CS 430 – FALL 2017
INTRODUCTION TO ALGORITHMS
HOMEWORK #6
DUE Fri Nov 17, 11:25am

1. (4 points) Exercise 17.1-2 - Show that if a DECREMENT operation were included in the k-bit counter example, n operations could cost as much as $\Theta(nk)$ time.

2. (4 points) Exercise 17.2-3 - Suppose we wish not only to increment a counter but also to reset it to zero (i.e., make all bits in it 0). Counting the time to examine or modify a bit as THETA(1), show how to implement a counter as an array of bits so that any sequence of n INCREMENT and RESET operations takes time O(n) on an initially zero counter. (Hint: Keep a pointer to the high-order 1.)

3. (4 points) Exercise 19.2-1 Show the Fibonacci heap that results from calling FIB-HEAP-EXTRACT-MIN on the Fibonacci heap shown in Figure 19.4(m).

4. (5 points) Consider the recursively defined sequence of Fibonacci heap operations defined by the following function:

```
function Tall-Heap(T, h, b)
1: if h = 1 then
2:         Make-Fib-Heap(T)
3:         Fib-Heap-Insert(T, b)
4: else if h = 2 then
5:         Make-Fib-Heap(T)
6:         Fib-Heap-Insert(T, b − 1)
7:         Fib-Heap-Insert(T, b)
8:         Fib-Heap-Insert(T, b + 1)
9:         Extract-Min(T)
10: else
11:        Tall-Heap(T, h − 1, b + 1)
12:        Fib-Heap-Insert(T, b − 0.5)
13:        Fib-Heap-Insert(T, b)
14:        Fib-Heap-Insert(T, b + 0.5)
15:        Extract-Min(T)
16:        Fib-Heap-Delete(T, b + 0.5)
17: end if
```

a) Prove by induction on h that the Fibonacci heap that results from the call Tall-Heap(F, h, b) is a chain of h − b + 1 nodes. Note there are two base cases in the algorithm.

b) What is the total time required by the call Tall-Heap(F, h, b)? Justify your answer.

5. (4 points) 21.1-3 During the execution of CONNECTED-COMPONENTS on an undirected graph G = (V, E) with k connected components, how many times is FIND-SET called? How many times is UNION called? Express your answers in terms of |V|, |E|, and k.

6. (4 points) We have students 1, 2, …, n who need to be assigned to dormitories at a university that has an arbitrarily large number of dorms. There are "m" same dormitory requests (s1,t1), (s2,t2), …,(sm,tm) meaning students si and ti must be assigned to the same dorm. There are also "k" different dormitory requests (u1,v1), (u2,v2),…,(uk,vk) meaning students ui and vi must be assigned to different dorms. Give an algorithm using the union-find data structure to determine whether it is possible to assign students to dorms so that all constraints are satisfied.