CS 32 WFW

Long Exam 1

July 19, 2013

General Instructions

- Answer the items completely. Show your solutions/justifications when asked.
- Write as legibly as possible. Illegible or unreadable answers and solutions may not merit any points.
- Refrain from making unnecessary motions and sounds during the exam. Any suspicious behavior will be dealt with accordingly.
- Direct all questions to the proctor.
- If you need to go to the CR, hand your questionnaire, answer sheet, and scratch paper to the proctor before heading out. Only one person at any given time is allowed to go out.
- Once you're done with the exam (one way or the other), place your scratch papers and the questionnaire inside your blue book.

Questions

1. Arrange the following functions in increasing order of complexity (assume we have sufficiently large value for n): $\log^2 n,\ 3n^{\frac{3}{2}},\ 10n,\ \log(\log n),\ n\log n,\ n^2\log n,\ 2^{2^n},\ n^3,\ 2^{\log n}$
   ANSWER: $\log(\log n),\ \log^2 n,\ 2^{\log n},\ 10n,\ n\log n,\ n^2\log n,\ 3n^{\frac{3}{2}},\ n^3,\ 2^{2^n}$

2. Given the following EASY code below:

```
for i ←1 to n do
  for j ← (i + 1) to n do
    for k ← 1 to j do
       output i, j, k
    endfor
  endfor
endfor
```

   (a) How many times will "output i, j, k" be executed? Show your solutions.
   ANSWER: $\frac{n^3-n}{3}$ times

   (b) Let the answer to (a) also be the exact running time of the code above. What is the corresponding time complexity? Make the bound as tight as possible.
   ANSWER: $O(n^3)$

3. Five stacks coexist in an array of size 500. The state variables of the stacks are as follows:

   - B(1:6) = (0, 100, 210, 290, 380, 500)
   - T(1:5) = (80, 200, 290, 379, 480)
   - OLDT(1:5) = (50, 200, 260, 350, 470)

   An item is pushed onto the 3rd stack, causing an overflow. Perform the Garwick-Knuth algorithm to reallocate memory. Show computations for each important variable.

ANSWER:
$freecells = 500 - (80 + 100 + 80 + 1 + 89 + 100) = 50$
$increment = 30 + 0 + 30 + 1 + 29 + 10 = 100$
$a = (0.1 \times 50)/5 = 1$
$b = (0.9 \times 50)/100 = 0.45$

$NEWB(1) = 0; s_2 = 0$

$t_2 = 0 + 1 + (30 \times 0.45) = 14.5$
$NEWB(2) = 0 + 80 + \lfloor 14.5 \rfloor - \lfloor 0 \rfloor = 94; s_3 = 14.5$

$t_3 = 14.5 + 1 + (0 \times 0.45) = 15.5$
$NEWB(3) = 94 + 100 + \lfloor 15.5 \rfloor - \lfloor 14.5 \rfloor = 195; s_4 = 15.5$

$t_4 = 15.5 + 1 + (31 \times 0.45) = 30.45$
$NEWB(4) = 195 + 81 + \lfloor 30.45 \rfloor - \lfloor 15.5 \rfloor = 291; s_5 = 30.45$

$t_5 = 30.45 + 1 + (29 \times 0.45) = 44.5$
$NEWB(5) = 291 + 89 + \lfloor 44.5 \rfloor - \lfloor 30.45 \rfloor = 394; s_6 = 44.5$

$t_6 = 44.5 + 1 + (10 \times 0.45) = 50$
$NEWB(6) = 394 + 100 + \lfloor 50 \rfloor - \lfloor 44.5 \rfloor = 500$

4. Design a singly-linked queue *with only one external pointer* such that the enqueue and dequeue can still be done in O(1) time. Demonstrate step-by-step how the following is done:

   - Enqueue to an empty queue
   - Enqueue to a non-empty queue
   - Dequeue from a non-empty queue

   ANSWER: This is tantamount to implementing a linked circular queue: let the lone external pointer (which we will refer to as *rear*) point to the rear of the queue, and let the link of the node pointed to by *rear* point to the first node of the queue, i.e. the front. The operations that were required to be demonstrated are done as follows (in EASY code):

   - To perform enqueue to an empty queue, let α be the new node to be enqueued:
     `rear` $\leftarrow \alpha$
     `LINK(rear)` $\leftarrow$ `rear`

   - To perform enqueue to a non-empty queue, let α be the new node to be enqueued:
     `LINK(`α`)` $\leftarrow$ `LINK(rear)`
     `LINK(rear)` $\leftarrow \alpha$
     `rear` $\leftarrow \alpha$

   - To perform dequeue from a non-empty queue, let β be a pointer we could use to point to the item to be dequeued:
     $\beta \leftarrow$ `LINK(rear)`
     `LINK(rear)` $\leftarrow$ `LINK(`β`)`
     `RETNODE(`β`)`