UNIVERSITY of TORONTO
SCARBOROUGH

Summer 2011 CSCB63
Assignment 1 Sample Solution

Due Date: **June 14th, 9:00 a.m.**

Running Time

1) For each of the following pairs of functions, either $f(n)$ is in $O(g(n))$, $f(n)$ is in $\Omega(g(n))$, or $f(n) = \Theta(g(n))$. For each pair, determine which relationship is correct. Briefly explain your answer.

a. $f(n) = \log n^2$; $g(n) = \log n + 5$

Solution: $f(n) = \Theta(g(n))$ since $\log n^2 = 2 \log n$.

b. $f(n) = n^{1/2}$; $g(n) = \log n^2$

Solution: $f(n)$ is in $\Omega(g(n))$ since n^c grows faster than $\log n^c$ for any c

c. $f(n) = \log^2 n$; $g(n) = \log n$

Solution: $f(n)$ is in $\Omega(g(n))$. Dividing both sides by $\log n$, we see that $\log n$ grows faster than 1

d. $f(n) = 2^n$; $g(n) = 3^n$

Solution: $f(n)$ is in $O(g(n))$. $3^n = 1.5^n 2^n$, and if we divide both sides by 2^n, we see that 1.5^n grows faster than 1

2) Using Big Oh notation, indicate the time requirement of each of the following tasks in the worst case. Describe any assumptions you make.

a. After arriving at a party, you shake hands with each person there.

Solution: O(n) where n is the number of people at the party under the assumptions that there is a fixed maximum time between hand shakes and there is a fixed maximum time for a handshake.

b. Each person in a room shake hands with everyone else in the room.

Solution: O(n^2)where n is the number of people at the party under the assumptions that there is a fixed maximum time between hand shakes and there is a fixed maximum time for a handshake.

c. You climb a flight of stairs.

Solution: O(n) where n is the number of steps under the assumptions that you never take a backward step and there is a fixed maximum time between steps.

d. You slide down the banister.

Solution: O(h) where h is the height of the banister under the assumptions that you never slow down and you reach a limiting speed.

OR

Solution: O(n) where n is the ratio of height to angle of the railing and all forces are assumed constant

e. After entering an elevator, you press a button to choose a floor.

Solution: O(1) under the assumption that you reach a decision and press the button within a fixed amount of time.

OR

Solution: O(n) where n is the number of other people pressing different floors in the elevator assuming that they all take an equal time to press the buttons.

f. You ride the elevator from the ground floor to the nth floor

Solution: O(n) where n is the number of the floor under the assumptions that the elevator only stops at floors, there is a fixed maximum time for each stop, and there is a fixed maximum time that an elevator requires to travel between two adjacent floors.

g. You read a book twice.

Solution: O(n) where n is the number of pages in the book under the assumptions that you never read a word more than twice, you read at least one word in a session, there is a fixed maximum time between sessions, and there is a fixed maximum number of words on a page.

3) Using Big Oh notation, indicate the time requirement (worst case) of each of the following fragments of Java code. Describe any assumptions you make.

a.

```
a = b + c;
d = a + e;
```

Solution: This fragment is $\Theta(1)$.

b.

```
sum = 0;
for( i =0; i < 3; i++)
     for( j=0; j<n; j++)
          sum++;
```

Solution: This fragment is $\Theta(n)$. The outer loop is executed a constant number of times that is independent of n, so it does not count.

c. Assume array A contains n values, `Random` takes constant time, and `sort` takes n log n steps.

```
for(i=0; i<n; i++){
     for(j=0; j<n; j++)
          A[i] = Random(n);
     sort(A , n);
}
```

Solution: This fragment is $\Theta(n^2 \log n)$ since the outer for loop costs n log n for each execution, and is executed n times. The inner loop is dominated by the call to sort.

d. Assume array `A` contains a random permutation of the values from 0 to n-1

```
sum3=0;
for(i=0; i<n; i++)
     for(j=0; A[j] != i; j++)
          sum3++;
```

Solution: For each execution of the outer loop, the inner loop is generated a "random" number of times. However, since the values in the array are a permutation of the values from 0 to $n-1$, we know that the inner loop will be run i times for each value of i from 1 to n. Thus, the total cost $\Theta(n^2)$.

4) For each of the following Java code fragments i) give an analysis of the running time (Big Oh), ii) implement the code in Java, and give the running time for several values of n, iii) compare your analysis with the actual running time. Include a tabular print out of the running times for different values of n you tried.

Notes: - make sure to choose a varying set of values for n from the small to the very large.

- To find the actual running time of a piece of Java code;

```
long lStart = System.currentTimeMillis();
// put code here
long lEnd = System.currentTimeMillis();
System.out.println(" time taken " + lEnd – lStart );
```

a.

```
sum = 0;
for(i=0 ; i < n; i++)
     for(j=0; j < n * n; j++)
          sum++;
```

Solution: The running time is $O(N^3)$.

b.

```
sum=0;
for(i=0; i < n; i++)
     for(j=0; j < i; j++)
          sum++;
```

Solution: The running time is $O(N^2)$.

c.

```
sum=0;
for(i=0 ; i < n; i++)
     for(j=0; j < i * i; j++)
          for(k=0; k < j; k++)
               sum++;
```

Solution: j can be as large as i^2, which could be as large as N^2. k can be as large as j, which is N^2. The running time is thus proportional to $N.N^2.N^2$, which is $O(N^5)$.

5) An Algorithm takes 0.5 ms for input size 100. How large a problem can be solved in 1 min if the running time is the following (assume the same hardware configuration is used in all cases):

a. linear

Solution: Five times as long, or 2.5 ms.

b. O(n log n)

Solution: Slightly more than five times as long.

c. quadratic

Solution: 25 times as long, or 12.5 ms.

d. cubic

Solution: 125 times as long, or 62.5 ms.

Binary Heap

6) A binary heap can either be implemented as a min-heap (pair with minimum key at root) or max-heap (pair with maximum key at root). This question is about max-heap implemented as a tree (not an array). Assuming we are going to insert the following in the same order form left to right:

```
10   5   12   3   2   1   8   7   9   4
```

Draw the resulting complete binary tree -- showing the insertions one at a time.

Solution:

The corresponding level order traversal for the heap:

```
12  9  10  5  4  1  8  7  3  2
```

7) Assuming a binary heap is implemented using a tree data structure (not an array). Give an algorithm to find all nodes less than some value, X. Your algorithm should run in O(K), where K is the number of nodes output. Describe any assumptions you make.

Solution:

Preorder

Tree

8) Describe using pseudo code, a method that takes as input a binary search tree, T, and two keys k1 and k2, which are ordered so that K1 <= K2, and prints all elements X in the tree such that K1 <= Key(X) <= K2. Do not assume any information about the type of keys except that they can be ordered (consistently). Your method should run in O(K+log n) average time, where K is the number of keys printed.

Solution:

```
public void printRange( Key lower, Key upper, BinaryTreeNode t )
{
     if( t != null )
     {
          if( lower.compareTo( t.element ) <= 0 )
               printRange( lower, upper, t.left );
          if( lower.compareTo( t.element ) <= 0 &&
                              t.element.compareTo( upper ) <= 0 )
               System.out.println( t.element );
          if( t.element.compareTo( upper ) <= 0 )
               printRange( lower, upper, t.right );
     }
}
```

The time is $O(K)$ to perform the inorder traversal, if a significant number of nodes are found, and also proportional to the depth of the tree, if we get to some leaves (for instance, if no nodes are found). Since the average depth is $O(\log n)$, this gives an $O(K + \log n)$ average bound.

Dictionary

9) Given input {4371, 1323, 6173, 4199, 4344, 9679, 1989} and a hash function h(x) = x (mod 10), show the resulting:

a. Separate chaining hash table.

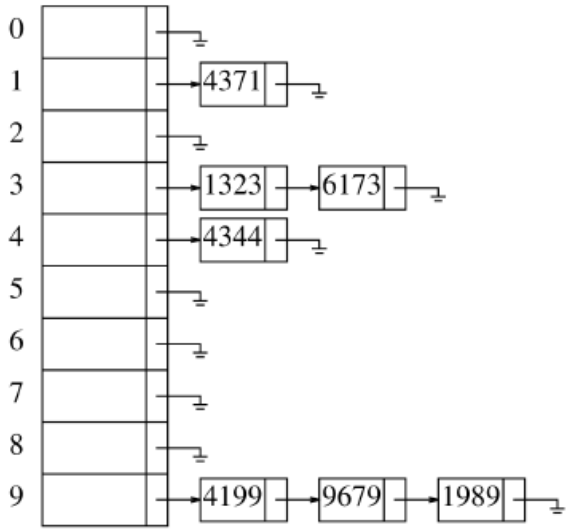
b. Hash table using linear probing.

c. Hash table using quadratic probing.

d. Hash table with second hash function $h_2(x) = 7 - (x \bmod 7)$.

Solution:

(a) On the assumption that we add collisions to the end of the list (which is the easier way if a hash table is being built by hand), the separate chaining hash table that results is shown here.



(b)

0	9679
1	4371
2	1989
3	1323
4	6173
5	4344
6	
7	
8	
9	4199

(c)

0	9679
1	4371
2	
3	1323
4	6173
5	4344
6	
7	
8	1989
9	4199

(d) 1989 cannot be inserted into the table because $hash_2(1989) = 6$, and the alternative locations 5, 1, 7, and 3 are already taken. The table at this point is as follows:

0	
1	4371
2	
3	1323
4	6173
5	9679
6	
7	4344
8	
9	4199

10) Show the result of inserting the following keys into an initially empty extensible hash table with 2 entries per bucket: 10111101, 00000010, 10011011, 10111110, 01111111, 01010001, 10010110, 00001011, 11001111, 10011110, 11011011, 00101011, 01100001, 11110000, 01101111.

Solution: