Quiz-2

● Graded

Student

Abhinav Shripad

Total Points

18.5 / 24 pts

Question 1

1 **18.5** / 24 pts

1.1 **(a)** **2** / 3 pts

+ 0.6 pts Written ”I do not know how to approach this problem”

+ 0.5 pts Mentioning graph is undirected (or directed with every edge present in both directions)

✓ **+ 0.5 pts** Vertices denote junctions

✓ **+ 0.5 pts** Edges denote the pipes

+ 0.5 pts The weights denote length of the pipes

✓ **+ 0.5 pts** |E| = n

✓ **+ 0.5 pts** Claiming |V| = O(n) since max-degree = 4

+ 0 pts Wrong answer

1.2 **(b)** Resolved **2** / 3 pts

+ 0.6 pts Written ”I do not know how to approach this problem”

+ 1 pt Computing safety distance for every vertex

✓ **+ 0.5 pts** For any given sensitivity, remove all unsafe junctions w.r.t that sensitivity and check if a path exists in remaining graph

✓ **+ 1 pt** Computing maximum sensitivity via binary search

✓ **+ 0.5 pts** Return the path corresponding to maximum sensitivity

+ 0 pts Incorrect answer

Safety distance measurement is necessary before proceeding to binary search.

Regrade Request **Submitted on: Sep 11**

Sir I wrote about safety distance in Idea 3. I wrote what condition should the distance of a node from a sensor follow. Didn't mention how to calculate this distance as I wrote that in the pseudocode.

No you are getting confused... for what you wrote you got full marks....see in idea 1 you are assuming the sensitivity to be d but you don't know what d is and that is what which you should have found by running Dijkstra or any any other tool.
i hope it clarifies your doubt.

Reviewed on: Sep 11

1.3 **(c)** Resolved **6.5 / 8 pts**

+ 1.6 pts Written "I do not know how to approach this problem"

✓ **+ 1 pt** Create G′ by adding an auxiliary vertex x and connecting it to motion sensor junctions via zero weight edges .

✓ **+ 1 pt** Run Dijkstra on G′ from x, to create distance array D

+ 0.5 pts Sorting D using any O(n log n) sort to create D′

Choosing optimal sensitivity via binary search over values in D′

✓ **+ 1.5 pts** For a given sensitivity s, deleting junctions with D[v] s

✓ **+ 1 pt** Checking if a y − z path exists using BFS / DFS

✓ **+ 2 pts** Continuing the binary search in the appropriate half of D′

+ 1 pt For optimal sensitivity, outputting the y − z path in Gs using BFS /DFS

+ 0 pts Incorrect answer

Unclear on where binary search ran also sorting edge weights makes no sense!! Other ideas are correct but needs to be put more formally.

Regrade Request **Submitted on: Sep 11**

Sir in the page 2 of 1(c) I have written how to find the nearest distance to servers. I have NOT made an auxiliary vertex, instead I have pushed all the servers "s" into the priority queue initially as (0,s). It achieves the same outcome as the making an auxiliary vertex and running a Dijkstra on it. It has 2(1+1) marks for it on the rubric. So sir please see once.

I see... it is correct thanks for bringing this up marks are been revised.

Reviewed on: Sep 11

1.4 **(d)** **4 / 6 pts**

Correct approach

+ **1.5 pts** Proving that the algorithm correctly computes the safety distance

✓ **+ 2 pts** Proving that maximum sensitivity s which changes the structure of G_s will be one of the values of D'

+ **1 pt** Claiming correctness of BFS

✓ **+ 1 pt** Claiming correctness of Binary search

+ **0.5 pts** Claiming all the above results imply that the algorithm is correct

---

+ **1.2 pts** I do not know how to approach this problem

+ **0 pts** Incorrect answer

+ 1 pt Your idea is correct

1 Your idea is correct, but you have not proved the calculation of shorted distance from sensors to each vertex, how you are checking reachability and why it is correct.

1.5 **(e)** **4 / 4 pts**

Correct approach

✓ **+ 0.5 pts** Claiming BFS/DFS takes O(n) time as only O(n) edges and vertices

✓ **+ 0.5 pts** Mentioning Dijkstra takes O(n log n) time

✓ **+ 0.5 pts** Sorting D takes O(n log n) time

✓ **+ 1 pt** Each iteration of binary search takes O(n) time

✓ **+ 1.5 pts** Binary search runs over O(log n) iterations

+ **0.8 pts** I do not know how to approach this problem

+ **0 pts** Incorrect answer

[redacted]

COL351: Analysis and Design of Algorithms
Quiz 2

Name: Abhinav Rajesh Shripad	Aug 30, 2024
Entry number: 2022CS11596	Total points: 24

Please write your answers within the box provided. Answers written outside the boxed region will not be graded.

Problem 1 [24 points]

Byomkesh has been assigned to go on a mission to the mansion of his arch enemy, Anukul. To limit exposure, he has decided to travel via an underground sewer network. He has a map of the sewer, composed of n bidirectional *pipes* that connect to each other at *junctions*. Each junction connects to at most four pipes, and every junction is reachable from every other junction via the sewer network. Every pipe is marked with its positive integer length. Some junctions are marked as containing identical motion sensors, any of which will be able to sense Byomkesh if his distance to that sensor (as measured along pipes in the sewer network) is too close. Unfortunately, Byomkesh does not know the sensitivity of the sensors. Describe an $O(n \log n)$ time algorithm to find a path along pipes from a given entrance junction to the junction below Anukul's mansion that stays as far from motion sensors as possible.

Please go through all questions before starting to write your solution.

(a) [**3 points**] Model the problem as a graph. What are the vertices and the edges? What is the size of the graph (in asymptotic notation)?

$$\boxed{\begin{array}{l}\text{Junctions} \rightarrow \text{Node} \rightarrow V \\ \text{Pipes} \rightarrow \text{Edges} \rightarrow E\end{array}}$$

Given:- $|E| = n$, and $|\deg(v)| \leq 4 \ \forall v \in V$

$\rightarrow \sum_{v \in V} \deg(v) \leq 4|V| \rightarrow 2E = 2n \leq 4|V|$

and ||ly since graph is connected

$\rightarrow \sum_{v \in V} \deg(v) \geq \sum_{v \in V}(1) = |V| \rightarrow 2n \geq |V|$

$\rightarrow \boxed{|V| = O(n) \text{ and } |E| = O(n)}$

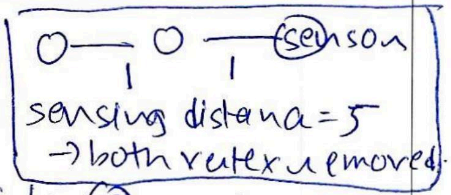
(b) [**3 points**] Write a brief high-level idea of the algorithm (in plain English, with minimal notation). You will be asked for the pseudocode in part (c).

b Idea 1
Observe that if I can find a path that stays "d" distance away from sensors $\rightarrow$ I can also find a path with distance $\leq d$ away from sensors.

$\rightarrow$ Monotonic $\rightarrow$ Binary search

Idea 2 :- Sort the edges in increasing order of weights, say $e_1, e_2, \ldots \ldots e_n$.
Say e_i is feasible, then so is every distance b/w e_{i-1} and e_i feasible.
$\rightarrow$ Binary search on weights of edges.

Idea 3 :- If w is the feasible distance,
$\rightarrow$ No travel to vertex at distance $< w$ ~~which are neighbours of~~ from any sensor vertex. Note:- This can remove non-neighbour of sensors too. Eg



Idea 4:- Remove vertex as per idea 3 and do BFS / DFS.

Name: Abhinav R.Shripad	Aug 30, 2024
Entry number: 2022CS11596	Total points: 24

(c) [**8 points**] Write the pseudocode of your algorithm. Clearly mention the input and the output.

Hint#1: Byomkesh does not know the sensitivity of the motion sensors. If the sensitivity is too high, there may not be a feasible path to Anukul's mansion. On the other hand, if the sensitivity is zero, then the connectivity of the network would imply that such a path certainly exists. Think about the *maximum* sensitivity for which such a path still exists. Can you help Byomkesh discover this quantity?

Hint#2: Some junctions have sensors, while other junctions can be detected from the ones with sensors, and still others may be out of range (and therefore "safe"). Specifically, for a fixed (unknown) sensitivity level of the sensors, a junction is safe if its shortest distance from every junction containing a motion sensor is strictly greater than the sensitivity level. The desired path (if it exists) should only use the safe junctions.

Hint#3: Consider adding an *auxiliary* vertex to your graph if it helps.

```
ndto_sensor = list of nearest distance to
              a sensor each vertex
              has. #how to calculate
              next page. — (1)
edges = sorted (Edges) #sorted as per weight
                                        — (2)
low:=1 , high:=n, mid:=0  ans:= -1
                                        — (3)
while ( low <= high):
    mid = (low+high)/2
    weight = edges[mid].weight
    new_graph = clean_graph(G, weight) — (4)
              #removes edges with distance
               to a server ≤ weight
                  implementation
    ~~if ( reachable(new~~   next page
if ( reachable(new_graph)): — ⑤
        ans = mid
        low = mid+1

else:
      high = mid - 1
return ~~ans~~ weight[ans]  # ans = -1 → no solution
```

2 Functions to implement

```
→ ~~[illegible]~~ clean_graph → ⑥
~~[illegible]~~
→ for u ∈ V:
      if nd to_sensor(u) < weight:
            remove v from G
```

nearest distance to server ★ (Crux of the problem)

→ Similar to Dijkstra,
instead initially we push into priority queue (dist, vertex) pair of (0, s) for all sensors s.
and we proceed similarly.

Name: Abhinav R. Shripad Aug 30, 2024

Entry number: 2022CS11596 Total points: 24

```
~~for~~ pq = priority-queue → 7
for v ∈ sensors:
    pq.push (0, v)

while (!pq.empty());
    ~~top~~ dist, at = pq.top(); pq.pop();
  for (to : adj[at]):
      if ( dist[to] > dist[at] + c(edge)):
          dist[to] = dist[at] + cost(edge)
          pq.push (dist[to], to)
```

THIS is time complexity Analysis. (e), Please see here.

(d) [**6 points**] Prove the correctness of your algorithm. Proof of correctness next page.

AS marked in the pseudocode

(1) → nearest_distance_to_server
→ Dijkstra with extra pushes
→ $\underbrace{O(N)}_{\text{extra push}} + \underbrace{O(E + V\log V)}_{\text{loop}}$
→ $\underline{O(n\log n)}$ # $O(V, E) = O(n)$

(2) → Sorting edges → $O(\underline{n\log n})$

(6) → Graph cleaning
→ process each vertex once
→ $O(n)$

(5) BFS/DFS
→ $O(n)$

(3) → Binary Search
→ $O(\log n * \underline{\text{internal operations}})$
→ $O(n\log n)$

Total T.C. ⇒ (1) + (2) + (3)
$= \underline{O(n\log n)}$

Name: Abhinav R. Shripad	Aug 30, 2024
Entry number: 2022CS11596	Total points: 24

Proof of correctness :-

As written in Idea ② of part ⓑ question, if edges are ~~in~~ sorted in ↑ order of weight, and if ~~e_{10} is good so is ge~~ weight of edge e_i is good so is any ~~edge~~ weight less than it.

and optimum answer will always be equal to an edge weight. Say if not, ~~a~~ increase the weight to ~~an~~ nearest edge weight, and it also satisfies as distance to a server is quantized (~~[illegible]~~ ie not all values are reachable.

① checking on edges ~~[illegible]~~ weight suffice

② Property is monotonic on edge weight

① and ② → binary search of edges weights.

Predicate function is also correct as it does exactly the brute force way to check ~~some~~ reachability.

(e) [**4 points**] Show that your algorithm has the desired running time guarantee.

Written on ~~p~~ solution to question (d) page 1

please check there.