Written Problem Set 1: Graphs

Response by: **Tianze Ren(tr2bx), Ziang Zhang(zz9fs), Mingye Li(fky9xf), Ziyue Xu(rca8vx)**

Collaborators: Tianze Ren(tr2bx), Ziang Zhang(zz9fs), Mingye Li(fky9xf), Ziyue Xu(rca8vx)

Resources: None

Problem 1: Martian Morels

Input A 2-d array with each cell containing either “same”, “different” or “uncertain” representing the decision of the Martian for each pair.

Output A boolean indicating whether the given set of decisions is consistent.

Algorithm We modeled this problem as a graph where each node($S0, ..., Sn - 1$) connected to a mushroom sample and the edge between two nodes indicates the Martian’s test result. After that, We will assign the weight (+1) to an edge if the Martian said the samples were "same", (0) if the Martian was "uncertain", and (-1) if they said the samples were "different”.
Next, we labeled each node as "edible" or "toxic". To start, we assigned an arbitrary label to one node and used a depth-first search to propagate labels to the other nodes. Specifically, we marked the starting node as "edible" and the opposite label to all its neighbors. We then recursively propagate labels to the neighbors of those neighbors and so on until either all nodes are labeled or we encounter a conflict (i.e., a node has two conflicting labels). If we encounter a conflict, then the set of decisions is inconsistent and we can return false. Otherwise, if all nodes are labeled, then we will return true.

Correctness Suppose that there exists a labeling of the nodes such that all the Martian’s decisions are consistent. We will observe that if we label one node as "edible", we can uniquely determine the labels of all its neighbors based on the Martian’s decisions. Specifically, if the Martian said two nodes were "same", then they must have the same label; if the Martian said two nodes were "different", then they must have opposite labels; and if the Martian was "uncertain", then we can assign either label to the other node. Therefore, our depth-first search will propagate consistent labels to all nodes that are reachable from the starting node. Furthermore, since there exists a consistent labeling, our algorithm will not encounter a conflict and will terminate with a valid labeling.
Suppose that the set of decisions is inconsistent. We claim that our algorithm will detect this and return false. We can observe that if there is no consistent labeling, then there must be some pair of nodes with conflicting labels that are adjacent in the graph. This is because if all pairs of adjacent nodes had the same label, then we could simply assign that label to all nodes and satisfy all of the Martian’s decisions. Similarly, if all pairs of adjacent nodes had opposite labels, then we could simply assign one of the labels to all nodes and satisfy all of the Martian’s decisions. Therefore, when we propagate labels using our depth-first search, we will eventually encounter a conflict and terminate with false.

Running Time The DFS runs in time $O(n + m)$, when m is the number of edges in the graph. To run all pairs of samples based on the circumstances, it takes $O(n^2)$. Therefore, the overall runtime is $O(n^2 + m)$. In the worst case, every single pair of samples is tested so it takes $O(n^2)$ time. But in reality, m should be smaller than n^2, so the runtime should be $O(m + n^2)$.

Problem 2: Dog Walking

Algorithm TODO: Provide a precise description of your algorithm here

Correctness TODO: Justify the correctness of your algorithm here (a formal proof is not necessary, just explain precisely why you believe your algorithm to be correct)

Running Time TODO: Give and justify a Θ bound on your algorithm's worst-case running time here (a formal proof is not necessary, just explain precisely why you believe your algorithm to have this running time).

Algorithm: First, find the shortest path from $H1$ to $P1$, label it $[A1, A2, A3...An]$, and the shortest path from $H2$ to $P2$, label it $[B1, B2, B3...Bm]$. Update these paths after each move with the same labels. Build a list of pairs to record the position of two dogs at each step. Calculate the path distance between them. If the distance of the shortest path between the two dogs is greater than 1, continue.
If the distance of the shortest path between the two dogs equals 1, there are three options. Always manipulate the longer shortest path. If two paths have equal distances, randomly choose one. Assume the path labeled A is the longer one:

- 1. Starting from $A1$, cut the edge connecting $A1$ and $A2$, iterate through all adjacent nodes, and find the shortest path for this dog again. Label it $[A1, A'2 \ldots A'n]$. If the number of nodes in this shortest path is the same as the number from $A2$ to An, and the shortest path from $A'1$ to $B2$ is greater than 1, choose this new path. If not, go to option two.

- 2. Wait at A1 for one step, let the other dog move to $B2$, then proceed. If the shortest path from $A1$ to $B2$ is smaller than 2, go to option 3.

- 3. Iterate through all adjacent nodes other than $A2$, proceed to this node if this adjacent node of A1 has a shortest path to $B2$ that has distance greater than 1. Then return to $A1$ as the other dog proceed from $B2$ to $B3$. If no adjacent node meets this condition, go through these 3 options for path labeled B.

If the distance of the shortest path between the two dogs equals 0, there are two options. Always manipulate the longer shortest path. If two paths have equal distances, randomly choose one. Assume the path labeled A is the longer one:

- 1. Cut the edge connecting $A1$ and $A2$, iterate through all adjacent nodes at $A1$, and find the shortest path for this dog again. Label it $[A'1, A'2 \ldots A'n]$. If the number of nodes in this shortest path is the same as the number from $A2$ to An, and the shortest path from $A'1$ to $B2$ is greater than 1, choose this new path. If not, go to option two.

- 2. Iterate through all adjacent nodes other than $A2$, proceed to this node if this adjacent node of $A1$ has a shortest path to $B2$ that has distance greater than 1. Then return to $A1$ as the other dog proceed from $B2$ to $B3$. If no adjacent node meets this condition, go through these 3 options for path labeled B.

Return the list of pairs.

Time complexity: $O(n^3)$. There are 3 shortest path algorithms at each step, and there are n nodes in total. Therefore the time complexity is. $O((3 * n^2) * n)$. There could be new shortest path at each step, but this n^2 would not exceed the set $O(n^3)$.

Correctness: The two dog walkers are never adjacent or in the same location in the schedule, because according to the algorithm, we would do one of the following: find a new path where overlap/adjacency does not exist, wait until the other dog walker walk away (until a distance > 2), or the dog walker will return back to a prior node that is of distance > 2 from the current location of the other dog walker.

When there are no intersection or adjacency between the two paths, we only need to find the shortest path of each dog walker respectively; When intersection or adjacency does exist, we are taking approaches from the most path-efficient to the least ones: first, try to find a new shortest path of the same distance; second, if first approach fails, the shorter shortest path will wait at the same location while the longer one moves forward. In this case, one more step is needed for each change compared to the most ideal situation. Third, if the second approach fails, the shorter shortest path will step back, while the longer one moves forward. In this case, two more steps are needed for each change compared to the most ideal situation. For each of the decisions made, we only compromise and take extra step(s) only if the most ideal situation is not approachable. Therefore, the schedule is proved to be the shortest.

Problem 3: Is it connected?

Let $G = (V, E)$ be an undirected graph such that $|V| = n$. Either prove the claim below or else find a counterexample.

Claim If every member of V has a degree of at least $\frac{n}{2}$ then G must be connected.

Proof/Counterexample TODO: Include your proof or counterexample below. Be sure to state which you are providing, and your proof strategy (if that's the direction you're answering).

Proof by induction

Base case: If $|V| = 1$, there is only one node, and it is trivially connected.

Inductive steps: Assume that every graph $G = (V, E)$ with $|V| = n = 2m$, where $m \geq 1$, and every node has a degree of at least $\frac{n}{2} = m$ edges, is connected. Let $G1 = (V1, E1)$ be a graph with $|V1| = n+1 = 2m+1$, where every node has a degree of at least $\frac{n+1}{2} = m + 1$ edges.

Choose any node v in $G1$. Since $deg(v) \geq m + 1$, there are at least $m + 1$ edges incident to v. Let $G2 = (V2, E2)$ be the subgraph of $G1$ induced by the vertices adjacent to v, and note that $|V2| = deg(v) \geq m + 1$. By the induction hypothesis, $G2$ is connected.

For any vertex u in $G2$ and any vertex w outside of $G2$. Since u is adjacent to v, and v is adjacent to w, there is a path from u to w. Therefore, the graph is connected, and the claim is true.

Problem 4: Asymptotics

Using the definition of O, Ω, and Θ in class, demonstrate each of the following:

a. $n \in O(n^2)$

b. $n^2 \notin O(n)$

c. $\sqrt{2n} \in \Omega(2\sqrt{n})$

d. $\max(n, m) \in \Theta(n + m)$

TODO: give a proof for each item below:

a.

To demonstrate that $n \in O(n^2)$, we need to show that there exist positive constants c and $n0$, such that for all $n \geq n0$, n is bounded above by $c * n^2$.

Formally, we need to prove that: $n \leq c * n^2$ for all $n \geq n0$.

We can prove this by choosing $c = 1$ and $n0 = 1$, and then we have: $n \leq n^2$ for all $n \geq 1$.

This statement is clearly true, since n^2 is always greater than or equal to n for any positive integer n.

Therefore, we can conclude that $n \in O(n^2)$, since we have found the required constants c and n0 such that n is bounded above by $c * n^2$ for all $n \geq n0$.

b.

To demonstrate that $n^2 \notin O(n)$, we need to show that for any positive constant c and any $n0$, there exist values of n greater than or equal to $n0$ such that n^2 is not bounded above by $c * n$.

Formally, we need to show that:

$n^2 > c * n$ for infinitely many values of $n \geq n0$.

Let's assume for contradiction that there exist constants c and $n0$ such that $n^2 \leq c * n$ for all $n \geq n0$.

We can rearrange this inequality to get: $n \leq c$ for all $n \geq n0$.

However, this contradicts the fact that n can be arbitrarily large and still satisfy $n \geq n0$. Therefore, there do not exist constants c and $n0$ that satisfy $n^2 \leq c * n$ for all $n \geq n0$, which implies that $n^2 \notin O(n)$.

c.

To demonstrate that $\sqrt{2n} \in \Omega(2\sqrt{n})$, we need to find positive constants c and $n0$ such that for all $n \geq n0$, $\sqrt{2n}$ is bounded below by $c * 2\sqrt{n}$.

Formally, we need to show that:

$\sqrt{2n} \geq c * 2\sqrt{n}$ for all $n \geq n0$.

We can simplify this inequality by dividing both sides by $2\sqrt{n}$:

$\sqrt{2/n} \geq c$

Now, we can choose $c = 1/2$ and $n0 = 1$, and then we have:

$\sqrt{2/n} \geq 1/2$ for all $n \geq 1$.

This inequality is true for all n, since $2/n$ is a decreasing function as n increases, and its limit as n approaches infinity is 0. Therefore, we can conclude that $\sqrt{2n} \in \Omega(2\sqrt{n})$, since we have found the required constants c and $n0$ such that $\sqrt{2n}$ is bounded below by $c * 2\sqrt{n}$ for all $n \geq n0$.

d.

To demonstrate that $\max(n, m) \in \Theta(n + m)$, we need to show that there exist positive constants $c1$, $c2$, and $n0$ such that for all $n, m \geq n0$, $\max(n, m)$ is bounded both above and below by $c1 * (n + m)$ and $c2 * (n + m)$, respectively.

Formally, we need to show that: $c1 * (n + m) \leq \max(n, m) \leq c2 * (n + m)$ for all $n, m \geq n0$.

First, we can show that $\max(n, m) \leq (n + m)$, since the maximum of two numbers cannot be greater than their sum. Therefore, we have: $\max(n, m) \leq (n + m)$

Next, we can show that $(n + m) \leq 2 * \max(n, m)$, since the sum of two numbers cannot be greater than twice the maximum of those two numbers. Therefore, we have: $(n + m) \leq 2 * \max(n, m)$

Combining these inequalities, we get: $\max(n, m) \leq (n + m) \leq 2 * \max(n, m)$

Now, we can choose $c1 = 1/2$, $c2 = 2$, and $n0 = 1$, and then we have:

$1/2 * (n + m) \leq \max(n, m) \leq 2 * (n + m)$ for all $n, m \geq 1$

This inequality is true for all n and m, since the left-hand side is always less than or equal to $\max(n, m)$ which is always less than or equal to the right-hand side. Therefore, we can conclude that $\max(n, m) \in \Theta(n + m)$, since we have found the required constants $c1$, $c2$, and $n0$ such that $\max(n, m)$ is bounded both above and below by $c1 * (n + m)$ and $c2 * (n + m)$, respectively, for all $n, m \geq n0$.