Homework 2

Thomas Kim tsk389 51835

David Munoz dam2989 51840

CS331 Algorithms and Complexity

Problem Q1(a). What is the number of topological orderings in this directed graph?
Solution: 2^{n+1}

Problem Q1(b). Let T be a tree such that every node in T has either 2 children or 0 children. If T has $n \geq 1$ leaves, prove that the total number of nodes in T is $2n - 1$.

Proof. **Base case:** (n = 1)
The tree must have precisely 1 node, $2n - 1 = 2 - 1 = 1$.
Suppose the tree has more than 1 node.
By the definition of tree, the graph must be connected.
The tree has precisely 1 leaf $l \implies$ every other node must have 2 children.
Suppose l is connected to p, which must have 2 children.
Each descendent of p must have 2 children, meaning the tree must have an infinite number of nodes.
Contradiction
Induction Hypothesis:
Suppose a tree T has n leaves.
Remove 2 leaves with the same parent to form tree T'.
By the definition of a leaf, each leaf is connected to precisely 1 other node (parent).
By removing 2 leaves, precisely 2 edges are removed.
$\therefore$ the number of nodes in T' is $2n - 3$.

□

Problem Q2. Let s be the vertex of a connected undirected graph G. Let $T^B_{G,s}$ and $T^D_{G,s}$ respectively be the trees obtained by running BFS and DFS on graph G starting at node s. Prove

$$T^B_{G,s} \equiv T^D_{G,s} \implies G \text{ is acyclic}$$

Lemma 1: $T^B_{G,s} \equiv T^D_{G,s} \implies T^B_{G,s}$ has no non-tree edges.

Proof. Suppose vertices $v_1, v_2 \in T^B_{G,s}$ are connected by a non-tree edge e.
In DFS, either v_1 or v_2 must have been explored first, because vertices are explored in a strictly non-parallel fashion.
Let v_p be the vertex which was explored first in DFS, and v_c be the other vertex.
If v_p was explored first, v_c must not have been explored when v_p was added to the DFS tree.
Since v_c was not explored, the edge $e = (v_p, v_c)$ must be added to the DFS tree by the DFS algorithm.
$\therefore e \in T^D_{G,s} \wedge T^B_{G,s} \implies T^B_{G,s} \not\equiv T^D_{G,s}$

□

Proof. By lemma 1, $T^B_{G,s}$ has no non-tree edges.
This implies $T^B_{G,s} = G$
By definition, if G is a tree, G is acyclic.

□

Problem Q3. Given n images and m unambiguous matches, design an algorithm that runs in $O(m+n)$ time and uniquely labels n images as either A or B, such that two images reported to be the same by ImgComp get the same label, and two images reported to be different by ImgComp get different labels. Prove the algorithm's runtime and correctness.

```
1: Let $I$ be the set of images such that $|I| = n$
2: Let $U$ be the set of unambiguous matches such that $|U| = m$
3: for $\forall i_x \in I$ do
4:     Create a node $v_x$ corresponding to $i$ in graph $G$
5: end for
6: for $\forall (i_x, i_y) \in U$ do
7:     if The match returns "DIFFERENT" then
8:         Create an edge between the nodes $v_x$ and $v_y$ in graph $G$
9:     end if
10: end for
11: $T = \emptyset$
12: while $\exists s \in G$ such that $s$ has not been explored by BFS do
13:     Execute BFS starting at $s$ to generate subtree $t_j$
14:     $T = T \cup \{t_j\}$
15: end while
```

Problem Q4. Express the $n + 1$ person ballroom murder problem as a graph problem and provide an efficient way to detect inconsistencies

Graph construction algorithm

```
1: Let $P$ be the set of people
2: Let $D$ be the set of statements of the form "$i$ danced with $j$"
3: Let $L$ be the set of statements of the form "$i$ saw $j$ leaving"
4: Let $E$ be the set of statements of the form "$i$ saw $j$ entering"
5: Let $G_1, G_2, G_3$ be empty graphs
6: for $p \in P$ do
7:     Add $p$ to $G_1, G_2, G_3$ as a vertex $v_{1,1}$ and $v_{2,1}$ respectively
8: end for
9: for $l \in L$ do
10:     Add a directed edge between $v_{1,i}$ and $v_{1,j}$
11: end for
12: for $l \in E$ do
13:     Add a directed edge between $v_{2,i}$ and $v_{2,j}$
14: end for
15: for $d \in D$ do
16:     Add a directed edge between $v_{3,i}$ and $v_{3,j}$
17: end for
```

Inconsistency detection algorithm

```
1: $T_1, T_2 = \emptyset$
2: while $\exists s \in G_1, s$ has not been explored yet do
3:     Perform BFS on $G_1$ starting on $s$ to generate subtree $t_{1,k}$
4:     $T_1 = T_1 \cup \{t_{1,k}\}$
5: end while
6: while $\exists s \in G_2, s$ has not been explored yet do
7:     Perform BFS on $G_2$ starting on $s$ to generate subtree $t_{2,k}$
8:     $T_2 = T_2 \cup \{t_{2,k}\}$
9: end while
10: Use BFS to check for cycles in $G_1, G_2$
```

```
11: if $\exists$ cycle in $G_1$ or $G_2$ then
12:     Inconsistency
13: end if
14: for $\forall v \in G_1$ do
15:     Ensure all outgoing edges $e$ from $v$ in $G_3$ do not correspond to an edge between adjacent levels in
        any $t_{i,k} \in T_1, T_2$
16:     Remove $e$ from $G_3$
17: end for
```

Inconsistency 1: Odd length cycles in G_1 or G_2 imply inconsistencies with the population occupying the room.

Proof. Suppose there is a cycle in G_1 or G_2.
This means $\exists p_1, p_2, \ldots, p_n \in G \mid |p_x$ entered after p_{x+1} left $\wedge p_n$ entered after p_1 left
$\implies$ p_1 entered after p_k left, $k > 1$, particularly p_1 entered after p_n left.
This is inconsistent, as p_1 cannot have entered after p_n left and p_n entered after p_1 left, as each person entered the ballroom only once.

□

Inconsistency 2: Edges present in G_3 which connect nodes in different layers of T_1 or T_2 imply inconsistency.

Proof. Suppose the graphs G_1, G_2 are acyclic. If they had cycles, there is already an inconsistency.
Suppose two nodes are in different layers of G_1 or G_2
This means these two nodes could not have been in the room at the same time, since a parent cannot be in the same room as any child in its subtree.
Suppose there is an edge present in G_3 which connected nodes in different layers of T_1 or T_2.
This means two people who were never in the same room at the same time danced together, which is impossible.

□

Proof of runtime:

Proof. Let the number of statements be a
The graph construction constructs three graphs each with $n+1$ nodes.
The graph iterates through each statement once, creating a edges.
Therefore the number of operations taken to construct the graph is $O(3n + a + 3)$
The inconsistency detection algorithm performs BFS twice, which is $O(2m + 2n) = O(2n + 2 + 2a)$
The inconsistency detection algorithm checks for cycles during BFS by checking for any non-tree edges, which does not contribute to the complexity.
The inconsistency detection algorithm then iterates through each vertex in T_1, T_2 and each node/edge in G_3, taking $O(3n + 3 + a)$
The final complexity is $O(3n + a + 3 + 2n + 2 + 2a + 3n + 3 + a) = O(8n + 4a + 8)$, which is linear complexity.

□