

COL351

Assignment 3

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

September 2021

Contents

1	Question 1	2
2	Question 2	3
3	Question 3	5
3.1	3.1	5
3.2	3.2	7
3.3	3.3	7
3.4	3.4	7
3.5	3.5	8
4	Question 4	9
4.1	4.1	9
4.2	4.2	9
4.3	4.3	9

1 Question 1

Question 1

Question. *Question*

Solution.

Claim 1.1.

Proof.



Algorithm 1 DP solution for partitioning

1:



2 Question 2

Question 2

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

$$F_j = \sum_{i < j} \frac{Cq_i q_j}{(j-i)^2} - \sum_{i > j} \frac{Cq_i q_j}{(j-i)^2} \quad (1)$$

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:

$$\begin{aligned} A(x) &= (0, q_1, q_2, \dots, q_n) \\ B(x) &= \left(-\frac{1}{(n-1)^2}, -\frac{1}{(n-2)^2}, \dots, -\frac{1}{1^2}, 0, \frac{1}{1^2}, \dots, \frac{1}{(n-2)^2}, \frac{1}{(n-1)^2} \right) \end{aligned} \quad (2)$$

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:

$$\begin{aligned} A(x) &= q_n x^n + \dots + q_{j+1} x^{j+1} + q_j x^j + q_{j-1} x^{j-1} + \dots + q_1 x^1 + 0x^0 \\ B(x) &= \dots + -\frac{1}{(n-j)^2} x^{j-1} + \dots + -\frac{1}{1^2} x^{n-2} + 0x^{n-1} + \frac{1}{1^2} x^n + \dots + \frac{1}{(j-1)^2} x^{j+n-2} \\ &\quad + \dots \end{aligned} \quad (3)$$

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:

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

Where $P(x)[p]$ denotes the coefficient of x^p in $P(x)$.

Equation 4 can be rewritten as:

$$\begin{aligned} P(x)[j+n-1] &= \sum_{i=j+k, k=1}^{n-j} q_i \cdot -\frac{1}{(j-1)^2} + \sum_{i=j-k, k=1}^{j-1} q_i \cdot \frac{1}{(j-1)^2} \\ &= -\sum_{i=j+1}^n \frac{q_i}{(j-1)^2} + \sum_{i=1}^{j-1} \frac{q_i}{(j-1)^2} \\ &= \sum_{i < j} \frac{q_i}{(j-1)^2} - \sum_{i > j} \frac{q_i}{(j-1)^2} \\ &= \frac{F_j}{Cq_j} \\ \implies F_j &= P(x)[j+n-1] \times Cq_j \end{aligned} \quad (5)$$

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 $j \in \{1, 2, \dots, n\}$

```

1: procedure COMPUTE FORCES( $(q, n)$ )
2:    $A \leftarrow q$ 
3:    $B \leftarrow \left[ -\frac{1}{(n-1)^2}, -\frac{1}{(n-2)^2}, \dots, -\frac{1}{1^2}, 0, \frac{1}{1^2}, \dots, \frac{1}{(n-2)^2}, \frac{1}{(n-1)^2} \right]$ 
4:    $P \leftarrow \text{multiply}(A, B) \triangleright$  multiply  $A(x)$  and  $B(x)$  using FFT “divide and conquer” algo
5:    $F \leftarrow C[n : 2n - 1] \triangleright$  taking subarray corresponding to coefficients of  $x^{j+n-1}$ 
6:   for  $i \in [1, n]$  do  $\triangleright$  1-indexed array
7:      $F[i] \leftarrow F[i] \times Cq[i]$ 
8:   end for
9:   return  $F$ 
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. \square

3 Question 3

3.1 3.1

Question 3

Question. Prove that the graph $H = (V, E_H)$ can be computed from G in $O(n^\omega)$ time, where ω is the exponent of matrix-multiplication.

Proof. Enumerate the vertices V in G as $\{1, 2, \dots, |V| = n\}$ and let A_G be the adjacency matrix of G . Consider the term A_G^2 . From **Lemma 1** of Lecture 22, we know that A_G^2 is positive only if there exists a walk of length *exactly* 2. Therefore, we have the following claim:

Claim 3.1. The adjacency list for graph H is given as $A_G + A_G^2 \succ 0$, where A_G is adjacency matrix of G .

Proof. From definition of H , we have that edges in graph H consists of all edges of graph G and end points of walks of length 2. Therefore, E_H has all edges of walks of length 1 and 2. In other words, $(A_H)_{ij}$ is positive only if there exists a walk of length 1 or 2 between nodes i, j . This can be formally written as:

$$\begin{aligned} (A_H)_{ij} &= (A_G)_{ij} > 0 \vee (A_G^2)_{ij} > 0 \\ &= (A_G)_{ij} + (A_G^2)_{ij} > 0 \\ \implies A_H &= A_G + A_G^2 \succ 0 \end{aligned} \tag{6}$$

□

Therefore, the algorithm for computing A_H is:

Algorithm 3 Computing A_H

```

1: procedure COMPUTEH( $G$ )
2:    $A_G \leftarrow \text{adjacency}(G)$ 
3:    $A_H \leftarrow A_G + A_G^2$ 
4:    $n \leftarrow |V_G|$ 
5:   for  $i, j \in [1, n] \times [1, n]$  do
6:     if  $(A_H)_{ij} > 0$  then
7:        $(A_H)_{ij} \leftarrow 1$ 
8:     else
9:        $(A_H)_{ij} \leftarrow 0$ 
10:    end if
11:  end for
12:  return  $\text{graph}(A_H)$ 
13: end procedure

```

Time Complexity Computing A_G^2 will take $O(n^\omega)$ time. All other steps take $O(n^2)$ time. We know that $\omega \geq 2$. Therefore, the overall time complexity of the algorithm will be $O(n^\omega)$. Therefore, we have proposed an algorithm which computes the graph H via its adjacency matrix in $O(n^\omega)$ time. This completes the proof. □

3.2 3.2

Question 1	
Question. <i>Question</i>	
<hr/>	
<i>Solution.</i>	<input type="checkbox"/>

3.3 3.3

Question 1	
Question. <i>Question</i>	
<hr/>	
<i>Solution.</i>	
Claim 3.2.	
<i>Proof.</i>	<input type="checkbox"/>
<hr/>	
Algorithm 4 DP solution for partitioning	
<hr/>	
1:	
<hr/>	
	<input type="checkbox"/>

3.4 3.4

Question 1	
Question. <i>Question</i>	
<hr/>	
<i>Solution.</i>	
Claim 3.3.	
<i>Proof.</i>	<input type="checkbox"/>
<hr/>	
Algorithm 5 DP solution for partitioning	
<hr/>	
1:	
<hr/>	
	<input type="checkbox"/>

3.5 3.5

Question 1

Question. *Question*

Solution.

Claim 3.4.

Proof. ☐

Algorithm 6 DP solution for partitioning

1: ☐

4 Question 4

4.1 4.1

Question 1	
Question. <i>Question</i>	
<hr/>	
<i>Solution.</i>	
Claim 4.1.	
<i>Proof.</i>	<input type="checkbox"/>
<hr/>	
Algorithm 7 DP solution for partitioning	
1:	
<hr/>	
	<input type="checkbox"/>

4.2 4.2

Question 1	
Question. <i>Question</i>	
<hr/>	
<i>Solution.</i>	
Claim 4.2.	
<i>Proof.</i>	<input type="checkbox"/>
<hr/>	
Algorithm 8 DP solution for partitioning	
1:	
<hr/>	
	<input type="checkbox"/>

4.3 4.3

Question 1	
Question. <i>Question</i>	
<hr/>	
<i>Solution.</i>	
Claim 4.3.	
<i>Proof.</i>	<input type="checkbox"/>

Algorithm 9 DP solution for partitioning

1:

