Illinois Institute of Technology
Department of Computer Science

Solutions to Second Examination

CS 430 Introduction to Algorithms
Fall, 2010

11:25am–12:40pm, Wednesday, November 3, 2010
121 Life Sciences Building

Exam Statistics

41 students took the exam. The range of scores was 20–92, with a mean of 49.2, a median of 49, and a standard deviation of 16.53. Very roughly speaking, if I had to assign final grades on the basis of this exam only, above 70 would be an A (6), 50–69 a B (14), 40–49 a C (8), 30–39 a D (8), below 30 an E (5).

Problem Solutions

1. (a) Yes, nodes can be augmented with $width(x)$ because $width(x)$ satisfies the hypothesis of Theorem 14.1 (page 346 in CLRS), namely that $width(x)$ depends only on $width(x.left)$ and $width(x.right)$,

$$width(x) = \begin{cases} 0 & \text{if } x \text{ is a leaf,} \\ \max(width(x.left), width(x.right)) & \text{if } width(x.left) \neq width(x.right), \\ width(x.left) + 1 & \text{otherwise.} \end{cases}$$

When arithmetic expressions are written as binary trees, the "width" gives the number of registers needed to evaluate the expression (why?). The "width" values are sometimes called *Strahler numbers*, useful when trees are are used to model river systems.

(b) No, nodes cannot be augmented with $depth(x)$. $depth(x)$ does not satisfy the hypothesis of Theorem 14.1, but that does not mean it could not augment red-black tree nodes: the theorem's hypothesis is sufficient, but not necessary. However, under a left rotation—see Figure 13.2 on page 313 of CLRS—all depth values in subtree α need to be decremented by 1 and all depth values in subtree γ need to be incremented by 1. That is, a tree of n nodes may require $O(n)$ changes, making $O(\log n)$ insertion/deletion impossible.

2. (a) We can only reach step k from steps $k-1$ or $k-2$, so the Principle of Optimality tells us

$$C(k) = \begin{cases} c_k & \text{if } k < 2, \\ c_k + \min(C(k-1), C(k-2)) & \text{if } k \geq 2. \end{cases}$$

(b) To understand the time required when the recurrence in (a) is directly implemented without memoization, it is sufficient to count the number of recursive calls. Let t_n be the number of recursive calls while computing $C(n)$. Then $t_0 = t_1 = 0$ and $t_n = 1 + t_{n-1} + t_{n-2}$. Solving (using annihilators or some other method) we find that $t_n = F_n - 1$, where F_n is the nth Fibonacci number which grows like ϕ^n, where $\phi = (1+\sqrt{5})/2 \approx 1.61$ is the golden ratio. The unmemoized implementation is thus exponential in time.

(c) We memoize the computation with an array C

```
1: C[0] = 0
2: C[1] = c_1
3: for k = 2 to n do
4:    if C[k − 1] < C[k − 2] then
5:       C[k] = c_k + C[k − 1]
6:    else
7:       C[k] = c_k + C[k − 2]
8:    end if
9: end for
```

The cost of this computation is clearly $O(n)$.

(d) We keep track of which one of $C[k-1]$ or $C[k-2]$ is smaller at each iteration of the loop:

```
1: C[0] = 0
2: C[1] = c_1
3: R[0] = 0
4: R[1] = 0
5: for k = 2 to n do
6:    if C[k − 1] < C[k − 2] then
7:       C[k] = c_k + C[k − 1]
8:       R[k] = k − 1
9:    else
10:      C[k] = c_k + C[k − 2]
11:      R[k] = k − 2
12:   end if
13: end for
```

The cheapest way to get to the roof is on rungs $\ldots$, $R[R[R[n]]]$, $R[R[n]]$, $R[n]$, n.

3. (a) A simple greedy algorithm is to fill the tank with gasoline whenever the distance to the next gas station (displayed on the GPS) is greater than the number of miles the car drive on its remaining fuel (displayed on the dashboard).

(b) To prove that the number of stops is minimal, suppose it is not, and ask when the *first* mistake occurs. Thus, let the greedy sequence of stops be at gasoline stations g_1, g_2, $\ldots$, g_n and suppose the first mistake is when we stop at gasoline station g_k; this means that stops at gasoline stations g_1, g_2, $\ldots$, g_{k-1} can still lead to an optimal solution, say g_1, g_2, $\ldots$, g_{k-1}, o_k, o_{k+1}, $\ldots$. That first mistake could not be stopping too early—we

stop only when we would run out of gasoline before the next available station. So the first mistake must have been *not stopping* at gasoline station o_k after g_{k-1} but before g_k. But then the sequence of stops g_1, g_2, ..., g_{k-1}, g_k, o_{k+1}, ... would also work and be optimal (it has the same number of stops and g_k is beyond o_k, so we can reach o_{k+1}). That means that the first "mistake" was not a mistake at all, because there is an optimal sequence of stops beginning g_1, g_2, ..., g_{k-1}, g_k. Thus the greedy strategy never makes a mistake.

4. (a) The worst-case cost is $O(m)$; specifically, $m+1$ bits need to be examined/modified.

   (b) We use the given potential function to compute the amortized costs of the INCREMENT and RESET operations.

   INCREMENT: Suppose that the increment resets t 1-bits and sets one 0-bit. The actual cost of the INCREMENT is thus $t+1$. The amortized cost is $t+1+\Phi_{\text{after}}-\Phi_{\text{before}}$ which is

   $$t+1+(m_{\text{after}}+\text{number of 1-bits in } A_{\text{after}})-(m_{\text{before}}+\text{number of 1-bits in } A_{\text{before}}).$$

   But $m_{\text{after}}-m_{\text{before}} \le 1$ because the most significant 1-bit can move left by at most one position, and (as in CLRS on page 461) the change in the number of 1-bits is $(1-t)$; the amortized cost is thus $O(1)$.

   RESET: The actual cost is $m+1$, the number of 1-bits does not increase, and the change in m is $m_{\text{after}}-m_{\text{before}} = 0-m = -m$, so the change in potential is at most $-m$, giving an amortized cost of $O(1)$.

5. Yes, $D(n)$ is still $O(\log n)$.

   We need to prove an analogue of Lemma 19.4 on page 525 of CLRS in order to get an analogue of Corollary 19.5 on page 526 for the modified Fibonacci heaps; to do that, we need an analogue of Lemma 19.1 on page 523.

   New Lemma 19.1: Let x be any node in a (modified) Fibonacci heap with $\mathit{degree}(x) = k$, and children $y_1, y_2, \ldots, y_k$ (in the order in which they were linked to x, from earliest to latest). Then $\mathit{degree}(y_i) \ge i-1$, for $1 \le i \le k$.

   The proof of this lemma is *exactly* the same as the first two sentences of the proof of the original version.

   New Lemma 19.3: Let x be any node in a (modified) Fibonacci heap, and let $\mathit{degree}(x) = k$. Then $\text{SIZE}(x) \ge 2^k$.

   The proof of this lemma, which includes the observation that $s_0 = 1$ and $s_1 = 2$, begins the same as the proof of the original version, up to the phrase "To bound s_k, we count...". Then we continue: To bound s_k, we count one for z itself and add the degrees of its children:

   $$\text{SIZE}(x) \ge s_k \ge 1 + \sum_{i=1}^{k} s_{\mathit{degree}(y_i)} \ge 1 + \sum_{i=1}^{k} s_{i-1} \ge 1 + \sum_{i=0}^{k-1} s_i.$$

It now follows by induction that $s_k \geq 2^k$ and we have the needed corollary,

New Corollary 19.5: The maximum degree $D(n)$ of an node in an n-node (modified Fibonacci heap is $\lfloor \lg n \rfloor$.

This corollary is (basically) the answer to Problem 19-2(d) on page 529—do you see why?