Illinois Institute of Technology
Department of Computer Science

Third Examination

CS 430 Introduction to Algorithms
Spring, 2016

Wednesday, April 27, 2016
10am–11:15am & 11:25am–12:40pm, 111 Life Sciences

Print your name and student ID, *neatly* in the space provided below; print your name at the upper right corner of *every* page. Please print legibly.

Name:
Student ID:

This is an *open book* exam. You are permitted to use the textbook, any class handouts, anything posted on the web page, any of your own assignments, and anything in *your own handwriting*. Foreign students may use a dictionary. *Nothing else is permitted*: No calculators, laptops, cell phones, Ipods, Ipads, etc.; no photocopies of other books, papers, web sites, etc.

Do all five problems in this booklet. *All problems are equally weighted, so do not spend too much time on any one question.*

Show your work! You will not get partial credit if the grader cannot figure out how you arrived at your answer.

Question	Points	Score	Grader
1	20		
2	20		
3	20		
4	20		
5	20		
Total	100		

1. **Fibonacci Heaps**

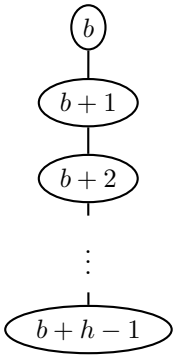
   (*Hint*: This problem is easier than it looks.)

   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

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

   (c) Exercise 19.4-1 claims that an n-node Fibonacci heap can attain height $\Theta(n)$. Prove the claim in that exercise.

1. **Fibonacci Heaps, continued.**

2. **Depth First Search**

Below is the text's depth first search function for directed graphs. Fill in the blanks at lines 5, 9, 11, and 13 with the type of edge discovered and the blank conditions at lines 8 and 10 so that the edges are classified as tree, back, cross, or forward edges. This was suggested as a good exam problem in the lecture of March 30.

function DFS-visit(u)

1: $color[u] \leftarrow$ GRAY
2: $d[u] \leftarrow time \leftarrow time + 1$
3: **for all** $v \in Adj[u]$ **do**
4: **if** $color[v] =$ WHITE **then**
5: _____ edge
6: $\pi[v] \leftarrow u$
7: DFS-visit(v)
8: **else if** _____ **then**
9: _____ edge
10: **else if** _____ **then**
11: _____ edge
12: **else**
13: _____ edge
14: **end if**
15: **end for**
16: $color[u] \leftarrow$ BLACK
17: $f[u] \leftarrow time \leftarrow time + 1$

3. **Shortest Paths**

Suppose we assign weights to *vertices* in a graph rather than to *edges*. In this case, the overall length of a path between two vertices is the sum of the weights of all vertices on the path, including the start and end vertices. Given an undirected graph $G = (V, E)$ whose vertices are weighted by the function $w_V(\cdot)$, we can instead get weighted edges by assigning the weight of an edge (x, y) to be

$$w_E(x, y) = \frac{w_V(x) + w_V(y)}{2}. \tag{1}$$

We want to prove that the shortest path in the vertex-weighted graph is the same as the shortest path in the edge-weighted graph specified by equation (1).

(a) Prove that equal length paths between vertices u and v in the vertex-weighted graph are also equal in the edge-weighted graph.

(b) Prove that if path P_1 is shorter than path P_2 between the vertices u and v in the vertex-weighted graph, then it is also shorter in the edge-weighted graph.

4. **NP-Completeness**

We are given n jobs, $J_1, J_2, \ldots, J_n$, some pairs of which are mutally incompatible and must be scheduled on separate days. Given an $n \times n$ Boolean matrix C in which

$$C_{ij} = \begin{cases} 1 & \text{if jobs } J_i \text{ and } J_j \text{ are incompatible,} \\ 0 & \text{otherwise,} \end{cases}$$

the EXAM3-SCHEDULING problem is to find the fewest number of days needed to schedule the n jobs. Rephrase this problem as a decision problem and show that it is NP-complete.

(*Hint*: Think about graph coloring.)

5. **Spanning Tree Approximation**

Consider the following stupid algorithm for finding a spanning tree: examine the edges of the graph in any order, adding an edge to the partial spanning tree if and only if it does not form a cycle with the edges already added.

(a) Show that this algorithm does not approximate the minimum spanning tree within any finite bound. Specifically, give a graph and an edge ordering that results in a spanning tree whose weight is at least k times the weight of the minimum spanning tree, for any $k > 0$.

(*Hint*: Consider a graph of three vertices and three appropriately weighted edges.)

(b) Suppose the edge weights are real numbers in the range 1 to 2. What kind of approximation bound can you give for the stupid algorithm now?