Martin Ziegler
Sewon Park, GyuHyeon Choi, Junsung Lim, Won-young Lee

Issued on Mar.28, 2016
Solutions due: Apr.7, 2016

CS500

Spring 2016, Assignment #2

PROBLEM 7 (2+2+2+2+2P)**:**
We have seen that a stable matching need not be unique.
(a) Specify, (b) describe, (c) analyze, and (d) justify the correctness of an (e) quadratic-time algorithm that verifies whether a given matching between n 'men' and n 'women' is stable.

PROBLEM 8 (2+2+2+2+2P)**:**

a) Prove that a binomial tree B_k has precisely $\binom{k}{d}$ nodes at depth d.

b) Recalling the relationship between merging two binomial heaps and adding two binary numbers, describe an $\mathcal{O}(\log n)$ algorithm for directly inserting a node.

c) Find inputs that cause `ExtractMin` and `DecreaseKey` to run in time $\Omega(\log n)$.

d) Argue that the running time of a sequence of n calls to `InsertKey` is $\mathcal{O}(n)$, not $\Omega(n \log n)$.

e) Construct a sequence of n calls that produce a degenerate Fibonacci Heap of height $\Omega(n)$.

PROBLEM 9 (1+3+3+3P)**:**
Recall that counting from 1 to n in binary takes $\Theta(n)$ steps; i.e., the increment operation has constant amortized cost as opposed to $\Theta(\log n)$ in the worst-case.

a) Analyze the amortized cost of any mixed sequence of n binary increment and decrement operations, where decrementing 0 results in 0.

b) The *signed* binary expansion represents $N \in \mathbb{N}$ as $\sum_{j=0}^{J-1} b_j 2^j$ for $b_j \in \{0,1,\bar{1}\}$, where $\bar{1} = -1$; e.g. $5 = 0101 = 011\bar{1} = 10\bar{1}\bar{1} = 1\bar{1}01$.
Describe an algorithm for both incrementing and decrementing; generalize the potential function Φ from the lecture to show them to have constant amortized cost.

c) *Redundant* arithmetic represents $N \in \mathbb{N}$ as $\sum_{j=0}^{J-1} b_j 2^j$ for $b_j \in \{0,1,2\}$; e.g.

$$10 = 1010 = 0202 = 0210 = 1002$$

Consider the following algorithm for incrementing a number in this representation:

> Replace the rightmost occurrence of $x2$ with $(x+1)0$;
> If the rightmost digit is 0, change it to 1; otherwise to 2.

Use it to count from 0 to 32, writing down all intermediate results.
How can this be turned into an algorithm with constant worst-case complexity?
What remains to prove in order to assert its correctness?
Implement and run it to try to find a counterexample.

d) Combine (b) and (c) to devise an algorithm for both incrementing and decrementing at constant worst-case cost. (You do not need to prove its correctness — as long as it is correct.)