CPSC 35 Midterm Exam

Spring 2009

9:30-10:20am, Monday 23 March, 2009
Closed book exam

NAME:______________________________

Problem	Max	Obtained
1	30	
2	20	
3	20	
4	30	
B	5	
Total	100	

Instructions: You have 50 minutes to complete this midterm exam. When doing asymptotic analysis (using big-O notation), please provide as tight of a bound as possible. That is, while it is correct to say that an algorithm that runs in n steps is $O(n^2)$, you should write $O(n)$ if you wish to receive full marks.

30 points **Problem 1:** Data Structures & Algorithm Potpourri :: Please provide **short** answers (3 sentence maximum) for each question.

(a) [*3 points*] If we are using an ArrayList as the main data structure in an implementation of the OrderedList interface, what is the best asymptotic runtime in terms of n, the number of elements in the list, of the method that can be used to find a key in that ArrayList?

O(log n)

(b) [*3 points*] If we push "s", "w", "a", "t" onto a stack, what letter will be returned when we pop() from the stack?

"t"

(c) [*3 points*] If we enqueue "s", "w", "a", "t" onto a queue, what letter will be returned when we dequeue() from the queue?

"s"

(d) [*3 points*] List one advantage of a linked list over an array.

No fixed size, can grow with the amount of data, easy to add data to beginning or end

(e) [*3 points*] List one advantage of an array over a linked list.

direct access to data in middle of data structure

(f) [*5 points*] How many unique binary search trees can be constructed with 3 unique keys? (Hint: Try drawing the trees with a set of 3 random keys.)

5

(g) [*5 points*] True or False: We can have two algorithms p and q such that $q(n) \in O(p(n))$ AND $p(n) \notin O(q(n))$. Explain.

true, let p be in an $O(n^2)$ *algorithm and q be an* $O(n)$ *algorithm.*

(h) [*5 points*] What does the following pseudocode compute when seeded with the root node of a binary tree? Each node in the tree stores the value of an integer called *key* and references to the *left* and *right* children. (Remember that unlike a binary search tree, the keys stored in external nodes of a binary tree are NOT empty.):

```
int compute(Node u)
    if isExternal(u), return u.key
    l <- 0
    r <- 0
    if u.left is not null
         l <- compute(u.left)
    if u.right is not null
         r <- compute(u.right)
    return (l + r)
```

Computes the sum of keys of the leaf/external nodes.

20 points **Problem 2:** Exact and Asymptotic Analysis

```
foo(int N)
    do stuff in A steps
    for i = 1..N
        do stuff in B steps
        for j = 1.. N
            do stuff in C steps

boo(int N)
    do stuff in A steps
    for i = 1..N
        do stuff in B step
        for j = 1.. i
            do stuff in C steps

goo(int N)
    do stuff in A steps
    for i = 1..N
        do stuff in B steps
        for j = 1.. 100
            do stuff in C step
```

(a) [*5 points*] For each of the three methods above (foo(), boo(), and goo()), what is the exact runtime of each method (in steps) in terms of A,B,C, and N? You should not count loop related operations (e.g., declaring, incrementing and comparing i or j).

Answer:
foo(): $A + BN + CN^2$
boo(): $A + BN + C(N(N+1)/2)$
goo(): $A + BN + 100CN$

(b) [*5 points*] For N = 10, which is the fastest algorithm? Please show work.

Answer:
foo(): $100C + 10B + A$
boo(): $55C + 10B + A$ *(Fastest Algorithm)*
goo(): $100C + 10B + A$

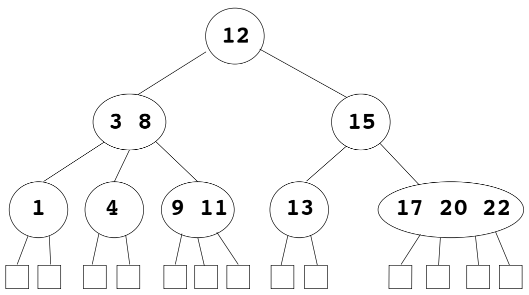
(c) [*5 points*] Assuming that A, B, and C are constant, what is the asymptotic runtime of each method in terms of N?

Answer:
foo(): $O(N^2)$
boo(): $O(N^2)$
goo(): $O(N)$

(d) [*5 points*] As N grows infinitely large, which is the fastest algorithm?

Answer: goo()

20 points **Problem 3:** (2,4) Trees :: For each parts (b) - (d), **start** with the following (2-4) tree:



(a) [*5 points*] There are the two important properties which assures that searching a (2-4) tree for a key will take $O(log\ n)$ time where n is the number of keys in the tree. Name and describe both of them in one sentence each.

SIZE property :: every internal node has at most 4 children. DEPTH Property: all external nodes have the same depth.

(b) [*5 points*] For the given (2-4) tree as shown at the beginning of this question, please draw the resulting tree after we add "16".

Adding 16 creates an OVERFLOW in the rightmost node. We fix this with a SPLIT in which we create two new nodes, (16) and (20,22) AND 17 moves into the parent node such that the parent now contains (15, 17).

(c) [*5 points*] For the given (2-4) tree as shown at the beginning of this question, please draw the resulting tree after we remove "1".

This creates an UNDERFLOW in the leftmost node which we will call u. Since u's adjacent sibling $v = (4)$ is a 2-node, we must FUSE u and v and shift the 3 from the parent node p into this new node. This is done so that p now has one key (8) and two children (3,4) and (9,11).

(d) [*5 points*] For the given (2-4) tree as shown at the beginning of this question, please draw the resulting tree after we remove "4".

This creates an UNDERFLOW in the node that used to contain 4. We will cal it v. Since the right sibling of u (which we will call w) is a 3-node, we can TRANSFER the key 9 from w to the parent p and TRANSFER the key 8 from p to v. In the end, $p = (3, 9)$ and has three children nodes (1), (8), and (11).

30 points **Problem 4:** Set Difference

(a) [*20 points*] Given two arrays, A and B, each storing m and n integers, respectively, write the pseudocode (or Java code) for an algorithm $setDifference(A, B)$ that returns a third array C that contains the integers in A that are not in B. For example, if $A = [7, 4, 9, 12, 2]$ and $B = [5, 12, 3, 7]$, then C should be $[4, 9, 2]$.

Answer:

```
setDifference(int[] A, int[]  B)
   int[] C = int[A.length];
   cnt = 0
   for i = 1.. A.length
     for j = 1..B.length
       if A[i] == B[j]  break  // breaks out of inner loop
     if A[i] != B[j]
       C[cnt] = A[i]
       cnt++;
  return c;
```

(b) [*10 points*] What is the asymptotic runtime of your algorithm in terms of m and n? Explain.

O(mn) since it is a double nested loop. The outer nested loop runs m time while the inner loop runs n times.

5 points **Bonus:** Draw a binary tree that contain the letters “b”, “n”, “o”, “s”, “u” such that the inorder traversal spells “bonus” and the preorder traversal spells “obuns”.

“o” is the root node, “b” is the left child of the root, “u” is the right child of the root. “n” is the left child of “u”. “s” is the right child of “u”.