

## Homework 2 – Due Friday, September 17, 2008

Please refer to the general information handout for the full homework policy and options.

### Reminders

- Your solutions are due before the lecture. Late homework will not be accepted.
- Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. *Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.*
- To facilitate grading, please write down your solution to each problem on a separate sheet of paper. Make sure to include all identifying information and your collaborators on each sheet. Your solutions to different problems will be graded separately, possibly by different people, and returned to you independently of each other.
- For problems that require you to provide an algorithm, you must give a precise description of the algorithm, **together with a proof of correctness** and an analysis of its running time. You may use algorithms from class as subroutines. You may also use any facts that we proved in class or from the book.

**Reading** Review chapter 4.1-4.5 in Kleinberg Tardos.

**Exercises** These should not be handed in, but the material they cover may appear on exams:

1. Exercises in Chapter 4.
2. (**Caching**) In our discussion of optimal caching, we used the fact that any eviction schedule may be changed to a *reduced* eviction schedule without affecting its cost. This is stated informally in the book (Lemma 4.11 on page 134).

Let us make this more precise. If an eviction schedule brings an item  $d$  into the cache in a step where the  $d$  was not requested, we say the eviction was *unreduced*. Prove the following by induction on the number of unreduced items: Let  $S$  be any eviction schedule which has no unreduced evictions up to some step  $j$ . There exists a reduced schedule  $\bar{S}$  that brings in as most as many items as  $S$  and that performs the same actions as  $S$  in the first  $j$  steps.

3. Point out the error in the following proof by induction.

**Claim 1** *In any set of  $h$  horses, all horses are the same color.*

**Proof:** We proceed by induction on the number  $h$  of horses. (Base case) If  $h = 1$ , then there is only one horse in the set, and so all the horses in the set are clearly the same color.

(Induction Step) For  $k \geq 1$ , we assume that the claim holds for  $h = k$  and prove that it is true for  $h = k + 1$ . Take any set  $H$  of  $k + 1$  horses. We show that all horses in this set are of the same color. Remove one horse from this set to obtain the set  $H_1$  with just  $k$  horses. By the induction hypothesis, all the horses in  $H_1$  are the same color. Now replace the removed horse and remove a different one to obtain a the set  $H_2$ . By the same argument, all the horses in  $H_2$  are the same color. Therefore all the horses in  $H$  must be the same color, and the proof is complete.

4. (**Analysis of  $d$ -ary heaps**) A  $d$ -ary heap is like a binary heap, described in Chapter 2.5 of Kleinberg Tardos, with the exception that non-leaf nodes have  $d$  children instead of 2.
  - (a) How would you represent a  $d$ -ary heap in an array?
  - (b) Implement  $\text{PARENT}(i)$  that, given the index  $i$  of a node, returns the index of its parent and  $\text{CHILD}(i, k)$  that, given the index  $i$  of a node, returns the index of its  $k$ th child.
  - (c) What are the minimum and the maximum number of elements in a  $d$ -ary heap of height  $h$ ?
  - (d) Design an efficient implementation of  $\text{HEAPIFY-UP}$  in a  $d$ -ary min-heap, analogous to the procedure on page 61 of KT. Analyze the running time of your algorithm in terms of  $d$  and  $n$ .
  - (e) Design an efficient implementation of  $\text{HEAPIFY-DOWN}$  in a  $d$ -ary min-heap, analogous to the procedure on page 63 of KT. Analyze the running time of your algorithm in terms of  $d$  and  $n$ .
  - (f) Suppose we implement a priority queue using a  $d$ -ary heap. Give the running times of all operations, described on pages 64–65 of KT, in terms of  $d$  and  $n$ .

### Problems to be handed in

**Page limits:** The answer to each problem should fit in **2 pages (or one double-sided sheet)** of paper. Longer answers will be penalized.

1. (**Asymptotic Notation**) Let  $f(n)$  and  $g(n)$  be asymptotically positive functions. Prove or disprove (by giving a counterexample) each of the following conjectures.
  - (a) Let  $f(n)$  a positive function. Prove or disprove:  $f(n) = \Theta(f(n/2))$ .
  - (b) Let  $f_1, f_2, f, \dots$  be an infinite sequence of positive functions such that for all  $k \geq 0$ , we have  $f_k(n) = O(n)$ . Prove or disprove:  $\sum_{i=1}^n f_i(n) = O(n^2)$ .
  - (c) For what positive number  $a$  does the following hold (prove the correctness of your answer): The probability that a fair coin flipped  $n$  times (independently) comes up heads exactly  $n/2$  times is  $\Theta(n^{-a})$ .  
(Hint: use Stirling's approximation to approximate the number of strings in  $\{0, 1\}^n$  that have exactly  $n/2$  ones. Stirling's approximation is
 
$$n! = (\sqrt{2\pi n}) \left(\frac{n}{e}\right)^n (1 + w_n),$$
 where  $w_n \geq 0$  for all  $n$ , and  $w_n = O(\frac{1}{n})$ .)
  - (d) Prove that  $\sum_{i=1}^n \log^2(i) = \Theta(n \log^2(n))$ . (Hint: this sum is bounded below by  $\frac{n}{2} \log^2(\frac{n}{2})$ .)
2. (**Greedy algorithms: Matching points to intervals**) Chapter 4, Problem 16.
3. (**Greedy Algorithms: El Goog**) Chapter 4, Problem 7.
4. (**Greedy Algorithms: Structural Argument**) Chapter 4, Problem 14.