CS 535 Design and Analysis of Algorithms **Fall Semester, 2025**

Homework 4

Assigned: September October 21 *Due: November 4*

All students: please use Canvas to submit your solutions.
Notes for pseudocode usage:

1. C/Java instructions are fine. But do not write object-oriented additions. Do not declare or use any class. Declare only procedures (if necessary) and explain in words what each procedure does, and what is the use of each parameter. Feel free to use as procedures algorithms from the textbook; indicate page and edition. Please avoid Python as the running time can be obscured.
2. One instruction per line
3. Match the brackets with a horizontal line
4. Number your lines
5. Write down if your array is indexed $0 \ldots n-1$ or $1 \ldots n$.

Problem 1 When analyzing splay trees, we have defined the rank function on nodes: $r(v) = \lfloor \lg |V(T_v)| \rfloor$, where lg is the base 2 logarithm, and T_v is the subtree rooted at v (v and all its descendants).

We also called $r'(v)$ as being the rank of v after one operation - a double rotation or a single rotation. We proved that the amortized cost of a double rotation at x does not exceed $3(r'(x)-r(x))$, and that the amortized cost of a single rotation at x does not exceed does not exceed $1+3(r'(x)-r(x))$.

This problem asks to show that the $+1$ term for a single rotation is needed. In other words, give an example of a binary search tree and a single rotation whose amortized cost as defined in class (actual cost plus the sum of ranks after the operation minus the sum of ranks before the operation) exceeds $3(r'(x)-r(x))$.

As an aside (no need to turn in anything about this paragraph), with single rotations only, the splay operations will not have amortized cost of $O(\lg n)$ each.

Problem 2 Suppose we implement the UNION-FIND Data structure with union-by-rank but no path-compression. For every n, give an example of a sequence of operation with n MAKE-SET(), p UNION and r FIND operations whose total running time exceeds $c(n+p+r)\log n)$ for some constant c. We also require that each element is a parameter in FIND() at most once in this sequence (any element can appear many times in UNION(,) operations).

Problem 3 Given a $O(|V|+|E|)$ algorithm to determine if an undirected graph $G=(V,E)$ is bipartite. Prove that your algorithm is correct. The definition of a bipartite graph apears in the textbook. Hint: modifiy BFS or DFS.

Problem 4 The input consists of n currencies c_1, c_2, $\cdots$, c_n and an $n \times n$ matrix Q of exchange rates, such that one unit of currency c_i buys $Q[i,j]$ units of currency c_j.

1. Give an $O(|V|^3)$ algorithm to determine whether or not there exists a sequence of currencies $< c_{i_1}, c_{i_2}, \cdots, c_{i_{k-1}}, c_{i_k} >$ such that
$$Q[i_1,i_2] \cdot Q[i_2,i_3] \cdots Q[i_{k-1},i_k] \cdot Q[i_k,i_1] > 1.$$
(as an aside, one can make guaranteed profit if such a sequence exists!).
Discuss correctness and running time.
2. Within the same time bounds, output such a sequence if it exists.