CSORE 4231 ANALYSIS OF ALGORITHMS I Homework 4

Francis Zhang xz3279

September 19, 2024

1. Dynamic Binary Search

(a) Describe how to perform the `SEARCH` operation for this data structure. Analyze its worst-case running time.

(b) Describe how to perform the `INSERT` operation. Analyze its worst-case and amortized running times assuming that the only operations are Insert and Search.

(c) Describe how to implement `DELETE`. Analyze its worst-case and amortized running times assuming that there can be `DELETE`, `INSERT`, and `SEARCH` operations.

2. BFS and DFS

(a) CLRS 22.2-7.
There are two types of professional wrestlers: "babyfaces" ("good guys") and "heels" ("bad guys"). Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have a list of r pairs of wrestlers for which there are rivalries. Give an $O(n + r)$-time algorithm that determines whether it is possible to designate some of the wrestlers as babyfaces and the remainder as heels such that each rivalry is between a babyface and a heel. If it is possible to perform such a designation, your algorithm should produce it.
Using results we have seen in class prove correctness and runtime of your algorithm.

(b) CLRS 22.3-8.
Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, and if $u.d < v.d$ in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.

(c) CLRS 22.3-9.
Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, then any depth-first search must result in $v.d \leq u.f$.

3. Strongly connected components

(a) CLRS 22.5-5.
Give an $O(V + E)$-time algorithm to compute the component graph of a directed graph $G = (V, E)$. Make sure that there is at most one edge between two vertices in the component graph your algorithm produces.
Prove correctness and runtime of your algorithm. Recall that you can use any result we have seen in class.

(b) Give an $O(V + E)$-time algorithm that, given a directed graph $G = (V, E)$, constructs another graph $G' = (V, E')$ such that G and G' have the same strongly connected components, G' has the same component graph as G, and E' is as small as possible. Prove that your algorithm meets these requirements and that it runs in the correct time.

4. Minimum Spanning Trees

(a) Provide a connected graph example where the set of light edges across cuts doesn't form a minimum spanning tree. Give a simple example of a connected graph such that the set of edges $\{(u,v)$: there exists a cut $(S, V \setminus S)$ such that (u, v) is a light edge crossing $(S, V \setminus S)\}$ does not form a minimum spanning tree.

(b) Show a graph has a unique minimum spanning tree if for every cut, there's a unique light edge crossing the cut. Counterexample the converse.

(c) Consider a new divide-and-conquer algorithm for computing minimum spanning trees, which goes as follows:
Given a graph $G = (V, E)$, partition the set V of vertices into two sets V_1 and V_2 such that $|V_1|$ and $|V_2|$ differ by at most 1. Let E_1 be the set of edges that are incident only on vertices in V_1, and let E_2 be the set of edges that are incident only on vertices in V_2. Recursively solve a minimum-spanning-tree problem on each of the two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Finally, select the minimum-weight edge in E that crosses the cut (V_1, V_2), and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.
Either argue that the algorithm correctly computes a minimum spanning tree of G, or provide an example for which the algorithm fails.