Due Date: October 20, 2015

Problem 1: Consider the Union-Find data structure discussed in the class. Suppose all $UNION$ operations are performed before any of the $FIND$ operations, and whenever $UNION(x, y)$ is called, x, y are the leaders of their sets (so you don't have to find the leaders of x and y). Describe an algorithm that performs $UNION$ and $FIND$ operations in $O(1)$ amortized time.

Problem 2: Chicago has many tall buildings, but only some of them have a clear view of Lake Michigan. Suppose we are given an array $A[1..n]$ that stores the height of n buildings on a city block, indexed from west to east. Building i has good view of Lake Michigan if and only if every building to the east of i is shorter than building i. Here is an algorithm that computes which buildings have a good view of Lake Michigan. What is the running time of this algorithm?

Algorithm 1: GOODVIEW($A[1..n]$)

```
initialize a stack S
for i = 1 to n
    while (S not empty and A[i] > A[TOP(S)])
        POP(S)
    PUSH(S, i)
return S
```

Problem 3: A vertex cover of a graph $G = (V, E)$ is a subset of vertices $S \subseteq V$ that includes at least one endpoint of every edge in E. Give a linear-time algorithm for the following task. Given an undirected tree $T = (V, E)$, return the size of the smallest vertex cover of T.

Problem 4: Suppose you are given an array $A[1..n]$ of real numbers.

(a) Describe and analyze an algorithm that finds the largest sum of elements in a contiguous subarray $A[i..j]$.
(b) Describe and analyze an algorithm that finds the largest product of elements in a contiguous subarray $A[i..j]$.

For example, given the array $A = [-6, 12, -7, 0, 14, -7, 5]$ as input, your first algorithm should return the integer 19 (sum of elements in $A[2..5]$), and your second algorithm should return the integer 504 (product of elements in $A[1..3]$).

Problem 5: Given an unlimited supply of coins of denominations $x_1, x_2, \ldots, x_n$, we wish to make change for a value v, that is, we wish to find a set of coins whose total value is v. This might not be possible: for instance, if the denominations are 5 and 10 then we can make change for 15 but not for 12. Given the denominations $x_1, x_2, \cdots, x_n$ and v, find an $O(nv)$ dynamic-programming algorithm to decide if it is possible to make change for v using coins of denominations $x_1, \ldots, x_n$.