}q_{j}}{(j-i)^{2}} - \sum_{i > j} \frac{Cq_{i}q_{j}}{(j-i)^{2}}$$
(1)

Design an algorithm that computes all the forces  $F_j$  in  $O(n \log n)$  time.

Solution. We will use polynomial multiplication to solve this question. Consider the polynomials:

$$A(x) = (0, q_1, q_2, \dots, q_n)$$

$$B(x) = \left(-\frac{1}{(n-1)^2}, -\frac{1}{(n-2)^2}, \dots, -\frac{1}{1^2}, 0, \frac{1}{1^2}, \dots, \frac{1}{(n-2)^2}, \frac{1}{(n-1)^2}\right)$$
(2)

In the above representation, only the coefficients of A(x), B(x) are shown. The degrees of A(x) and B(x) are n and 2n-2 respectively. Now, in the product  $P(x) = A(x) \cdot B(x)$ , consider the coefficient of  $x^{j+n-1}$ . To visualise this, we will write the polynomials as:

$$A(x) = q_n x^n + \dots + q_{j+1} x^{j+1} + q_j x^j + q_{j-1} x^{j-1} + \dots + q_1 x^1 + 0x^0$$

$$B(x) = \dots + -\frac{1}{(n-j)^2} x^{j-1} + \dots + -\frac{1}{1^2} x^{n-2} + 0x^{n-1} + \frac{1}{1^2} x^n + \dots + \frac{1}{(j-1)^2} x^{j+n-2} + \dots$$

$$+ \dots$$

Multiplication of corresponding terms gives terms with power of x as j + n - 1, and thus formally, the coefficient of  $x^{j+n-1}$  can be written as:

$$P(x)[j+n-1] = \sum_{k=1}^{n-j} q_{j+k} \cdot -\frac{1}{k^2} + 0 + \sum_{k=1}^{j-1} q_{j-k} \cdot \frac{1}{k^2}$$
 (4)

Where P(x)[p] denotes the coefficient of  $x^p$  in P(x). Equation 4 can be rewritten as:

$$P(x)[j+n-1] = \sum_{i=j+k,k=1}^{n-j} q_i \cdot -\frac{1}{(j-1)^2} + \sum_{i=j-k,k=1}^{j-1} q_i \cdot \frac{1}{(j-1)^2}$$

$$= -\sum_{i=j+1}^{n} \frac{q_i}{(j-1)^2} + \sum_{i=1}^{j-1} \frac{q_i}{(j-1)^2}$$

$$= \sum_{ij} \frac{q_i}{(j-1)^2}$$

$$= \frac{F_j}{Cq_j}$$

$$\implies F_j = P(x)[j+n-1] \times Cq_j$$
(5)

Therefore, we have derived an alternate method for computing  $F_j$ . Since this involves computing product of polynomials, we can perform the polynomial product in  $O(n \log n)$  since both A(x), B(x) are polynomials of degree O(n). Once C(x) has been computed, we can then compute  $F_j$  in O(1) for each j by dividing the corresponding coefficient with  $Cq_j$ . The exact algorithm is given as:

#### **Algorithm 2** Computing $F_j$ for $j \in \{1, 2, ..., n\}$

```
1: procedure Compute Forces((q, n))
 2:
         B \leftarrow \left[ -\frac{1}{(n-1)^2}, -\frac{1}{(n-2)^2}, \dots, -\frac{1}{1^2}, 0, \frac{1}{1^2}, \dots, \frac{1}{(n-2)^2}, \frac{1}{(n-1)^2} \right]
          P \leftarrow multiply(A, B) \triangleright \text{multiply } A(x) \text{ and } B(x) \text{ using FFT "divide and conquer" algo}
          F \leftarrow C[n:2n-1]
                                                  \triangleright taking subarray corresponding to coefficients of x^{j+n-1}
         for i \in [1, n] do
                                                                                                             ▷ 1-indexed array
 6:
               F[i] \leftarrow F[i] \times Cq[i]
 7:
          end for
 8:
         return F
 9:
10: end procedure
```

**Proof of Correctness:** The proof of correctness of the *FFT Algorithm* has been discussed in the lectures. The correctness of lines 5-8 has been proved from Equation 5.

**Time Complexity:** All operations except the *FFT Algorithm* are O(n) operations. The complexity of *FFT Algorithm* has been shown to be  $O(d \log d)$  where d is the degree of the polynomial. Since the degrees of A(x), B(x) are O(n), the *FFT Algorithm* can be computed in  $O(n \log n)$  time.

Therefore, all forces  $F_j$  can be computed in  $O(n \log n)$  time. This completes the design of the algorithm along with proof of correctness and time complexity.

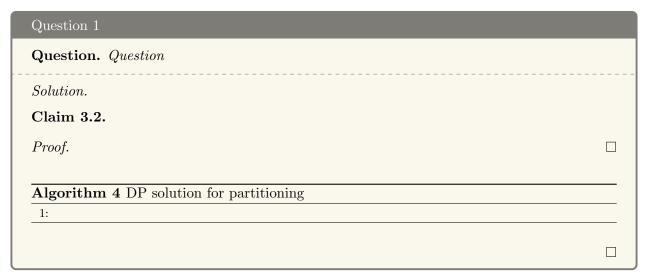
### 3.1 3.1

Question 3	
<b>Question.</b> Prove that the graph $H = (V, E_H)$ can be computed from $G$ in $O(n^{\omega})$ time, $\omega$ is the exponent of matrix-multiplication.	where
Solution.	

#### 3.2 3.2

Question 1	
Question. Question	
Solution.	
Claim 3.1.	
Proof.	
Algorithm 3 DP solution for partitioning	
<u>1:</u>	

#### 3.3 3.3



#### 3.4 3.4

Question 1	
Question. Question	
Solution.	
Claim 3.3.	
Proof.	
Algorithm 5 DP solution for partitioning	
<u>1:</u>	

#### 3.5 3.5

Question 1	
Question. Question	
Solution.	
Claim 3.4.	
Proof.	
Algorithm 6 DP solution for partitioning  1:	

# 4.1 4.1

Question 1	
Question. Question	
Solution.	
Claim 4.1.	
Proof.	
Algorithm 7 DP solution for partitioning	
_1:	

### 4.2 4.2

Question 1	
Question. Question	
Solution.	
Claim 4.2.	
Proof.	
Algorithm 8 DP solution for partitioning	
1:	

### 4.3 4.3

Question 1	
Question. Question	
Solution.	
Claim 4.3.	
Proof.	

Algorithm 9 DP solution for partitioning	
1:	