CS 374 HW 9 Problem 1

quddus2, Aldo Sanjoto, Hieu Huynh

TOTAL POINTS

100 / 100

QUESTION 1

1 1A **30 / 30**

✓ - **0 pts** Correct

- **5 pts** [Data Structure] Incorrect description of MST
- **2 pts** [Data Structure] Minor mistake in description
- **5 pts** [Query algorithm] Incorrect / missing query algorithm to compute distance
- **2 pts** [Query algorithm] Minor mistake in query algorithm
- **15 pts** [Proof of Correctness (PoC)] Incorrect / missing PoC
- **5 pts** [Proof of correctness] Minor mistake in PoC
- **10 pts** [Proof of correctness] Major mistake in PoC, but correct idea
- **2.5 pts** [Complexity] Incorrect space complexity
- **2.5 pts** [Complexity] Incorrect time complexity for construction
- **22.5 pts** IDK
- **30 pts** We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"

QUESTION 2

2 1B **10 / 10**

✓ - **0 pts** Correct

- **2.5 pts** Minor mistake
- **7.5 pts** Major mistake, but correct idea
- **7.5 pts** IDK
- **10 pts** We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"

QUESTION 3

3 1C **10 / 10**

✓ - **0 pts** Correct

- **2.5 pts** Minor mistake
- **7.5 pts** Major mistake, but correct idea
- **7.5 pts** IDK
- **10 pts** We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"

QUESTION 4

4 1D **20 / 20**

✓ - **0 pts** Correct

- **5 pts** Minor mistake in algorithm
- **10 pts** Major mistake in algorithm, but correct idea (binary search, etc)
- **15 pts** IDK
- **20 pts** We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"

QUESTION 5

5 1E **30 / 30**

✓ - **0 pts** Correct

- **5 pts** Did not perform sorting of edges
- **10 pts** [Union find creation] Incorrect creation and initialization of union find data structures (D_i)
- **3 pts** [Union find creation] Minor mistake
- **15 pts** [Updates] Incorrect updates to D_i
- **5 pts** [Updates] Minor mistake in updates to D_i
- **5 pts** Incorrect proof of correctness
- **22.5 pts** IDK
- **30 pts** We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"

Q1)

A) The data structure that we use is Tree with each node is a vertex, and the edge (u,v) in the tree has value $B(u,v)$.

Build the data structure:

- Modify Floy-Warshall Algorithm to find bottle neck distance for all pairs of vertices.

  *Instead of using the original recursion, we use the following recursion formula:

$$dist(i,j,k) = \min \begin{cases} dist(i,j,k-1) \\ \max\{dist(i,k,k-1), dist(k,j,k-1)\} \end{cases}$$

  * The bottleneck distance between vertices v_i and v_j is the value: $dist(i,j,n)$
  * The modified algorithm is still correct because if the path π from v to v that has minimum bottleneck price goes through k, then the minimum bottleneck price of π equal to $\max\{B(\pi_1), B(\pi_2)\}$ with π_1 be the path from v to k that has minimum bottle neck price and π_2 be the path from k to v that has minimum bottle neck price.

- After computing bottle neck distance for all pairs of vertices, we create a new undirected graph G' with $V(G') = V(G)$, and adding edges to G' as follow:

```
For vi ∈ V(G) {
    For vj ∈ V(G) {
        If ( vi ≠ vj && dist(vi, vj, n) < ∞ && (vi, vj) ∉ E(G'))
            Add undirected edge (vi, vj) with
                weight = dist(vi, vj, n) to E(G')
    } }
```

- We run Kruskal Algorithm to find MST of graph G'
- This MST is the data structure that we use to compute $\beta(u,v)$ in $O(n)$ time.
- This MST has n vertices $\Rightarrow$ space $= O(n)$

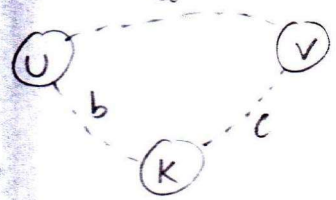
- Query algorithm that computes the distance $\beta(u,v)$:
  - Run DFS on the MST to find path from u to v
  - Traverse through the path to get maximum edge weight return that value, if there is path from u to v
  - Return ∞ if there is no path from u to v
- This take $O(n+m) = O(n)$ since $m = O(n)$

- Proves of correctness:
  - Let a be the maximum edge weight of edges in Path from u to v in MST

Claim: $\beta(u,v) = a$

Prove by contradiction:

Suppose in graph G', there is a path from u to k with bottle neck price $= b$ and path from k to v with bottle neck price $= c$ and $a > \max(b,c)$



Then, when we run Kruskal algorithm, all edges in path from u to k & from k to v will be considered before the edge with weight $= a$ in path from u to v.

$\Rightarrow$ The edge with weight $= a$ will not be added to the MST

$\Rightarrow$ This is the contradiction

- The running time for computing the data structure is $O(n^3)$ because
  - Floy-Warshall Algorithm take $O(n^3)$
  - Creating Graph G' take $O(n^2)$
  - Computing MST take $O(m \cdot n) = O(n^3)$ because $m = O(n^2)$

σ

1 1A **30 / 30**

✓ **- 0 pts** **Correct**

- 5 pts [Data Structure] Incorrect description of MST

- 2 pts [Data Structure] Minor mistake in description

- 5 pts [Query algorithm] Incorrect / missing query algorithm to compute distance

- 2 pts [Query algorithm] Minor mistake in query algorithm

- 15 pts [Proof of Correctness (PoC)] Incorrect / missing PoC

- 5 pts [Proof of correctness] Minor mistake in PoC

- 10 pts [Proof of correctness] Major mistake in PoC, but correct idea

- 2.5 pts [Complexity] Incorrect space complexity

- 2.5 pts [Complexity] Incorrect time complexity for construction

- 22.5 pts IDK

- 30 pts We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"

(Q1B) Prove by induction on i

Base case: $i = 1$:

This case is true because we compute only 1 MST by using Kruskal algorithm and because $L_1(m)$ is set of all edges

$\Rightarrow$ All edges in $F_1(m)$ are edges of MST T_1

IH: Assume for $i < K$, we have $F_1(m), F_2(m) \ldots F_i(m)$ are edges of MST $T_1, T_2, \ldots T_i$.

Need to prove: $F_{i+1}(m)$ are edges of MST T_{i+1}.

- We have $L_{i+1}(m) = L_i(m) \setminus F_i(m)$

$$= (L_{i-1}(m) \setminus F_{i-1}(m)) \setminus F_i(m)$$

$$= (((L_1(m) \setminus F_1(m)) \setminus F_2(m)) \ldots \setminus F_i(m))$$

- In other word, $L_{i+1}(m)$ are set of all edges that are not contained in $F_1(m), F_2(m), \ldots F_i(m)$

- Executing Kruskal on the set of edges $L_{i+1}(m)$ is similar to executing Kruskal algorithm on graph after removing all edges in $T_1, T_2, \ldots, T_i$ because of IH

$\Rightarrow$ This will give MST T_{i+1}

Therefore edges in $F_{i+1}(m)$ are edges of tree T_{i+1}.

2 1B 10 / 10

✓ - 0 pts Correct

- 2.5 pts Minor mistake
- 7.5 pts Major mistake, but correct idea
- 7.5 pts IDK
- 10 pts We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"

1C) Prove by contradiction

- Assume u, v are in same connected component of $(V, F_i(t))$ and u, v are not in same connected component of $(V, F_j(t))$ with $j < i$

- u and v are not in same connected component of $(V, F_j(t))$ means that in set $L_j(t)$, there are no edges that can create path from u to v

We also have, $L_i(t) = ((L_j(t) \setminus F_j(t)) \setminus F_{j+1}(t) \setminus \dots \setminus F_{i-1}(t))$

$\Rightarrow L_i(t) \subset L_j(t)$

$\Rightarrow$ Edges in $L_i(t)$ cannot create path from u to v

$\Rightarrow$ Executing Kruskal on $L_i(t)$ will not be able to have u and v in the same connected component

$\Rightarrow$ u, v are not in same connected component of $(V, F_i(t))$

$\Rightarrow$ Contradiction!

Therefore, if u, v in same connected component of $(V, F_i(t))$, u, v are also in same connected component of $(V, F_j(t))$ for all $j < i$

3 1C 10 / 10

✓ - 0 pts Correct

- 2.5 pts Minor mistake

- 7.5 pts Major mistake, but correct idea

- 7.5 pts IDK

- 10 pts We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"

(Q1D)

```
Foo_D (D1(t), D2(t), ....., U't+1 , V't+1 ) {  // et+1 = U't+1 V't+1
    left ← 1
    right ← number of D (t) that we have
    While ( left < right − 1 000 ) {
        mid = ⌊ (left + right) / 2 ⌋
        If ( D(t)mid . Find ( U't+1 ) == D(t)mid . Find ( V't+1 ) ) {
            left = mid + 1;
        }
        else {
            right = mid;
        }
    }
    For i ← left to right {
        If ( Di(t).Find ( U't+1 ) == Di(t). Find ( V't+1 ) ){
            continue;
        }
        Else {
            return i ;
        }
    }
    return −1 ; // This means that V't+1 and U't+1
                   are in same connected component
                   for all V(F(t))
}
```

This takes $O(\log(n) \cdot \alpha(n))$ because this is same as binary search!

4 1D **20 / 20**

✓ **- 0 pts** Correct

- 5 pts Minor mistake in algorithm

- 10 pts Major mistake in algorithm, but correct idea (binary search, etc)

- 15 pts IDK

- 20 pts We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"

(Q1 E)

```
Compute_All_MST {
  • Sort all edges based on cost and store them in array E[1...m]
  • k ← 0;
  • F1 ← Empty set
  • D1 ← Empty Union-Find structure
  For i ← 1 to m {
      e = (u, v) ← E[i]                    // Foo-D() is the function in part (D)
      j ← Foo-D(D1, D2, ... Dk, u, v);
      If (j == -1) {
          Create new set F_{k+1} = {(u, v)}
          Create new Union-Find structure D_{k+1}
          D_{k+1}.Make Set(u);
          D_{k+1}.Make Set(v);
          D_{k+1}.Union(u, v);
          k ← k+1;
      }
      Else {
          If (Dj.Find(u) == NULL) {
              Dj.Make set(u);
          }
          If (Dj.Find(v) == NULL) {
              Dj.Make set(v);
          }
          Dj.Union(u, v);
          Fj = Fj ∪ {(u, v)};
      }
  }
  j ← 0;
  For i ← 1 to k {
      If (Fi.size() == (n-1)) {
          j++ ; Tj ← Fj
      }
  }
}
```

MSTs are stored in $T_1, T_2, \ldots T_j$

5 1E **30 / 30**

✓ **- 0 pts** Correct

- 5 pts Did not perform sorting of edges

- 10 pts [Union find creation] Incorrect creation and initialization of union find data structures (D_i)

- 3 pts [Union find creation] Minor mistake

- 15 pts [Updates] Incorrect updates to D_i

- 5 pts [Updates] Minor mistake in updates to D_i

- 5 pts Incorrect proof of correctness

- 22.5 pts IDK

- 30 pts We are unable to follow the logic of the answer, or the answer is just way too long. In the future, you might want to consider using "IDK"