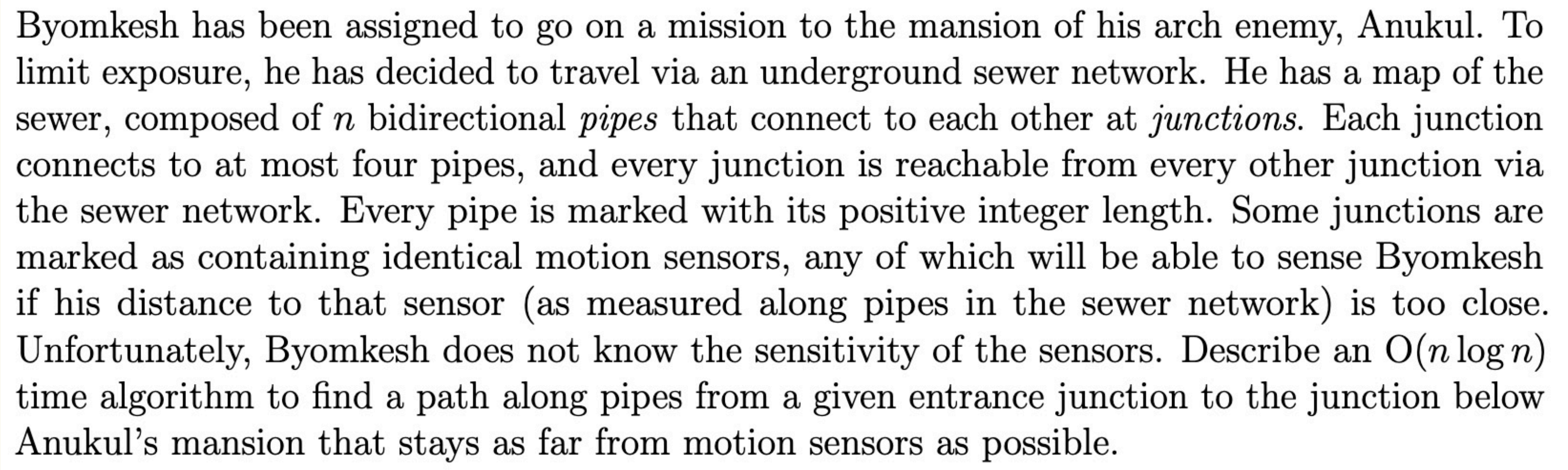
COL 351 : ANALYSIS & DESIGN OF ALGORITHMS

LECTURE 16

QUIZ 2

AUG 30, 2024 | ROHIT VAISH

Problem 1



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.

(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)?

Construct an undirected and weighted graph $G = (V, E, W)$ where the vertices in V denote junctions, the edges in E denote the pipes, and the weights W denote the lengths of the pipes.

Size: $|E| = n$. $|V|$ is $O(n)$ since max degree is four.

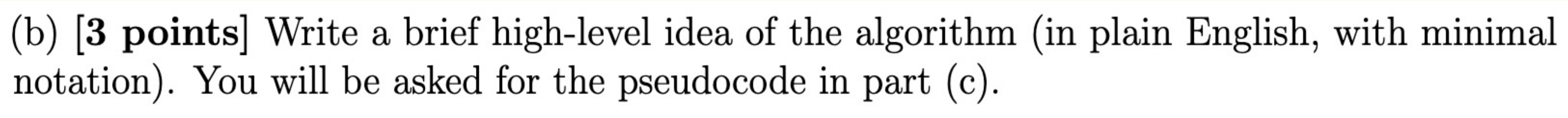
Thus, $|V| + |E|$ is $O(n)$.

(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).

The algorithm proceeds in the following three steps:

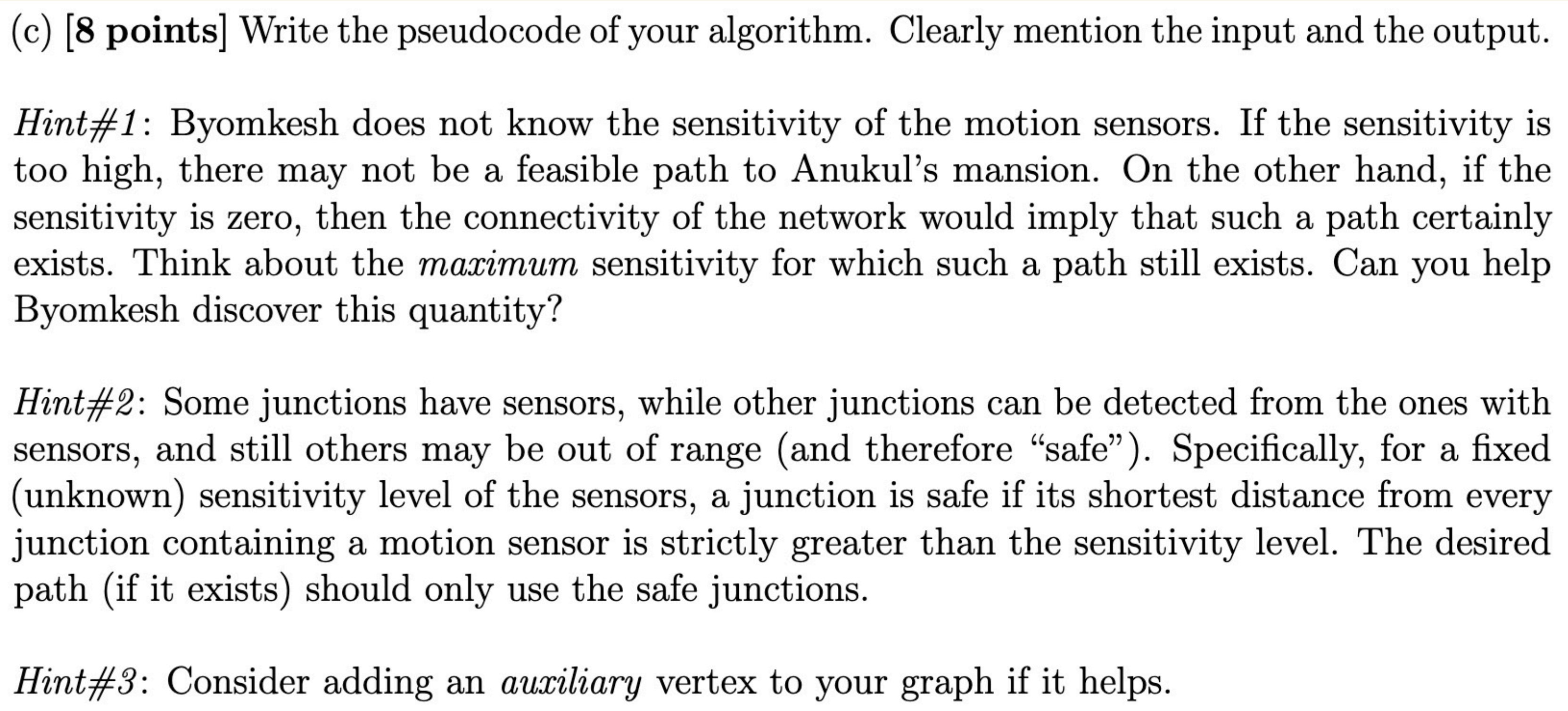
① First, it computes the safety distance for every vertex v, i.e., the shortest distance from v to a vertex with motion sensor. This computation involves running Dijkstra's algorithm on graph G with an auxiliary vertex.

② Then, it computes the maximum sensitivity s^* via binary search. For any sensitivity level s (where s takes values in the safety distance array), the algorithm constructs a graph G_s by removing all unsafe junctions from G and looks for the desired path (via BFS or DFS).



(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).

③ Having determined the maximum sensitivity s^*, the algorithm returns the corresponding path in the graph G_{s^*}.



(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.

input : an undirected and weighted graph $G = (V, E, W)$

a starting vertex y (Byomkesh's starting location)

a destination vertex z (Anukul's mansion)

Output : a path from y to z that goes through safe junctions at maximum sensitivity

// compute "safety distance" array via Dijkstra on modified graph

* initialize an n-length array $D = \emptyset$

* Create an new graph G' by adding an auxiliary vertex "x" to G. Connect x to junctions with motion sensors via zero-weight edges.

```
* Run Dijkstra on $G'$ starting from $x$.

* Update array $D$ with the distance values of the vertices of $G$.

* Let $D'$ be the sorted version of $D$ in ascending order.
  (say, using merge sort)

  // Compute maximum sensitivity $s^*$ via binary search

* Choose sensitivity $s$ via binary search on the array $D'$:
    * Create graph $G_s$ by deleting junctions $v$ with $D[v] \leq s$.
    * Check if a $y$—$z$ path exists (using BFS or DFS)
    * if $y$—$z$ path exists, resume binary search in the larger
      half; otherwise continue in the smaller half.
```

- Let s^* be the largest value in D' for which a $y-z$ path exists.

  // thus, in G_{s^*}, y and z are in the same connected component and in G_{s^*+1}, y and z are in different connected components.

- return the $y-z$ path in G_{s^*} (using BFS or DFS).

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

Let s^{max} denote the actual maximum sensitivity of graph G.

To prove correctness, we should show that the path returned by the algorithm is a valid $y-z$ path at sensitivity s^{max}.

We make two intermediate observations:

Lemma 1: The algorithm correctly computes the safety distances.

Lemma 2: " " " " the maximum sensitivity.

Lemma 1 : The algorithm correctly computes the safety distance D.

Proof : Dijkstra's algorithm on the auxiliary graph G' correctly computes the safety distance for each vertex because

$$\text{length}\begin{pmatrix}\text{Shortest path from}\\ x \text{ to } v \text{ in } G'\end{pmatrix} = \text{length}\begin{pmatrix}\text{Shortest path from } v \text{ to a}\\ \text{motion sensor junction } w \text{ in } G \text{ or } G'\end{pmatrix}$$

$$+$$

$$\cancelto{0}{\text{length}}\left(\text{edge}\,(x, w)\right)$$

∎

Lemma 2: The algorithm correctly computes the maximum sensitivity.

Proof: By correct computation of safety distances (Lemma 1):
a vertex v is safe at sensitivity $s \iff s$ belongs to G_s.

Let $D' = (d_1', d_2', \dots, d_n')$. For any sensitivity value between consecutive entries of D' (i.e., $d_i' \leq s \leq d_{i+1}'$) the structure of G_s does not change.

Thus, it suffices to guess the sensitivity from only among the entries of D'.

Correctness of BFS $\Rightarrow$ our algorithm correctly discovers a safe path for any sensitivity s, whenever such a path exists.

Correctness of binary search $\Rightarrow$ the value s^* is the largest sensitivity for which a safe path exists.

$$\Rightarrow s^* = s^{max}.$$ ∎

(end of proof of Lemma 2)

Finally, from Lemmas 1, 2 and correctness of BFS, it follows that the path returned by algorithm is safe for maximum sensitivity. Thus, the algorithm is correct.

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

* The graphs G, G' and G_s (for any s) all have $O(n)$ vertices and $O(n)$ edges. Thus, BFS / DFS takes $O(n)$ time and Dijkstra's algorithm takes $O(n \log n)$ time.

* Sorting the array D takes $O(n \log n)$ time.

* Binary search considers $O(\log n)$ values of s, and for each such value, BFS and graph construction take $O(n)$ time. This phase takes $O(n \log n)$ time.

Overall, the running time is $O(n \log n)$.