CS 373: Combinatorial Algorithms, Summer IMCS 2000

`http://www-courses.cs.uiuc.edu/~cs373`

Homework 2 (due Thursday June 15, 2000)

Submit solutions by 1700 GMT (12 noon Central Standard Time) by **attaching** a **postscript** file to an email sent to `maharri@cs.uiuc.edu` with the subject `cs373hw submit`. You will then get an automatic email acknowledgment. Late homeworks (those received after the deadline, no matter when they are sent) are not accepted.

Note: When a question asks you to "give/describe/present an algorithm", you need to do four things to receive full credit:

1. Design the most efficient algorithm possible within the specifications given. Significant partial credit will be given for less efficient algorithms, as long as they are still correct, well-presented, and correctly analyzed.
2. Describe your algorithm succinctly, using structured English/pseudocode. We don't want full-fledged compilable source code, but plain English exposition is usually not enough. Follow the examples given in the textbooks, lectures, homeworks, and handouts.
3. Justify the correctness of your algorithm, including termination if that is not obvious.
4. Analyze the time and space complexity of your algorithm.

---

1. (10 pts) ScapegoatSelect(A, k)
   Say that a binary search tree is *augmented* if every node v also stores $|v|$, the number of nodes in its subtree.

   Describe an algorithm ScapegoatSelect(k) that selects the kth smallest item in an augmented scapegoat tree in $O(\log n)$ *worst-case* time. (The scapegoat trees presented in class were already augmented.) Hint: Make sure that $|v|$ can be maintained in time that doesn't affect the time of the other operations.

2. (10 pts) Fib-Heap-Second-Smallest

   Give an $O(1)$ time implementation of the operation Fib-Heap-Second-Smallest, which returns the second smallest item in the heap, by two different methods (and two different running times):

   (a) $O(\log n)$ time using only previously defined heap operations as black boxes.

   (b) $O(1)$ time by modifying the data structure and some operations. For this one, the changes you make to Fibonacci heap data structure to support your implementation should not affect the amortized running time of any other Fibonacci heap operations.