CSC172 Midterm with Answers
17 October 2014

Please write your name on the bluebook. Two sides 8 1/2 x 11 handwritten notes permitted. There are 75 points, or one per minute. Stay cool and please write answers neatly.

1 Induction for Arithmetic Series (15 min)

A. An n-term *arithmetic series* with first term a and increment b has the form $a, (a+b), (a+2b), ..., (a+(n-1)b)$. Prove by mathematical induction on n that the sum of such a series is

$$\sum_{i=0}^{n-1} a + bi = n(2a + (n-1)b)/2.$$

B. Show this formula is compatible with the one for the sum the first n positive integers, namely $S = n(n+1)/2$.

Ans.
A. To prove:

$$\sum_{i=0}^{n-1} a + bi = n(2a + (n-1)b)/2.$$

Base Case: true for $n = 1$ (one term sum):
$n(2a + (n-1)b)/2 = 1(2a + 0b)/2 = a$.

Using inductive hypothesis and adding the nth term in the sum:

$$\sum_{i=0}^{n} a + bi = n(2a + (n-1)b)/2 + (a + nb)$$

After some simple algebra

$$\sum_{i=0}^{n} a + bi = (n+1)(2a + nb)/2$$

Which is what we need.

B. Sum of first n numbers is arithmetic series with $a = 0, b = 1$. Substituting those constants into the arith series sum formula gives the desired expression.

2 Combinatorics: 15 min

A. A 52-card deck has cards of four different *suits* (♣, ♢, ♡, ♠). Each suit has 13 cards of different *ranks*: 2,3,...,10,J,Q,K,A.

A *one-pair hand* is an unordered set of five cards, only two of which have the same rank. (so 4J424 is not a one-pair hand, nor is 56567, etc.).

How many different one-pair hands are there in a deck? Please produce a formula, not a number, and explain how you derived the formula.

(Hint: the most elegant (and only needed) operators are $\binom{\ }{\ }$, exponentiation, and multiplication. BUT the solution you believe in that comes easiest to you is best today.)

Ans. 1 Select rank of pair; There are four cards of that rank, one for each suit. Select suits for the 2 of these 4 cards that will be the pair in the hand; select 3 ranks from the 12 unused (non-pair) ranks and assign one each to the 3 non-pair cards in the hand. Select one suit (of four) for each of the 3 non-pair cards. That is,

$$13 * \binom{4}{2} * \binom{12}{3} * 4^3$$

B. *FOCS*, our Bible of Combinatorics, shows (p. 182) that there are $\binom{n+b-1}{n} = \binom{n+b-1}{b-1}$ ways to distribute n identical objects to b bins. E.g. there are $\binom{8}{3}$ ways to distribute 5 apples (objects) to 4 kids (bins).

Please explain why the general formula is correct. (Hint: string of A's and *'s).

Ans.

B. We model the situation by a string with A's for apples and *'s dividing the bins (need 3 for 4 kids (i.e. bins)) so a distribution could look like A*AA**AA where 3rd bin got nothing. The bincoef counts the ways to pick (assign) the location of the b-1 *'s from the m+n-1 locations in the string. (could use n instead of b-1 equivalently, picking where to put the A's).

3 Hashing: 10 min

In a closed hash table with 12 initially-empty positions, use double hashing to resolve collisions in hashing integer keys. The first hash function is $h_1(key) = \Sigma k_i (\mathrm{mod} 12)$, where the k_i are the digits of key. E.g. $h_1(576) = (5+7+6)(\mathrm{mod} 12) = 6$.

The rehash function is $h_2(key) = 7 - (key(\mathrm{mod} 7))$.

Insert the following keys in order: 9999, 13579, 85, 106.

Describe the probes for each key, including its final destination. after the four insert operations. Comment on any problems and give any suggestions that occur to you.

Ans. 9999 goes to 0
13579 to 1
85 goes to 1, collides and rehashes to 1+ (7-(85 mod 7)), or 1+6=7.

106 goes to 7, collides and rehashes to 7+ (7-(106 mod 7)), or 7+(7-1) = 13, which we must mod by 12 to fit it into the table at 1. This collides, and we probe at 1+6 =7, which collides. If we are paying attention, we give up and declare the table full.

Suggestion: use prime table sizes (Weiss pp. 183-185).

4 Big Oh-oh...(15 min)

The algorithm `OhBoy` below is a function of N.
`for i= (start to finish by incr)` loops with `i`'s value incrementing by `incr` from the `start` to `finish` value (inclusive). `random()` returns a (uniformly distributed) random number in the interval [0,1]. `log()` returns the log of its argument, and $\wedge$ is the exponentiation operator.

A. (5 min) What is the $O()$ (upper-bound) complexity of OhBoy's running time?

B. (5 min) What is the $\Omega()$ (lower-bound) complexity of OhBoy's running time?

C. (5 min) What can you say about the average running time of OhBoy? How would you estimate it?

```
for i = (1 to N by 1) {
  for j = (1 to 2N by 1) {
    s = (i+j)*(i+j); }}

for i = (N to 0 by -1)
  if (random() < 0.5)
    s = N*log(N);
  else
    for j = (1 to N by 1) {
      for k = (N/2 to 10*N by 1) {
           s = N^4; }}
```

Ans:
A. All loop bodies are $\Theta(1)$. The first loop is $O(N^2)$ (and $\Theta(N^2)$.) The second is $O(N^3)$. So whole thing is $O(N^3)$.

B. The first loop is $\Omega(N^2)$. The second is $\Omega(N)$. First loop is always run, so whole thing is $\Omega(N^2)$.

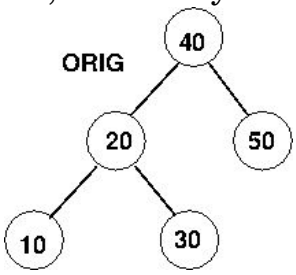
C. OhBoy has a $\Theta(N)$ component, also $\Theta(N^2)$ and $\Theta(N^3)$, all with different constants. I can't think of much to say just by eyeballing the code. However, the Θ's (or code itself) show(s) us that T(N) must be a cubic function of N (depending on N, N^2, N^3 and four constants.) So I'd just code it up and run it maybe 1000 times on random N's in a range (maybe [1,100]?), and then fit a cubic to the results (using least squares — look that up if it's new to you: very useful!)).

5 BST and AVL Trees: 10 min

An AVL tree is a BST such that every node's left and right subtrees have heights differing at most by one.

A. Write the recurrence equations for the minimum number of nodes in an AVL tree of height h. Do they remind you of anything?

B. Start with this BST named `ORIG`, which obeys the AVL balance condition.



Draw the tree after inserting 12 into `ORIG`. If necessary, rotate to maintain the AVL and BST conditions.

Ans.

A. S(h) = S(h-1)+S(h-2) +1, Fibonacci.

B. Pretty snotty double rotation....

```
    20
 10    40
   12  30 50
```

6 Stacks and Queues: 10 min

A *deque* (sometimes *dequeue*), pronounced “deck”, is a double-ended queue, with ’enqueues’ and ’dequeues’ at each end, giving operations that the language Ada calls `Append` (insert at back), `Prepend` (insert at front), `DeleteLast`, and `DeleteFirst`. Also there are two ’peeks’: `LastElement`, `FirstElement`.

A. How would you implement a deque with pointers? (hint: there’s one obvious way). Give the Big-Oh time for the six operations using your implementation.

B. How would you implement a deque with one or more arrays? (hint: several possible ways). Give the Big-Oh time for the six operations.

Ans.

A. Doubly-linked list gives O(1) for all.

B See Wikipedia: Single array, maybe allocated and used from the middle out, (depending on what we know about patterns of use), re-allocating and copying if hit an end. Or a circular buffer. Or lots of smaller arrays allocated as needed and a list of pointers to them. O(1) for all but Append and Prepend ops, which are “amortized” O(1).