Illinois Institute of Technology
Department of Computer Science

Second Examination

CS 430 Introduction to Algorithms
Spring, 2009

11:25am–12:40pm, Wednesday, April 8, 2009
104 Stuart Building

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, etc.!

Do all four 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	25		
2	25		
3	25		
4	25		
Total	100		

1. **Dynamic Programming**

   You are given an array of n integers. Consider the problem of finding the maximum sum in any contiguous subvector of the input. For example, in the array:

   $$\{-6, 12, -7, 0, 14, -7, -3\}$$

   the maximum sum of 19 is achieved by summing the contiguous elements $\{12, -7, 0, 14\}$.

   Give and explain a $\Theta(n)$-time dynamic programming algorithm for solving this problem. For some partial credit, you may instead give a $\Theta(n^2)$-time algorithm. Explain your algorithm in words rather than just giving pseudocode.

2. **Double-deletion Heaps**

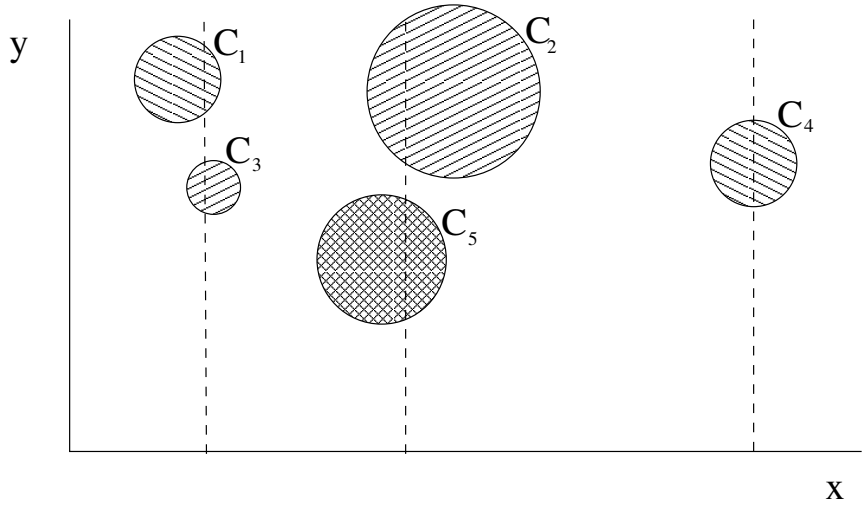
A student suggested the following "extra lazy" variation of the Fibonacci heap called a *Double-deletion heap*: rather than moving a node up to the root list after it loses its second child, a node is only moved to the root list after it loses its third child. The student discovers that most of the analysis follows as in CLRS, but he cannot bound the maximum degree and comes to you for help.

(a) Give a recurrence for the minimum number of nodes in a subtree whose root has degree k.

(b) What is a necessary and sufficient condition which would cause Double-deletion heaps to have the same asymptotic amortized running times as standard Fibonacci heaps?

3. **Greedy Heuristic**

You are walking along a path above which n balloons $C_1, C_2, \ldots, C_n$ are floating. You are required to pop them all using a pistol and only shooting directly up. If a bullet hits a balloon anywhere, it pops. You are told the position of balloon C_i as its distance x_i along the path, its altitude y_i, and its radius r_i. Assuming the balloons do not move while you are shooting and that your pistol has enough range to pop the highest balloon, give an algorithm to pop all the balloons using the least number of pistol shots. Prove that this algorithm uses the smallest possible number of shots and analyze its time complexity in terms of n.



This arrangement of 5 balloons requires 3 shots.

3. **Greedy Heuristic, continued.**

4. **Combined-Min**

Suppose you have a data structure that supports the following operations:

- `make`(x): make a new structure D containing only the element x.
- `combine-equal`(D_1, D_2): make a new structure D consisting of all the elements of D_1 and D_2; D_1 and D_2 must have an equal number of elements.
- `delete-min`(D): delete the minimum element from D and return its value.

You may assume that `make` requires $O(1)$ time.

(a) Show how to sort any n items with this data structure.

(b) Prove that it is impossible for `combine-equal` to run in time $o(n)$ if `delete-min` runs in time $o(\log n)$. *Hint:* Recall that comparison-based sorting requires time $\Omega(n \log n)$.