COL351
Assignment 3

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

September 2021

Contents

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*

$$F_j = \sum_{i<j} \frac{Cq_iq_j}{(j-i)^2} - \sum_{i>j} \frac{Cq_iq_j}{(j-i)^2} \tag{1}$$

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:

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

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:

$$\begin{array}{llllll} A(x) = & & q_nx^n + \cdots + & q_{j+1}x^{j+1} + & q_jx^j + & q_{j-1}x^{j-1} + \cdots + q_1x^1 + 0x^0 \\ B(x) = \cdots + & -\dfrac{1}{(n-j)^2}x^{j-1} + \cdots + & -\dfrac{1}{1^2}x^{n-2} + & 0x^{n-1} + & & \dfrac{1}{1^2}x^n + \cdots + \dfrac{1}{(j-1)^2}x^{j+n-2} \\ & & & & & + \cdots \end{array} \tag{3}$$

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:

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

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

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

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

```
1: procedure COMPUTE FORCES((q, n))
2:     A ← q
3:     B ← [−1/(n−1)^2, −1/(n−2)^2, ..., −1/1^2, 0, 1/1^2, ..., 1/(n−2)^2, 1/(n−1)^2]
4:     P ← multiply(A, B) ▷ multiply A(x) and B(x) using FFT "divide and conquer" algo
5:     F ← C[n : 2n − 1]        ▷ taking subarray corresponding to coefficients of x^{j+n−1}
6:     for i ∈ [1, n] do                                              ▷ 1-indexed array
7:         F[i] ← F[i] × 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. □

3 Question 3

3.1 3.1

Question 3(a)

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 $\{1, 2, \ldots, |V| = n\}$ 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:

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

□

Therefore, the algorithm for computing A_H is:

Algorithm 3 Computing A_H

procedure ComputeH(G)
    $A_G \leftarrow adjacency(G)$
    $A_H \leftarrow A_G + A_G^2$
    $n \leftarrow |V_G|$
    for $i, j \in [1, n] \times [1, n]$ **do**
        if $(A_H)_{ij} > 0$ **then**
            $(A_H)_{ij} \leftarrow 1$
        else
            $(A_H)_{ij} \leftarrow 0$
        end if
    end for
    return $graph(A_H)$
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 3(b)

Question. *Argue that for any* $x, y \in V$, $D_H(x, y) = \left\lceil \frac{D_G(x, y)}{2} \right\rceil$

Solution. We will prove the given statement by first showing that there exists a path of length $\left\lceil \frac{D_G(x,y)}{2} \right\rceil$ for each x, y in H. We will then prove that we cannot have a shorter path length in H.

Note: For this and subsequent parts, we call edges which are directly in G as edges of *type* 1 and the other edges as edges of *type* 2.

Claim 3.2. *For each* $x, y \in V$*, there exists a path of length* $\left\lceil \frac{D_G(x,y)}{2} \right\rceil$ *in graph* H*, corresponding to the shortest path in* G*.*

Proof. Let the shortest path between x, y in G be given as:

$$\begin{aligned} P_G(x, y) &= \{x, a_1, a_2, \ldots, a_k, y\} \\ \implies D_G(x, y) &= k + 1 \end{aligned} \tag{7}$$

We now have two cases, when k is odd and when k is even. For the case when k is odd, we have:

$$\begin{aligned} P_H(x, y) &= \{x, a_2, a_4, \ldots, a_{k-1}, y\} \\ \implies \mathit{length}(P_H(x, y)) &= \frac{k-1}{2} + 1 \\ &= \frac{k+1}{2} \\ &= \left\lceil \frac{D_G(x, y)}{2} \right\rceil \end{aligned} \tag{8}$$

When k is even, we have:

$$\begin{aligned} P_H(x, y) = \{x, a_2, a_4, \ldots, a_k, y\} &\ ((a_k, y) \text{ is the only edge of type 1}) \\ \implies \mathit{length}(P_H(x, y)) &= \frac{k}{2} + 1 \\ &= \frac{(k+1)+1}{2} \\ &= \left\lceil \frac{D_G(x, y)}{2} \right\rceil \end{aligned} \tag{9}$$

Therefore we have shown the correctness of the claim for both cases of k. □

We will now show that there cannot exist a path between x, y of shorter length in H.

Claim 3.3. *The shortest distance between* x, y *is given exactly as* $\left\lceil \frac{D_G(x,y)}{2} \right\rceil$

Proof. We will prove the claim using contradiction. Assume that there exists a shorter path $Q_H(x, y)$:

$$\begin{aligned} Q_H(x,y) &= \{x, b_1, b_2, \ldots, b_m, y\} \\ \implies length(Q_H(x,y)) &= m + 1 < \left\lceil \frac{D_G(x,y)}{2} \right\rceil, \text{ from assumption} \end{aligned} \tag{10}$$

Consider the edges in G corresponding to this path $Q_H(x, y)$:

$$\begin{aligned} &Q_G(x,y) = \{x, c_1, b_1, c_2, b_2, \ldots, c_m, b_m, c_{m+1}, y\}, \ c_i \text{ may be the same as } b_i \\ \implies &length(Q_G(x,y)) \leq 2m + 2 < 2\left\lceil \frac{D_G(x,y)}{2} \right\rceil \\ \implies &length(Q_G(x,y)) < \begin{cases} D_G(x,y) + 1, & D_G(x,y) \text{ is odd} \\ D_G(x,y), & D_G(x,y) \text{ is even} \end{cases} \end{aligned} \tag{11}$$

We know that $D_G(x, y)$ is the shortest path in G between vertices x, y. Therefore, we have that such a path cannot exist if $D_G(x, y)$ is even and in the case when $D_G(x, y)$ is odd, we notice that the inequality in $length(Q_G(x, y))$ has an even number $(2m + 2)$ in the RHS. Therefore, the equality cannot hold in this case as well. Thus, we have arrived at a condradiction on the length of shortest x, y path in G. Therefore, $\left\lceil \frac{D_G(x,y)}{2} \right\rceil$ is the shortest path in H. □

Thus, from Claim 3.2 and Claim 3.3 we have shown that $D_H(x, y) = \left\lceil \frac{D_G(x,y)}{2} \right\rceil$.
Hence, proved. □

3.3 3.3

Question 1

Question. *Question*

Solution.

Claim 3.4.

Proof. □

Algorithm 4 DP solution for partitioning

1:

□

3.4 3.4

Question 1

Question. *Question*

Solution.

Claim 3.5.

Proof. □

Algorithm 5 DP solution for partitioning

1:

□

3.5 3.5

Question 1

Question. *Question*

Solution.

Claim 3.6.

Proof. □

Algorithm 6 DP solution for partitioning

1:

□

4 Question 4

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:

□