CS3343: Analysis of Algorithms
Notes about Practice Midterm

- This is **just an example** of some of the kinds of questions that could appear on a midterm. Be sure to also review homeworks, lectures, extra practice problems, and quizzes.

- Likewise the actual midterm questions may be either easier or harder than the ones shown here.

- On the actual exam you may consult only inanimate sources. You **may not** read, copy, or rewrite the solutions written by others (including solutions from previous terms or from other sources such as the internet). Any questions should be directed to your cs3343 instructor.

- You should also consider timing yourself. This number of questions here would likely be for a **1 hour 30 minute exam**. Try to complete it within that time frame.

- Remember to annotate all your proofs with comments/text. This is important for ensuring you can receive partial credit for wrong answers.

- We will discuss solutions in the in class review.

- Good luck and have fun! ☺

CS3343: Analysis of Algorithms
Practice Midterm

Annotate all your proofs with comments/text in order to receive full credit.

Name: _____

Question	Points	Score
Growth of Functions	4	
Recursive Algorithm and Analysis	5	
Asymptotic Notation	4	
Binary Search	5	
Probability	6	
Big-Oh Induction	5	
Heaps	3	
Hash Table	3	
Hash Table Probabilities	3	
Total:	38	

Question 1 : **Growth of Functions** . . . . . 4 points

Show that $2n^4 - 5n^3 + 4n \in \Theta(n^4)$ by proving the following:

(a) (2 points) $2n^4 - 5n^3 + 4n \in O(n^4)$ //LHS bigger; $f(n) \leq c \cdot g(n)$

$\Rightarrow 2n^4 - 5n^3 + 4n \leq c \cdot n^4$ //add $5n^3$

$\Rightarrow 2n^4 + 4n \leq c \cdot n^4$ //multi $4n$ by n^3

$\Rightarrow 2n^4 + 4n^4 = 6n^4 \leq c \cdot n^4$

$c =$ 6

$n_0 =$ 1

(b) (2 points) $2n^4 - 5n^3 + 4n \in \Omega(n^4)$ //Make LHS smaller; $f(n) \geq c \cdot g(n)$

└ replace #s to help solve

$\Rightarrow 2n^4 - 5n^3 + 4n \geq c \cdot n^4$ // subtract a $4n$

$\Rightarrow 2n^4 - 5n^3 + \cancel{4n} - \cancel{4n} \geq c \cdot n^4$

$\Rightarrow 2n^4 - 5n^3 \geq c \cdot n^4$ //replace 5 w/ n ($n \geq 5$)

$= 2n^4 - n \cdot n^3 \geq c \cdot n^4$

$= 2n^4 - n^4 \geq c \cdot n^4 = n^4 \geq c \cdot n^4$

$c =$ 1

$n_0 =$ 5

Question 2 : **Recursive Algorithm and Analysis** . . . . . 5 points

Suppose company X has designed a new version of merge sort which uses bubble sort to sort arrays length 100 or less:

Algorithm 1 void mergeSort(int $A[1 \ldots n]$)

---

if $n <= 100$ **then**
  //Sort an array of 100 elements or less using bubble sort
  //Reminder: Bubble sort takes $\Theta(n^2)$.
  bubbleSort(int $A[1 \ldots n]$)
  return;
end if
mergeSort($A[1 \ldots \lceil n/2 \rceil]$); 1
mergeSort($A[\lceil n/2 \rceil + 1 \ldots \lceil n \rceil]$); 2 $\Rightarrow 2T(n/2)$
Merge the two sorted arrays $= \theta(n)$
return;

Assume that merging the two sorted lists takes time $\Theta(n)$.

(a) (3 points) Create a recurrence to represent the run time of our new merge sort from the pseudocode (Be careful with your $f(n)$).

$T(n) =$ $2T(n/2) + \theta(n^2)$ // n^2 is wrong here

$T(n) = 2T(n/2) + n$

(b) (2 points) Use master theorem to solve your above recurrence relation.

$\Rightarrow T(n) = 2T(n/2) + n^2$

$a = 2, \; b = 2, \; f(n) = n^2$

$\Rightarrow n^{\log_2 2} = n^1 \Rightarrow$ $f(n)$ grows faster than $n^{\log_b a}$ // case 3

$\Rightarrow n^2 \in \Omega(n^{1+\varepsilon})$ let $\varepsilon = 1$

$\Rightarrow n^2 \in \Omega(n^2)$ // Regularity cond.

$2(n/2)^2 \leq cf(n)$

$\frac{2n^2}{4} = \frac{1}{2}n^2 \leq c \cdot n^2$

$c = 1/2$

$\Rightarrow$ therefore

$2T(n/2) + n^2 \in \theta(n^2)$ □

Question 3 : **Asymptotic Notation** . . . . . 4 points

Suppose you know following facts about f, g, and h:

$f(n) \leq c_1 g(n)$ for all $n \geq n_1$

$f(n) \leq c_2 h(n)$ for all $n \geq n_2$

(a) (2 points) Prove that there exists a $c > 0$ and $n_0 > 0$ such that:
$f(n) \leq c \min(g(n), h(n))$ for all $n \geq n_0$

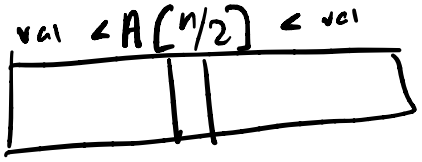
(b) (2 points) $c =$ _____ $n_0 =$ _____

Question 4 : **Binary Search** . . . . . 5 points

The code below does a binary search on an array of n numbers:

Algorithm 2 int binarySearch(int $A[1 \ldots n]$, int val)

```
if A[n/2] == val then
    //Found val at the current location
    return True;
else if n == 1 then
    //val is not in the array
    return False;
else if A[n/2] >= val then
    //val is in first half of the array
    return binarySearch(A[1 ... n/2], val);
elseA[n/2] <= val
    //val is in second half of the array
    return binarySearch(A[n/2 ... n], val);
end if
```



search LHS

search RHS

Question 4 continues...

Let $X_i(A, val)$ be the RV which is 1 if the binary search will have to recurse at least i^{th} to find val; otherwise X_i is 0.

Let $X(A, val)$ be the RV denoting the total number times the binary search recurses to find val in A.

Let's suppose the probability of X_i being 0 is $\frac{2^{i+1}-1}{n}$. For example, the chance of not needing to recurse at all is $p(X_0 = 0) = \frac{2^{0+1}-1}{n} = \frac{2-1}{n} = \frac{1}{n}$. //Finding target at middle

(a) (1 point) What is the expected value of X_i?

$E(X_i) = 1 \cdot p(Xi = 1) + \underbrace{0 \cdot p(Xi = 0)}_{=0} = 1 \cdot \left(1 - \frac{2^{i+1}-1}{n}\right)$ (100%)

⇒ Taking probability of $Xi = 0$ minus 100% to get $Xi = 1$

(b) (2 points) Define X using a sum of many copies of the random variable X_i. For simplicity you can assume n is a power of 2.

Hint: Think carefully about the largest number of recursive calls that the binary search can make before finding val. ↳ generally $\log n$

// Bound of i

$\Rightarrow 0 \le i \le \log_2 n - 1$

$$X = \sum_{i=0}^{\log_2 n - 1} Xi$$

(c) (2 points) Compute the expected value of X using the linearity of expectation.

$$E(X) = E\left(\sum_{i=0}^{\log_2 n - 1} Xi\right) = \sum_{i=0}^{\log_2 n - 1} E(Xi)$$

$$= \sum_{i=0}^{\log_2 n - 1} 1 - \frac{2^{i+1}-1}{n}$$

(high-low) +1

$$\Rightarrow \sum_{i=0}^{\log_2 n - 1} 1 - \sum_{i=0}^{\log_2 n - 1} \frac{2^{i+1}-1}{n}$$ ← split again (remove $\frac{1}{n}$)

$= \log_2 n - 1 - 0) + 1$

$= \log_2 n$

$$\Rightarrow \sum_{i=0}^{\log_2 n - 1} 2^{i+1} - \underbrace{\sum_{i=0}^{\log_2 n - 1} 1}_{= \log_2 n}$$

$i = 0$ ← want to start at 1

$$= \log_2 n - \frac{1}{n}\left(\Rightarrow \sum_{i=1}^{\log_2 n} 2^i - \log_2 n\right)$$

$$\therefore \sum_{i=1}^{\log_2 n} 2^i = \frac{2^{\log_2 n + 1} - 1}{2-1}$$

$$= 2^{\log_2 n + 1} - 1$$

$$= 2^{\log_2 n} \cdot 2^1 - 1$$

$$= n \cdot 2 - 1$$

$$= 2n - 1$$

$$= \log_2 n - \frac{1}{n}\left(2n - 1 - \log_2 n\right) = \log_2 n - \frac{2n-1}{n} - \frac{\log_2 n}{n}$$

O(1)

$\Rightarrow \boxed{E(X) \in O(\log n)}$ $\in \Theta(\log n)$

Question 5 : **Probability** ..... 6 points

Suppose you play a strange game where you flip a fair coin k times. If the coin comes up heads on the i^{th} flip, you win $20i$ dollars and otherwise you lose $4i^2$ dollars (for all i such that $1 \leq i \leq k$).

Let X_i be a random variable denotes the amount of money you win from the i^{th} flip (losing money can be thought of as "winning" negative money $\ddot{\smile}$).

Let X be a random variable denotes the amount of money you win across all k of your flips.

(a) (1 point) Compute the expected value of X_1.

$E(X_1) =$

(b) (1 point) Give a general formula for the expected value of the random variable X_i (where $1 \leq i \leq k$).

$E(X_i) =$

(c) (1 point) Describe the random variable X as the sum of k random variables.

$X =$

Question 5 continues...

(d) (2 points) Derive a general formula for the expected value of X using the linearity of expectation. The formula you find shouldn't contain a $\sum$.
(Hint: remember $\sum_{i=1}^{n} 1 = n$, $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$, and $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$)

$E(X) =$

(e) (1 point) How many times should someone choose to flip the coin (i.e., what k **maximizes** the expected value)?

$k =$

Notice division by 4
lets try $n_0 = 4$ (2 would still work, 4 makes math easier)

Question 6 : **Big-Oh Induction** ..... 5 points

Suppose we define the recursive run time function $T(n)$ such that $T(n) = 10$ (for $n < 2$) and $T(n) = 4T(n/4) + 4n$ (for $n \geq 2$).

Use strong induction to show that there are values for c and n_0 such that $T(n) \leq cn \log_4 n$ for all $n \geq n_0$ (by proving this we can conclude that $T(n) \in O(n \log_4 n)$).

(a) (1 point) **Base case:**

$T(1) = 10$ for $n < 2$

Try $n_0 = 1$; $T(1) \leq c \cdot 1 \cdot \log_4 1$

$10 \leq c \cdot 0 \Rightarrow 10 \leq 0$ ⓧ not true

Try $n_0 = 4$

$T(4) = 4T(4/4) + 4(4) = 4 \cdot 10 + 4(4)$ ($= T(1) = 10$)

$= 40 + 16$

$= 56$

$\Rightarrow T(4) \leq c \cdot 4 \cdot \log_4 4$

$\Rightarrow 56 \leq c \cdot 4 \cdot 1$

$\Rightarrow \frac{56}{4} \leq \frac{c4}{4}$

$c \geq 14$

(b) **Inductive case:**

i. (1 point) What are you assuming is true:

Assume: $T(x) \leq cx \log_4 x$

$\Rightarrow 14 \leq x < n$

ii. (1 point) What are you proving is true:

Prove: $T(n) \leq cn \log_4 n$

iii. (2 points) Complete the proof:

$T(n) \leq \ldots \leq cn \log_4 n$

$\Rightarrow T(n) = 4T(n/4) + 4n$ // $n/4$ satisfies conditions

$\Rightarrow 4(c \cdot n/4 \log_4 (n/4)) + 4n$ // from I.H

$= c \cdot n \log_4 (n/4) + 4n$ (split log)

$= cn \log_4 n - cn \log_4 4 + 4n$ ($\log_4 4 = 1$)

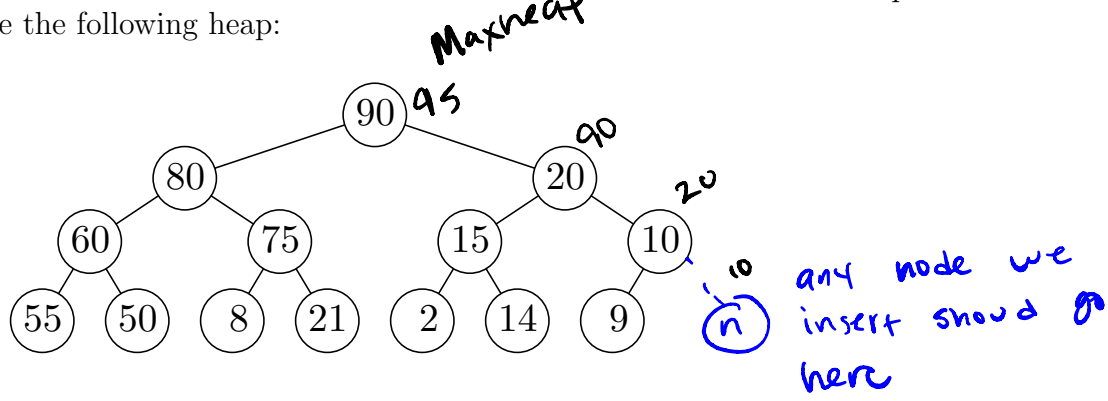
$= cn \log_4 n - cn + 4n$ ∴ $c \geq 14$, thus this will be negative # at least (-1cn) thus just replace w/ 0 as its negative #

$= cn \log_4 n$ □

$= RHS$

Question 7 : **Heaps** ..... 3 points

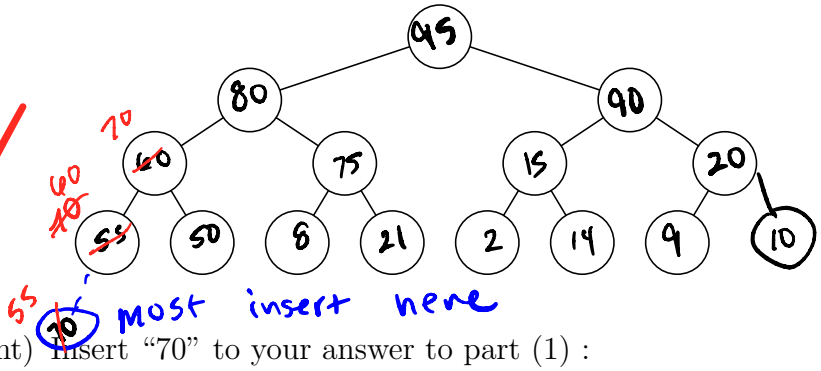
Suppose we have the following heap:



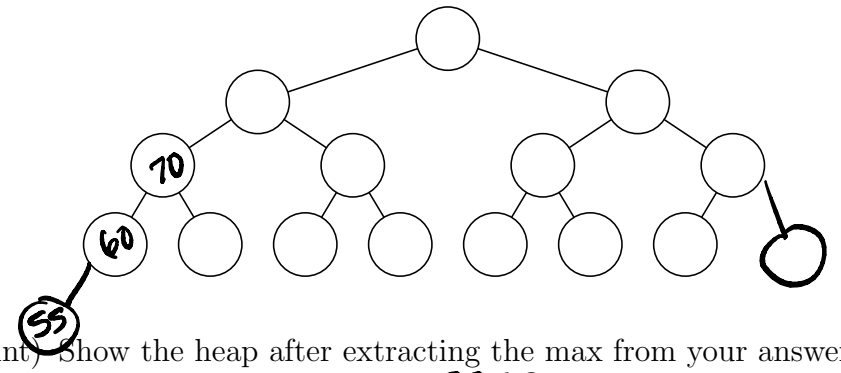
(a) (1 point) Show the resulting heap from inserting "95" (add your node and fill in numbers for the figure below):

insert(95)

only fill in nodes that have changed

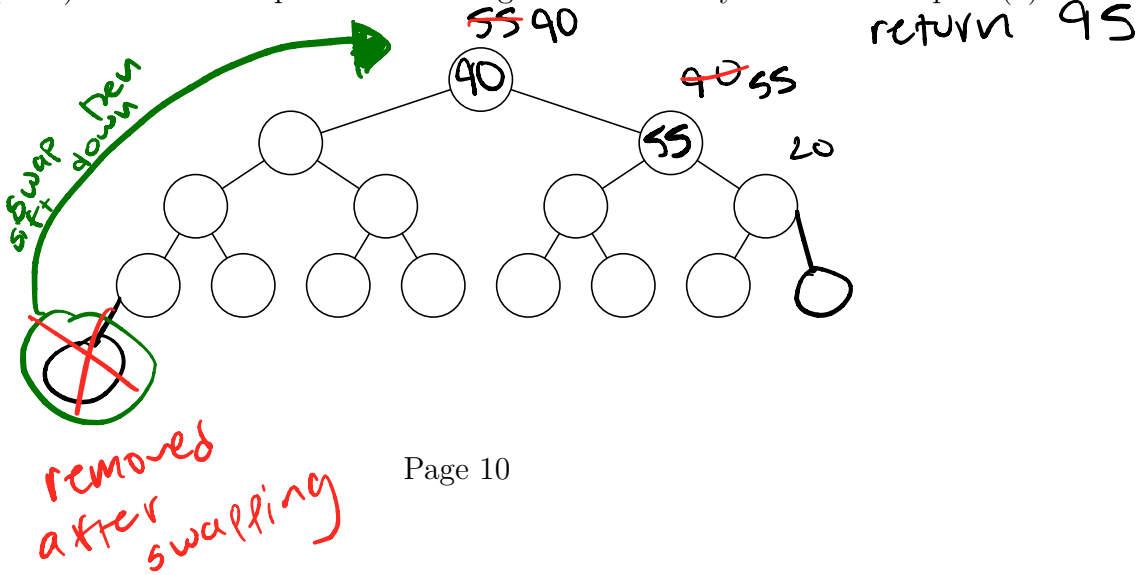


(b) (1 point) Insert "70" to your answer to part (1) :



(c) (1 point) Show the heap after extracting the max from your answer to part (2) :

return 95



Question 8 : **Hash Table** ..... 3 points

(a) (1 point) Consider inserting the keys $34, 21, 18, 16, 37, 25, 30$ into a hash table of length $m = 9$ with the hash function $h(k) = k \bmod m$. Illustrate the result of inserting these keys using chaining (i.e., linked lists) to resolve collisions.

Slot 0	
Slot 1	
Slot 2	
Slot 3	
Slot 4	
Slot 5	
Slot 6	
Slot 7	
Slot 8	

(b) (1 point) Consider inserting the keys $34, 21, 18, 16, 37, 25, 30$ into a hash table of length $m = 9$ with the hash function $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$ where $A = 0.5$. Illustrate the result of inserting these keys using chaining (i.e., linked lists) to resolve collisions.

Slot 0	
Slot 1	
Slot 2	
Slot 3	
Slot 4	
Slot 5	
Slot 6	
Slot 7	
Slot 8	

(c) (1 point) Our above hash function didn't seem to work to well. Identify what problem occurred and what caused it?

Question 9 : **Hash Table Probabilities** ..... 3 points

Suppose 2 keys are inserted into an empty hash table with 3 slots.

Unfortunately, our hash function was poorly chosen. Specifically, 50% entries are inserted into slot 0 of the table. Entries are distributed into the other 2 slots following a normal distribution (i.e., 25% each).

What is the probability of exactly 0 collisions occurring?