ESO207A: Data Structures and Algorithms

Theoretical Assignment 2

Due Date: 7th October, 2023

Total Number of Pages: 3 | Total Points 100

Instructions-

1. For submission typeset the solution to each problem and compile them in a single pdf file. Hand-written solutions will not be accepted. Use LaTeX only for typesetting.
2. Start each problem from a new page. Write down your Name, Roll number and problem number clearly for each problem.
3. For each question, give the pseudo-code of the algorithm with a clear description of the algorithm. Unclear description will receive less marks. Less optimal solutions will receive only partial marks.
4. Don't add any screenshots of code, etc. in your solution.

Question 1. Search Complicated

You are given an array $A[0,\ldots,n-1]$ of n distinct integers. The array has following three properties:

- First $(n-k)$ elements are such that their value increase to some maximum value and then decreases.
- Last k elements are arranged randomly
- Values of last k elements is smaller compared to the values of first $n-k$ elements.

(a) (10 points) You are given q queries of the variable **Val**. For each query, you have to find out if **Val** is present in the array A or not. Write a pseudo-code for an $\mathcal{O}(k\log(k)+q\log(n))$ time complexity algorithm to do the task.
(Higher time complexity correct algorithms will also receive partial credit)

(b) (5 points) Explain the correctness of your algorithm and give the complete time complexity analysis for your approach in part (a).

Question 2. Perfect Complete Graph

A directed graph with n vertices is called Perfect Complete Graph if:

- There is exactly one directed edge between every pair of distinct vertices.
- For any three vertices a, b, c, if (a,b) and (b,c) are directed edges, then (a,c) is present in the graph.

Note: **Outdegree** of a vertex v in a directed graph is the number of edges going out of v.

(a) (20 points) Prove that a directed graph is a Perfect Complete Graph if and only if between any pair of vertices, there is at most one edge, and for all $k \in \{0,1,\ldots,n-1\}$, there exist a vertex v in the graph, such that **Outdegree**$(v)=k$.

(b) (10 points) Given the adjacency matrix of a directed graph, design an $\mathcal{O}(n^2)$ algorithm to check if it is a perfect complete graph or not. Show the time complexity analysis. You may use the characterization given in part (a).

Question 3. PnC (20 points)

You are given an array $A = [a_1, a_2, a_3, \ldots, a_n]$ consisting of n **distinct**, **positive** integers. In one operation, you are allowed to swap the elements at any two indices i and j in the **present array** for a cost of $\max(a_i, a_j)$. You are allowed to use this operation any number of times.

Let Π be a permutation of $\{1, 2, \ldots, n\}$. For an array A of length n, let $A(\Pi)$ be the permuted array $A(\Pi) = [a_{\Pi(1)}, a_{\Pi(2)}, \ldots, a_{\Pi(n)}]$.

We define the score of an array A of length n as

$$S(A) = \sum_{i=1}^{i=n-1} |a_{i+1} - a_i|$$

(a) (5 points) Explicitly characterise **all** the permutations $A(\Pi_0) = [a_{\Pi_0(1)}, a_{\Pi_0(2)}, a_{\Pi_0(3)} \ldots, a_{\Pi_0(n)}]$ of A such that

$$S(A(\Pi_0)) = \min_{\Pi} S(A(\Pi))$$

We call such permutations, a *"good permutation"*. In short, a *good permutation* of an array has minimum score over all possible permutations.

(b) (15 points) Provide an algorithm which computes the minimum cost required to transform the given array A into a *good permutation*, $A(\Pi_0)$.
The cost of a transformation is defined as the sum of costs of each individual operation used in the transformation.
You will only be awarded full marks if your algorithm works correctly in $\mathcal{O}(n \log n)$ in the worst case, otherwise you will only be awarded partial marks, if at all.

(c) (0 points) Bonus: Prove that your algorithm computes the minimum cost of converting any array A into a *good permutation*.

Some examples are given below for the sake of clarity:

- Regarding the operation:

Array	(i, j)	Cost
$A = [7, 2, 5, 4, 1]$	$(1, 3)$	$\max(a_1, a_3) = \max(7, 5) = 7$
$P_1 = [5, 2, 7, 4, 1]$	$(2, 5)$	$\max(2, 1) = 2$
$P_2 = [5, 1, 7, 4, 2]$	$(3, 5)$	$\max(7, 2) = 7$
Final Array $= [5, 1, 2, 4, 7]$	–	–

  In essence, the order of operations contributes significantly to the cost of a transformation.

- Regarding the cost of a transformation:
  The cost of transforming the array $A = [7, 2, 5, 4, 1]$ to $P_3 = [5, 1, 2, 4, 7]$ using the **exact** sequencce of operations mentioned above is $7 + 2 + 7 = 16$.

- Regarding permutations:
  Let Π be such that $[1, 2, 3, 4, 5] \rightarrowtail [5, 3, 1, 4, 2]$ and $A = [7, 2, 5, 4, 1]$, then $A(\Pi) = [5, 1, 2, 4, 7]$.

Question 4. Mandatory Batman Question (20 points)

Batman gives you an undirected, unweighted, connected graph $G = (V, E)$ with $|V| = n, |E| = m$, and two vertices $s, t \in V$.
He wants to know $dist(s, t)$ given that the edge (u, v) is destroyed, for each edge $(u, v) \in E$. In other words, for each $(u, v) \in E$, he wants to know the distance between s and t in the graph $G' = (V', E')$, where $E' = E \backslash \{(u, v)\}$.

Some constraints:

- The *dist* definition and notation used is the same as that in lectures.
- It is guaranteed that t is always reachable from s using some sequence of edges in E, even after any edge is destroyed.
- To help you, Batman gives you an $n \times n$ matrix $M_{n \times n}$. You have to update $M[u, v]$ to contain the value of $dist(s, t)$ if the edge (u, v) is destroyed, for each $(u, v) \in E$.
- You can assume that you are provided the edges in adjacency list representation.
- The edge (u, v) is considered the same as the edge (v, u).

(a) (12 points) Batman expects an algorithm that works in $\mathcal{O}(|V| \cdot (|V| + |E|) = \mathcal{O}(n \cdot (n + m))$.

(b) (4 points) He also wants you to provide him with proof of runtime of your algorithm, i.e., a Time-Complexity Analysis of the algorithm you provide.

(c) (4 points) Lastly, you also need to provide proof of correctness for your algorithm.

Question 5. No Sugar in this Coat (15 points)

You are given an **undirected**, **unweighted** and **connected** graph $G = (V, E)$, and a vertex $s \in V$, with $|V| = n$, $|E| = m$ and $n = 3k$ for some integer k. Let distance between u and v be denoted by $dist(u, v)$ (same definition as that in lectures).
G has the following property:

- Let $V_d \subseteq V$ be the set of vertices that are at a distance equal to d from s in G, then

$$\forall\ i \geq 0 : \quad u \in V_i, v \in V_{i+1} \Rightarrow (u, v) \in E$$

Provide the following:

(a) (10 points) An $\mathcal{O}(|V| + |E|)$ time algorithm to find a vertex $t \in V$, such that the following property holds for every vertex $u \in V$:

$$\min(dist(u, s), dist(u, t)) \leq k$$

Note that your algorithm can report s as an answer if it satisfies the statement above.

(b) (5 points) Proof of correctness for your algorithm.