ASSIGNMENT - CSL251

Name - Paritosh Lahre
ID - 12341550

Question 1

Problem Statement

For this question, you'll need the following definition:

Definition: A sunlet is a graph with $2n$ vertices that consists of a cycle of length n, and each vertex in the cycle is directly connected to exactly one node of degree one.

Model: Input is an undirected graph G with $2n$ vertices. The algorithm can query an edge (i, j) and it will be told whether or not edge (i, j) is in graph G. Each query has cost 1.

Problem: Output "Yes" if the input graph G is a sunlet, otherwise output "No".

task: Prove that every correct algorithm for the problem has worst-case cost at least $\binom{2n}{2}$.

Claim

In the worst case, any deterministic algorithm must make $\binom{2n}{2}$ edge queries to correctly identify whether the hidden graph is a sunlet.

Proof (Detailed Explanation)

Suppose there exists a deterministic algorithm $\mathcal{A}$ that can determine whether an unknown graph G is a sunlet using fewer than $\binom{2n}{2}$ edge queries. This means that $\mathcal{A}$ does not check the existence of all possible edges between pairs of the $2n$ vertices.

Let U be the set of unordered vertex pairs whose edge status was never queried by $\mathcal{A}$. Then $U \neq \emptyset$ by assumption.

We now consider an adversarial strategy. The adversary will respond to $\mathcal{A}$'s queries in a way that makes the graph appear to be a valid sunlet. Let G_1 be the actual sunlet graph (a cycle of n nodes, each with a unique leaf attached). Now, consider a pair $(u, v) \in U$ — the algorithm never asked whether there was an edge between u and v.

Now the adversary constructs another graph G_2:

- G_2 is identical to G_1 in all queried edges.
- But G_2 contains one additional edge (u, v) where both u and v are degree-one leaf nodes.

This new graph G_2 is no longer a valid sunlet. Why?

- In a sunlet, every non-cycle node must have degree exactly 1.
- Adding an edge between two leaf nodes results in one or both of them having degree 2.
- Therefore, G_2 violates the sunlet condition.

Since $\mathcal{A}$ never queried (u, v), it cannot distinguish between G_1 and G_2. Thus, it will produce the same output for both graphs — either incorrectly rejecting the true sunlet or incorrectly accepting the non-sunlet.

Hence, in order to avoid such adversarial constructions, any algorithm must query every pair (i, j) with $1 \le i < j \le 2n$. That is:

$$\boxed{\binom{2n}{2}} \text{ edge queries are necessary in the worst case.}$$

Example (n=3)

- Vertices: 1 to 6
- Cycle: $(1{-}2), (2{-}3), (3{-}1)$
- Leaves: $(1{-}4), (2{-}5), (3{-}6)$
- Suppose the algorithm does not query $(4, 5)$
- The adversary adds edge $(4, 5)$ to break the sunlet condition (since now 4 and 5 have degree 2)
- The algorithm cannot distinguish this from a valid sunlet and may output the wrong result.

Problem 2

Consider the following model and problem:

Model

For the range of numbers $1, 2, \ldots, n$, there is a special threshold value $t \in \{0, 1, \ldots, n\}$. For all numbers $i > t$, the number i is considered "unsafe". All other numbers in the range $1, 2, \ldots, t$ are considered "safe". The algorithm can query any number $i \in \{1, 2, \ldots, n\}$, and it will be told whether i is "safe" or "unsafe". However, if the algorithm ever queries an "unsafe" i, the system shuts down and no further queries are possible.

Problem

Determine the exact value of t.

Task

(a) Prove that any algorithm that solves the problem must perform at least n queries in the worst case.

(b) Let's change the model a bit: suppose that one "unsafe" query is allowed. That is, the system shuts down after exactly two "unsafe" queries. Prove that any algorithm that determines the exact value of t must use $\Omega(\sqrt{n})$ queries in the worst case.

(c) Design an algorithm that uses $O(\sqrt{n})$ many queries for the problem in part (b).

(a) Claim: Worst-case n Queries When No Unsafe Queries are Allowed

Problem Recap

We are given a hidden threshold $t \in \{0, 1, \ldots, n\}$. For any queried number i:

- If $i \le t$, the query returns "safe".
- If $i > t$, the query returns "unsafe" and the system shuts down immediately — no more queries can be made.

The goal is to find the **exact** value of t using the minimum number of queries in the **worst case**.

Worst-Case Lower Bound: n Queries

Since making a single unsafe query ends the process, the only valid strategy is to:

- Query numbers sequentially from 1 to n in increasing order.
- Stop just before the first unsafe number appears.

Why can't we skip values? Suppose the algorithm skips values and directly queries some $i > 1$:

- If $i > t$, and i is unsafe, the system halts and we get no info about the skipped values.
- We risk overshooting t without knowing where the threshold lies.

Thus, querying must be cautious and strictly sequential.

Explanation:

- We test each i from 1 to n.
- When we reach the first i that is unsafe, we stop and return $t = i - 1$.
- If all n queries return safe, then $t = n$.

Example

Let $n = 5$, and $t = 3$:

Query	Response
1	safe
2	safe
3	safe
4	**unsafe (system shuts down)**

The algorithm halts after 4 queries and concludes $t = 3$.

Conclusion

$$\boxed{\text{In the worst case (e.g., when } t = n\text{), the algorithm must make } n \text{ queries.}}$$

Thus, any correct algorithm under this constraint must perform **at least n queries in the worst case**.

(b) Lower Bound: $\Omega(\sqrt{n})$ Queries with One Unsafe Allowed

Claim. Allowing one unsafe query (the second unsafe halts the algorithm) still forces a worst-case query cost of $\Omega(\sqrt{n})$.

Decision-Tree Argument. Represent a deterministic algorithm by a decision tree of depth k. Each root-to-leaf path contains at most two "U" (unsafe) labels before terminating. The number of distinct sequences of length k with at most two U's is

$$\binom{k}{0} + \binom{k}{1} + \binom{k}{2} = 1 + k + \tfrac{k(k-1)}{2}.$$

Since there are $n + 1$ possible thresholds $0, 1, \ldots, n$, we must have

$$1 + k + \tfrac{k(k-1)}{2} \;\geq\; n + 1 \quad \Longrightarrow \quad k = \Omega(\sqrt{n}).$$

Intuitive Insight. Even though one unsafe probe is tolerated, any algorithm skipping too many positions ends up with too few possible S/U patterns to identify among all $n + 1$ threshold values.

(c) Upper Bound: $O(\sqrt{n})$ Queries

Two-Phase Search Algorithm. Let $m = \lceil \sqrt{n} \rceil$. We perform:

1. **Block Search:** Query indices $m, 2m, 3m, \ldots$ until the first unsafe at jm. (At most one unsafe.)
2. **Local Scan:** Let $L = (j - 1)m$. Sequentially query $L + 1, L + 2, \ldots$ until the next unsafe at index i. Then $t = i - 1$. (Second unsafe shuts down, but only after this final probe.)

Complexity. At most m block queries plus m local probes, giving $2m = O(\sqrt{n})$ total. We use at most two unsafe queries (one in each phase) without premature termination.

Worked Example ($n = 20,\ t = 13$). Here $m = 5$. The algorithm runs:

- *Block Search:* Query 5(S), 10(S), 15(U) $\quad (j = 3)$.
- Now $L = 10$, so $t \in \{11, 12, 13, 14\}$.
- *Local Scan:* Query 11(S), 12(S), 13(S), 14(U) $\quad (i = 14)$. Conclude $t = 14 - 1 = 13$.

Total queries $= 3 + 4 = 7 = O(\sqrt{20})$.