CSCE 221 Cover Page

Homework Assignment #2

First Name Chris Last Name Comeaux UIN 622006681

User Name cmc236 E-mail address cmc236@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources				
People				
Web pages (provide URL)	www.chegg.com			
Printed material	Lecture Slides			
Other Sources				

I certify that I have listed all the sources that I used to develop the solutions/codes in the submitted work.

On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.

Your Name Chris Comeaux Date 3/24/2016

Homework 2

due March 27 at 11:59 pm.

1. (10 points) Describe (in pseudo code) how to implement the stack ADT using two queues. What is the running time of the push and pop functions in this implementation?

   Let Q1 represent the first queue and Q2 represents the second queue.

   push(T){// T is the key being inserted

   Q1.enqueue(T)}

   pop(T){ //T is the key being removed

   while(Q1.size() >1)

   Q2.enqueue(Q1.dequeue())

   s = Q1.dequeue()

   while(Q2.size() > 0)

   Q1.enqueue(Q2.dequeue())

   return s

   }

   Push: O(1)

   Pop: It will take 2n-1 operations to dequeue n-1 elements from Q1 and enqueue n-1 elements to Q2. Then it will take 1 operation to copy the last element into a variable. It will then take 2n operations to dequeue n elements from Q2 and enqueue n-1 elements to Q2. So f(n) = 2n-1+1+2n = 4n = O(n)

2. (10 points) Solve C-5.8 on p. 224

   To evaluate an expression in post fix form without a recursive function you must use a stack. When the algorithm comes across a number or a variable it will push it on the stack. When the algorithm comes across an operation it will pop off the correct number of operands, according to the operator, perform the operation, and push the answer back on the stack. It will continue in this manner until the whole expression has been visited and the answer is the only element in the stack(using a loop).

3. (10 points) Linked list questions.

(a) Write a recursive function in C++ that counts the number of nodes in a singly linked list.

```
int count = 0; //accumalator initilized in global scope so it is evaluated once
int SinglyLinkedList::size(SListNode* Node){ //pass in the first node of list to start
{
    if(Node==NULL) return count;
    count++; //update accumalator
    return size(Node->getNext()); // call again with next node
}
```

(b)

Write a recurrence relation that represents the running time for your algorithm.

T(n) =T(n-1) +1, T(0) = 0 where n is the number of nodes and k max = n.

(c)

Solve this relation and provide the classification of the algorithm using the Big-O asymptotic notation.

T(n) =T(n-1) +1
T(n) =T(n-2) +2
T(n) =T(n-3) +3
.
.
.
T(n) =T(n-k) +k
===>
T(n) =T(0) +n
T(n) =n = O(n)

4. (10 points)
Write a recursive function that finds the maximum value in an array of integers without using any loops.

```
int findMax(int A[], int size, int Max){
    if(size == 1){
        if(A[0]>Max) return Max;
        else return Max;
    else if(size==0){
        cout<<"range error\n";
        return -1;
    }
    else if(A[size-1]>Max)
        return findMax(A,size-1,A[size-1]);
    else if(A[size-1]<Max)
        return findMax(A,size-1,Max);
}
```

(a) Write a recurrence relation that represents running time of your algorithm.

T(n) = T(n-1)+1, T(1) = 1, kmax=n+1, where n is the number of nodes

(b) Solve this relation and classify the algorithm using the Big-O asymptotic notation.

T(n)=T(n-1)+1
T(n-1) = T(n-2) +1+1
T(n-2) = T(n-3)+1+1+1
.
.
.
T(kmax) = T(1)+n+1 = n+2 = O(n)

5. (10 points) Consider the quick sort algorithm.

(a) Provide an example of
the inputs and the values of the pivot point for the best, worst and average cases for the quick sort.

Average: if the input id unsorted an the pivot point is the maximum or minimum element.
Worst: If the input is sorted and the pivot is maximum or minimum element.
Best: Quick sort is the fastest when the input is sorted or unsorted and the pivot point is in the middle of the list.

(b) Write a recursive relation for running time function and its solution for each case.

Average: If the left side of the pivot has (a x n) elements and the right side has (1-a)n elements. T(n) = T(a x n) + T((1-a)n) +n where a <=0.5 then O(nlog(n))
Worst: T(n) = T(n-1) + n T(1) = 0
.
.
.
T(1) + 2 + 3 + 4 +...+ (n-2) + (n-1) + n = $\sum_{i}^{n} i = 1 - \frac{n(n-1)}{2} = O(n^2)$ (NOTE: i = 2 in sigma)

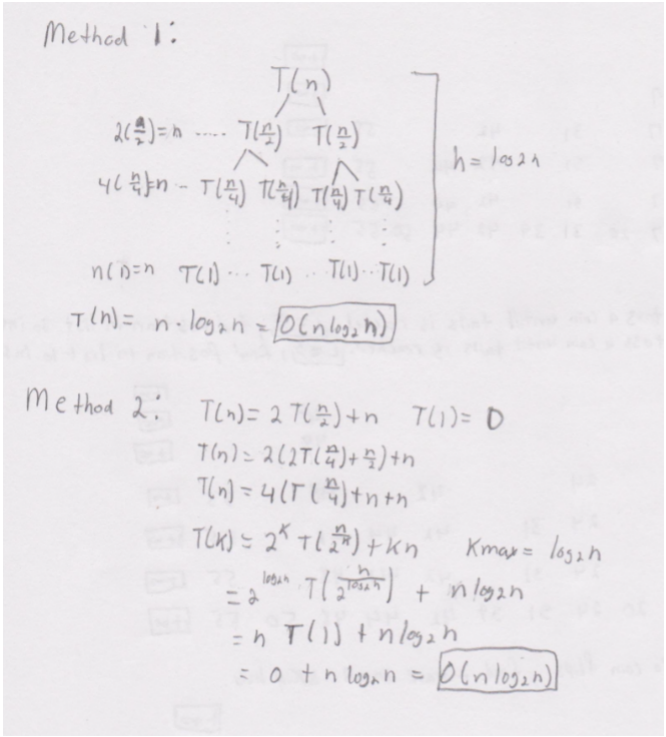
Best: T(n) = $2T(\frac{n}{2})$+n, T(1) =0
Using Master Theorem: a=2, b=2, f(n) = n
$n^{log_2 2} = n = f(n)$, therefore it is case 2.
T(n) = O($n^{log_2 2} * log_2 n$)= O($nlog_2 n$)

6. (10 points) Consider the merge sort algorithm.

(a) Write a recurrence relation for running time function for the merge sort.

The recurrence relation for the running time of merge sort is T(n) = $2T(\frac{n}{2})$+n, T(1) =0

(b) Use two methods to solve the recurrence relation.



(c) What is the best, worst and average running time of the merge sort algorithm? Justify your answer.

Merge sort does not have a best/worst/average case. In every case it runs O($nlog_2 n$). However, the downside is that merge sort must use more memory because the array must be copied everytime.

7. (10 points) R-10.17 p. 493
For the following statements about red-black trees, provide a justification for each true statement and a counterexample for each false one.

   (a) A subtree of a red-black tree is itself a red-black tree.

   Fasle. The root of a red-black tree must be black, however a subtree of a red-black tree may have a red root and therefore cannot be a red-black tree

   (b) The sibling of an external node is either external or it is red.

   True. The black depth of a node is h and the depth of an external node is h+1 so its sibling must either a black external node or a red node.

   (c) There is a unique (2,4) tree associated with a given red-black tree.

   True. A node with 2 red children can be represented by a 4-node, a node with 1 red child can be represented with a 3-node, and a node with no red children can be represented with a 2-node.

   (d) There is a unique red-black tree associated with a given (2,4) tree.

   False. A 3-node can be represented in more than one way in a red-black tree.

8. (10 points) R-10.19 p. 493
Consider a tree T storing 100,000 entries. What is the worst-case height of T in the following cases?

   (a) T is an AVL tree.

   $1.44log_2(100000+1) = 1.44log_2(100001) = 23.9179032 \approx 24$. The worst case height of a AVL tree with 100000 elements would be 24.

   (b) T is a (2,4) tree.

   $log_2(100000+1) = log_2(100001) = 16.609655 \approx 17$. The worst case height of a (2-4) tree with 100000 elements would be 17.

   (c) T is a red-black tree.

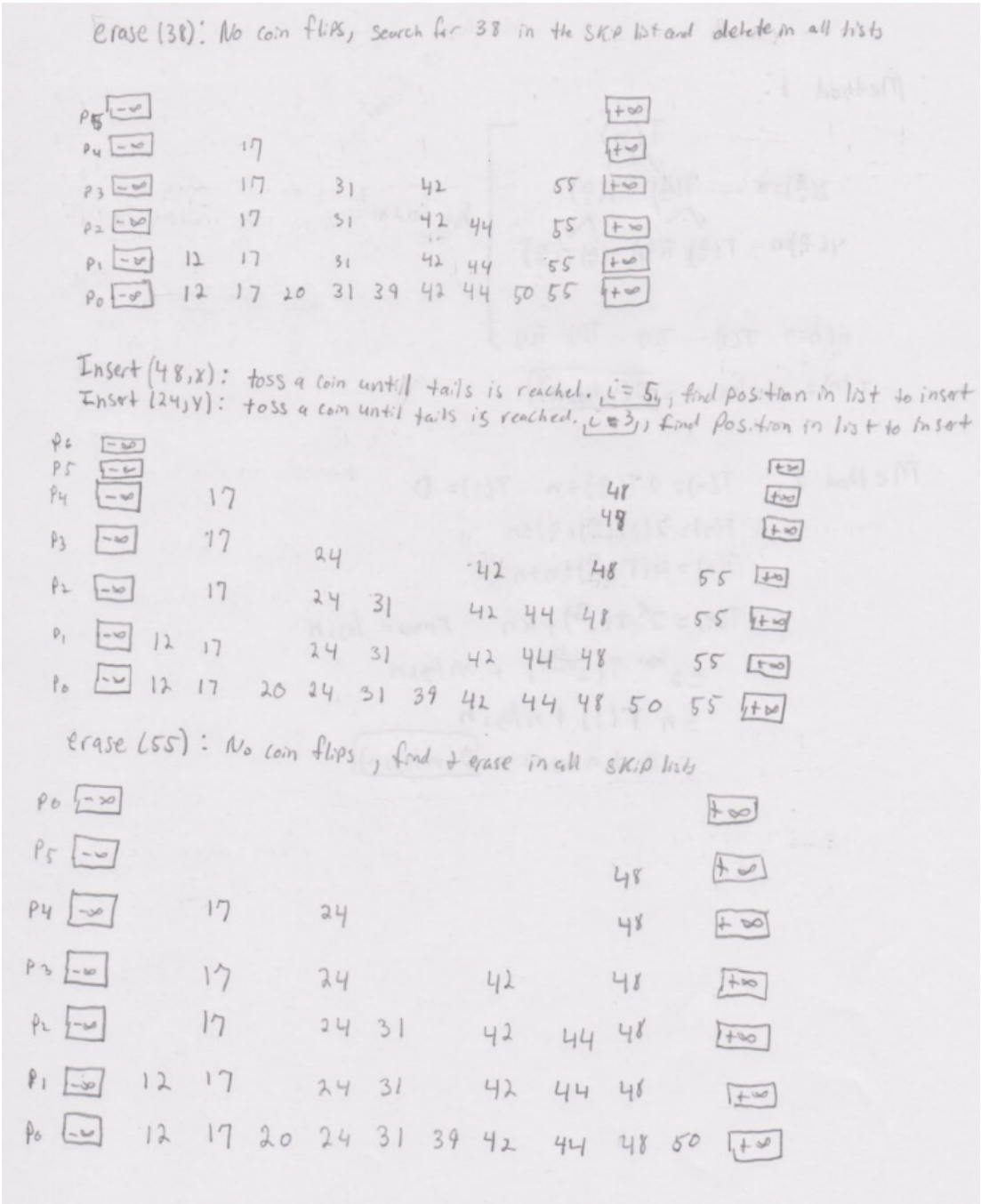
   $2log_2(100000+1) = 2log_2(100001) = 33.21931 \approx 33$. The worst case height of a red-black tree with 100000 elements would be 33.

   (d) T is a binary search tree.

   The worst case height for a binary search tree is O(n) (linear binary search tree). Therefore for 100000 elements, the worst case height is 100000.

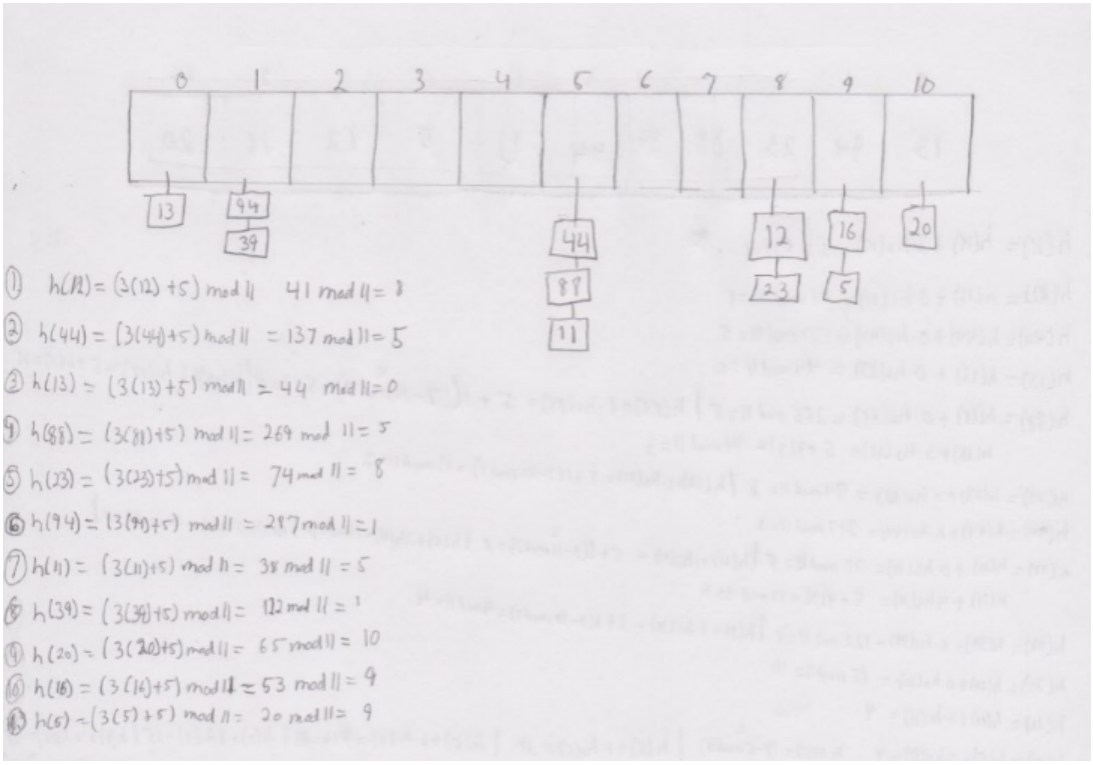
9. (10 points) R-9.16 p. 418

Draw an example skip list that results from performing the following series of operations on the skip list shown in Figure 9.12: `erase(38)`, `insert(48,x)`, `insert(24,y)`, `erase(55)`. Record your coin flips, as well.



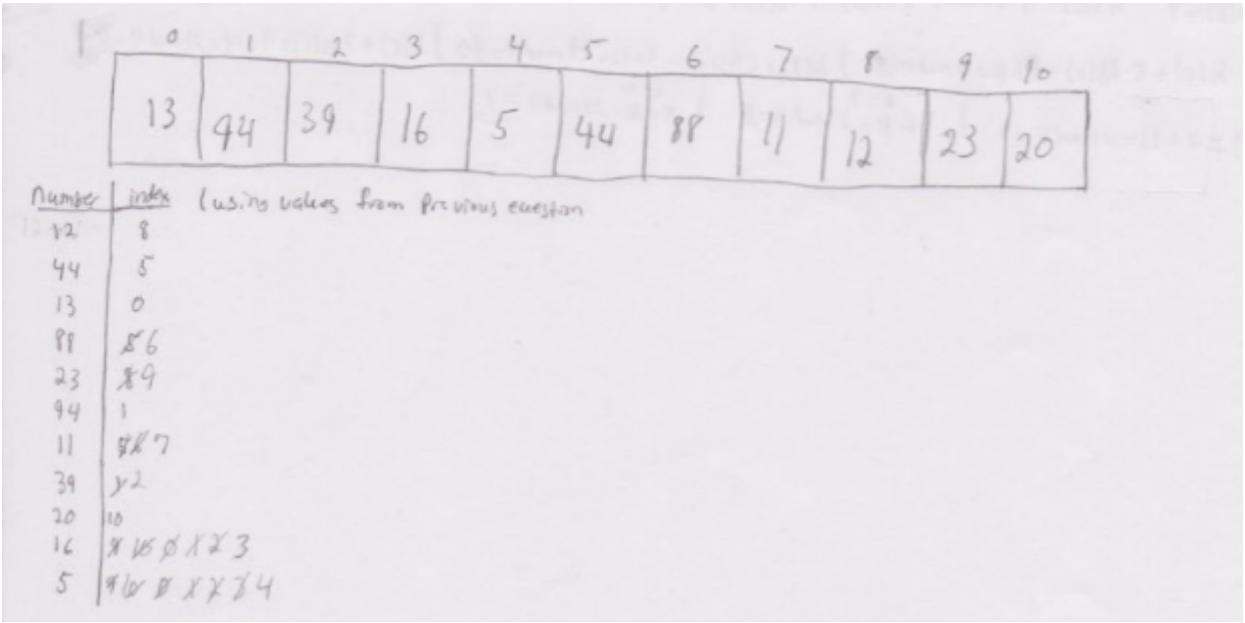
10. (10 points) R-9.7 p. 417

Draw the 11-entry hash table that results from using the has function, $h(k) = (3k + 5) \bmod 11$, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining.



11. (10 points) R-9.8 p. 417

What is the result of the previous exercise, assuming collisions are handled by linear probing?



12. (10 points) R-9.10 p. 417

What is the result of Exercise R-9.7, when collisions are handled by double hashing using the secondary hash function $h_s(k) = 7 - (k \bmod 7)$?