CSE 421 Spring 2025: Set 2

Instructor: Chinmay Nirkhe
Due date: April 16th, 2025 11:59pm

Instructions: Solutions should be legibly handwritten or typeset (ideally in LaTeX). Mathematically rigorous solutions are expected for all problems unless explicitly stated.

You are encouraged to collaborate on problems in small teams but everyone must individually submit solutions. Solutions for the problems may be found online or in textbooks – but do not use them.

For grading purposes, list, with each problem, the names of your collaborators. Please start each problem on a new page.

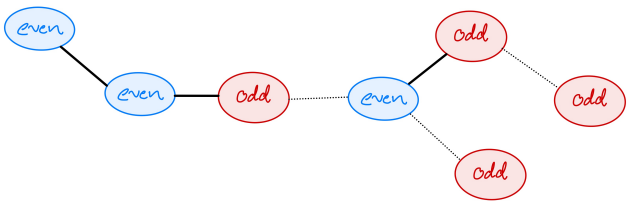
Notation: Graphs are assumed to have n vertices and m edges in this problem set.
Notation: "Iff" is a portmanteau of "if and only if" commonly used in mathematics.

Problem 1 (Graph property)**. [5 points]** A DFS tree is the tree constructed by running the DFS algorithm and recording which edges are used in the search. *We define, for this problem*[1]*, a back edge as any edge from a vertex to its ancestor in the tree (including itself and it's parent).* Prove that if a directed connected graph has a back edge with respect to any DFS tree, then it has a directed cycle. Second, prove that every directed cycle in a directed graph must include at least one back edge with respect to any DFS tree.

Problem 2 (Trees)**. [5 points]** Given a connected tree T with $n \geq 2$ vertices, suppose each vertex of the tree is marked with either "odd" or "even" with an even number of vertices marked as "odd". Prove there exists a subset F of the edges of T such that each "odd" marked vertex is adjacent to an odd number of edges in F and every "even" marked vertex is adjacent to an even number of edges in F. In the example below, the dashed edges would form a valid F.

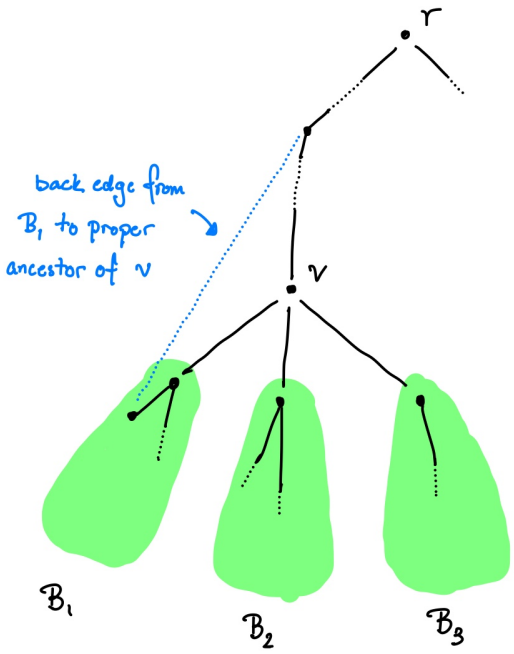
Problem 3 (Articulation points)**.** For an undirected connected graph $G = (V, E)$, a vertex v is defined as an **articulation point** if the removal of v and its adjacent edges *disconnects* the graph into two or more connected components. We will construct a $O(n+m)$-time algorithm for finding *all* the articulation points of a connected graph.

[1]The literature, including the textbooks for this course, tends to vary in its definition of a back edge. Some literature will call a self-loop from $v \to v$ a back edge. Some will not. Some literature will distinguish this case from other ancestors by referring to any edge to a parent or further back as a proper ancestor. I too, often confuse the definitions. We will attempt to make it clear in our problem statements, like so.



1. **[5 points]** Let G be a connected graph with at least two vertices. Let T be a DFS tree of G rooted at r. Prove that r is an articulation point iff r has at least two children in T.

2. **[5 points]** Let v be a vertex in the graph and let $B_1, \ldots, B_k$ be the subtrees of the DFS tree T descending from v. Prove that v is **not** an articulation point iff for each subtree B_i, there exists a back edge from B_i to a proper ancestor[2] of v in T.



3. **[5 points]** Enumerate the vertices, $\#(v)$, in the order they are visited by the DFS algorithm generating T from root r. For any vertex v, let B_v be the subtree of T rooted at v. Define $\text{low}_T(v)$ as the minimum of $\#(v)$ and $\#(u)$ for any proper ancestor u of v connected to B_v by a single back edge.

   Construct an algorithm[3] for computing $\text{low}_T(v)$ for the DFS tree T rooted at r that runs in time $O(n+m)$. Prove runtime and correctness.

[2] A proper ancestor of v is a vertex along the path between v and r in T not including v.

[3] You do not need to reinvent the wheel here. If your algorithm is a minor modification of a known algorithm such as DFS, you can simply write, "We modify DFS to calculate [blank] after we have traversed its children."

4. **[5 points]** Having computed $\text{low}_T(v)$ for every vertex for a DFS tree rooted at T, use the property proven in Part 2 to compute all the articulation points of the graph in time $O(n+m)$. Prove runtime and correctness.

Problem 4 (Bug detection)**. [15 points]** An entomologist has discovered a new collection of bugs. However, she notices that if two bugs are placed in a terrarium together, they either are hostile to each other or not. She hypothesizes that she has actually discovered two species that are visually indistinguishable but are docile with other members of the same species and hostile to members of the other species.

Without loss of generality, they have decided that Bug 1 is of Species A. Her graduate students have placed various pairs of bugs (x_i, y_i) in the terrarium and recorded their interaction as either hostile or docile.

Given a list of m data points $(x_i, y_i, \text{hostile/docile})$ consisting of n bugs, construct an algorithm which outputs either:

1. An assignment of each bug $1 \ldots n$ as either species A, species B, or "insufficient information" assuming the entomologist's hypothesis.

2. An output of "hypothesis false" if the collected data is inconsistent with the hypothesis.

Prove runtime and correctness.

Note: The hypothesis could be proven false, even if some bug's species cannot be concretely determined.

[2 extra credit points[4]**]** The entomologist wants to decided if there could be three species s.t. each species is hostile to members of the other species but docile with members of its own species. Show that deciding if the data is consistent with three species is NP-complete.

[4]Extra credit points are evaluated after initial grades are determined and can only boost your score. However, the number of extra credit points in this course is so small compared to the number of actual points, that for all intents and purposes, it will not change your grade. You should attempt extra credit problems as a labor of love.