A6Q3

Wrote Muzi Zhao Read Jingbo Yang

(a) Describe the Graph

- **Vertices:** Every pump station with coordinates (x_i, y_i), where $1 \leq i \leq n$. Let's say $V = \{v_1, v_2, \ldots, v_n\}$.
- **Edges:** All the straight bike ways that go directly between any two pump stations. $E = \{(u, v) \mid u, v \in V, u \neq v\}$.
- **Edge Weights:** For each pair (u, v), $u, v \in V$, the edge weight $w(u, v)$ is given by:

  $$w(u, v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2},$$

  which is the Euclidean distance between these two pump stations.

Restate the Problem: Given a graph G (undirected) with V, E, and $w(u, v)$ as described above, we need to find a path from vertex s to vertex t such that the maximum edge weight along the path is minimized.

(b) Prove the Claim

Let P be the $s \to t$ path in the MST T of G. We want to show:

P is a path with the minimum maximum edge weight among all paths from s to t in G.

Proof by contradiction:

- Assume there is a path P' from s to t with edges $\{e_1, e_2, \ldots, e_m\} = E(P')$, and let P have edges $\{e'_1, e'_2, \ldots, e'_k\}$. Assume:

  $$\max\{w(e_1), w(e_2), \ldots, w(e_m)\} < \max\{w(e'_1), w(e'_2), \ldots, w(e'_k)\}.$$

  That is, P' has a smaller maximum edge weight among all the paths from s to t. Moreover:

  $$\forall e \in \{e_1, e_2, \ldots, e_m\}, w(e) < \max\{w(e'_1), w(e'_2), \ldots, w(e'_k)\}.$$

- Let $w(e_{\text{op}}) = \max\{w(e'_1), w(e'_2), \ldots, w(e'_k)\}$.

Consider a cut $(S, V - S)$ by removing e_{op} from T, where S contains node s and $V - S$ contains node t.

- Since P' is also a path from $s \to t$, there exists an edge $e'_{\text{op}} \in \{e_1, e_2, \ldots, e_m\}$ connecting S and $V - S$. By the previous discussion, we know:

  $$w(e'_{\text{op}}) < w(e_{\text{op}}).$$

- In that case, e_{op} is not an edge with the minimum weight crossing the cut $(S, V - S)$, which contradicts the assumption that e_{op} is an edge of T.

Conclusion: For all other paths P' from s to t, we have:

$$\max\{w(e_1), w(e_2), \ldots, w(e_m)\} \geq \max\{w(e'_1), w(e'_2), \ldots, w(e'_k)\}.$$

Thus, P is a path from $s \to t$ with the minimum maximum edge weight among all the paths.

(c) Algorithm

1. **Step 1:** By part (a), we know G is an undirected weighted graph and T is a MST of G. Use Kruskal's algorithm on $G = (V, E)$ to find the edges of T:

   - Sort the edges E by weight.
   - Create an empty set R. For each edge, if adding it does not create a cycle, add it to R; otherwise, skip it.
   - Stop when $|R| = n - 1$, where $n = |V|$.

2. **Step 2:** Using the edges from Step 1 to build the MST T, perform BFS on T starting from node s. Stop when node t is found. The algorithm for BFS is as follows:

```
find_path(T, s, t):
    queue waiting = [(s, [s])]    # Storing start point s and the path to s
    set visited = []              # Keep track of visited nodes
    visited.append(s)
    while waiting is not empty:
        current = waiting.dequeue()
        if current[0] == t:
            return current[1]
        for each neighbor of current in T:
            if neighbor is not in visited:
                visited.append(neighbor)
                waiting.enqueue((neighbor, current[1] + [neighbor]))
    return
```

Output: The function `find_path` will return the path as a set of nodes. If no path is found, it will return nothing.

(d) Time Complexity Analysis

Let's say for $G = (V, E)$, $|E| = m$. We also know $|V| = n$. Since G is completely connected:

$$m = \binom{n}{2} = \frac{n(n-1)}{2} = O(n^2).$$

- **Step 1:** The time complexity of Kruskal's algorithm is $O(m \log n) = O(n^2 \log n)$. Computing weights for each edge will not exceed $O(m) = O(n^2)$.
- **Step 2:** Using BFS on the MST will take $O(n + m) = O(n + n^2) = O(n^2)$, since each edge and vertex will be visited at most $O(1)$. Other operations for each node also take $O(1)$.

Total Time Complexity:

$$O(n^2) + O(n^2 \log n) + O(n^2) = O(n^2 \log n).$$