Question 1: 10 points (3 + 4 + 3) 10

a.

Show that $n^2 + 2n$ is $o(2^n)$

let $f(n) = n^2+2n$ and $g(n) = 2^n$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^2+2n}{2^n}$

Applying L'Hopital's rule

$\lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{2n+2}{2^n c}$

$\lim_{n\to\infty} \frac{f''(n)}{g''(n)} = \lim_{n\to\infty} \frac{2+0}{2^n c^2} = \lim_{n\to\infty} \frac{2}{2^n c^2} = 0$ ✓

Hence $f(n)$ is $o(g(n))$

b. Determine whether f is O(g) or not. Show your work.

i) f = 2^(n + 1), g = 2^n

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2^{(n+1)}}{2^n} = \lim_{n\to\infty} \frac{2^n \cdot 2^1}{2^n} = \lim_{n\to\infty} \frac{2}{1} = 2$ ✓

Since $2 > 0$, hence f is $O(g)$

ii) f = 2^(2n), g = 2^n

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2^{2n}}{2^n} = \lim_{n\to\infty} \left(\frac{2^2}{2}\right)^n = \lim_{n\to\infty} \left(4/2\right)^n = \infty$ ✓

Hence f is not $O(g)$

Question 1: (continued)

c. Show that $n^2 + 2n$ is O(n3). Do NOT use f/g; use n_0 and c.

let $f(n) = n^2 + 2n$ and $g(n) = n^3$

$f(n)$ is $O(g(n))$ iff there is a positive integer c such that $f(n) \leq c\, g(n)$

let $c = 1$

$\Rightarrow n^2 + 2n \leq n^3 \Rightarrow n + 2 \leq n^2 \Rightarrow n^2 \geq n + 2$

hence $n_0 \geq 2$.

$\therefore f(n)$ is $O(g(n))$ for $n \geq n_0$

Question 3: (continued)

b. Use the QuickSelect algorithm to manually compute the 5th smallest element of the array [1, 5, 23, 0, 8, 4, 33]. Assume that the rightmost element is used as the pivot in each case. Show what happens in each self-call, indicating the new input array and the current value of k.

S = (1, 5, 23, 0, 8, 4, 33), k = 5, E = 33

L: [1, 5, 23, 0, 8, 4]   E: [8]   G: [ ]   k < |L|

S = (1, 5, 23, 0, 8, 4), k = 5, E = 4

L: [1, 0]   E: [4]   G: [5, 23, 8]

k ≥ |L| + |E|

k' = 5 - 2 - 1 = 2

S = (5, 23, 8), k = 2, E = 8

L: [5]   E: [8]   G: [23]

k = |L| + |E|

since |L| < k ≤ |L| + |E|

return 8

c. Use RadixSort, with two bucket arrays and radix = 11, to sort the following array: [63, 1, 48, 53, 24, 10, 12, 30, 100, 115, 17]. Show all steps of the sorting procedure. Then explain why the running time is O(n).

① Remainder (A[i] % 11) → take mod 11 of each element and store the element in the remainder bucket at the position of A[i] % 11

Remainder bucket

0	1	2	3	4	5	6	7	8	9	10
	100, 12, 1	24		48	115	17		30, 63	53	10

② Quotient → Take mod 11 of each element in the remainder bucket and store the element in the quotient bucket at position of the quotient

Quotient bucket

0	1	2	3	4	5	6	7	8	9	10
10, 1	17, 12	30, 24		53, 48	63				100	115

return [1, 10, 12, 17, 24, 30, 48, 53, 63, 100, 115]

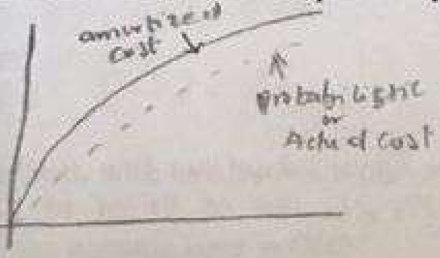
Question 3: 11 points (3 + 2 + 3 + 3) 8,5 + 0,5 = 9

a. (i) Explain why Amortized analysis is better than Average Case analysis using probabilistic method.

Amortized analysis is better than Average case analysis using probabilistic method because Amortized analysis simpler and faster to compute whereas probabilistic method is difficult and expensive. Amortized analysis covers all possible range of inputs but probabilistic method may not have all possible inputs since generating input data is expensive.

−1,5 → −1

(ii) From Average case analysis standpoint, does Amortized analysis provide upper bound or lower bound? Explain why.



From the graph above, amortized analysis provides upper bound to the actual or probabilistic method and also has a lower bound of the actual or probabilistic analysis. Thus amortized cost can't go below the probabilistic cost.

$\sum_{i=0}^{n} c \le \sum_{i=0}^{n} \hat{c}$

Need to explain/compare with $\sum x p(x)$

Question 4: 13 points (4 + 4 + 5) 13

a. Show how Quicksort is not stable by using in-place random partitioning algorithm and the following 4 numbers {4a, 4b, 4c, 4d} (show all steps).

$4_a, 4_b, 4_c, \underline{4_d}$ let 4d be the pivot

↑i ↑j

Since 4a = 4c = 4d hence i and j are stuck so we swap and advance one step each

$4_c, 4_b, 4_a, 4_d$

↑i ↑j

Both i and j are stuck so we swap and advance one step each.

$4_c, 4_b, 4_a, 4_d$

↑j ↑i Finally we swap i and the pivot

$4_c, 4_b, 4_d, 4_a$

Since 4a which was first is now last it shows that quicksort is not stable ✓

b. (i) Is mathematics decidable? Explain the Halting problem in your own words (no need to prove).

NO, mathematics is not decidable.

For a running program, we can't say whether the program will halt, terminate successfully or will run infinitely. There is an algorithm for the halting problem but no algorithm to solve it. ✓

(ii) Is Mathematics Sound? Explain your answer with an example.

No, mathematics is not sound.

There are some equations which are true but their proof shows as false

Eg. the proof for p^P shows false. ✓

Question 4: (continued)

c. Use Decision tree and binary tree basic ideas to prove the following theorem:

"Every comparison based sorting algorithm has, for each n, running on input of size n, a worst case in which its running time is Ω(nlog n)".

How does comparison based sorting achieves Ω(nlog n) compared to O(n^2) running time of inversion bound sorts like insertion sort and bubble sort? Explain your answer.

The max number of leaves for binary tree is 2^h ; where h is the height of the tree

The number of leaves for a decision tree is $n!$

Since decision tree is a subset of binary tree,

$\Rightarrow n! \leq 2^h$

From Stirling's theorem $n! \leq \left(\frac{n}{e}\right)^n$

$\Rightarrow \left(\frac{n}{e}\right)^n \leq 2^h$ taking log on both sides.

$n \log(n/e) \leq h \log 2$

$\Rightarrow n\log n - n\log e \leq h$

$\therefore h \geq n\log n - n\log e$

$\Rightarrow h$ is $O(n\log n)$

Since the depth of a leave is the maximum height which also the number of decisions to reach the leave it shows that the running time is $\Omega(n\log n)$

Comparison based sorting achieves $\Omega(n\log n)$ running time because it uses divide and conquer which reduces the number of comparisons hence reduced running time.

with divide and conquer the height of the tree is logn therefore for n operations $T(n) = O(n\log n)$

Question 2: 12 points (4 + 3 + 5) 12

For each of the following recurrences, derive an expression for the running time using iterative, substitution or Master Theorem.

a. Consider the following recurrence algorithm **[Use Master Theorem – See LAST Page]**

```
    Procedure (Array A, int n) {
If (n == 0) return True;            1+1 = 2

    for i = 1...n {                 1+n+n-1 = 2n
        A[i] = A[i] + 1;            1+1+1+1 = 4
    }
    if (n>1) Procedure(A, n/2);     1+1+T(n/2) = 2+T(n/2)
} // end procedure
```

i. (2 points) Write a recurrence equation for T(n)

$T(n) = T(n/2) + 2n + 8$

ii. (2 points) Solve recurrence equation using Master's method i.e. give an expression for the runtime T(n).

$a = 1,\ b = 2,\ k = 1$

since $a < b^k$ thus $1 < 2^1$

$T(n) = \Theta(n)$

b. Use **Iterative method**

$$\begin{cases} T(n) = 3T(n-1) + 1 \\ T(1) = 0 \end{cases}$$

$T(n) = 3T(n-1) + 1$

$f(2) = 3(3T(n-2) + 1) + 1)$

$= 3^2 T(n-2) + 3 + 1$

$f(3) = 3^2(3T(n-3) + 1) + 3 + 1$

$= 3^3 T(n-3) + 3^2 + 3 + 1$

$\Rightarrow f(k) = 3^k T(n-k) + 3^{k-1} + 3^{k-2} + \dots + 3^0$

$= 3^k T(n-k) + \sum_{i=1}^{k-1} 3^i$

To reach the base case, $n-k=1 \Rightarrow n = k+1$

$T(k) = 3^k T(k+1-k) + \sum_{i=1}^{k-1} 3^i$

$= 3^k T(1) + \sum_{i=1}^{k-1} 3^i = 3^k(0) + \sum_{i=1}^{k-1} 3^i$

$= \sum_{i=1}^{k-1} 3^i$

$= \frac{3^{k-1+1} - 1}{3-1} = \frac{3^k - 1}{2}$

Since $k = n-1$

$\Rightarrow T(n) = \frac{3^{n-1} - 1}{2} = \frac{3^n \cdot 3^{-1} - 1}{2}$ Hence $T(n)$ is $O(3^n)$

Question 4: 13 points ([1 +1 +2] + 4 + 5)

11

a) What is the worst case running time of Quicksort? Can you improve the worst case running time of quicksort? If so, describe how.

the running time of Quick sort is $O(n^2)$ It can be improved and became $O(n \log n)$. You can improve the running time of quick sort by choosing a good pivot. a good pivot is less than $3n/4$ of elements.

but how ?

-1

b. (i) Explain with an example what is meant by "Mathematics is not sound".

Mathematics is not sound. Actually the only true statement should be proved but in mathematics we can prove also the false assertion

Example

P	$\overline{P}$	$P \wedge \overline{P}$
1	0	0
0	1	0

(ii) Is Mathematics consistent? Explain with a proof.

Mathematics is not consistent, an assertion can not be prove both true and false. For instance $2^0 \to 1$

$2^1 \to 2$

$2^2 \to 4$

⋮

$2^\infty \to$ undefined ?

Question 4: 13 points ([1 +1 +2] + 4 + 5)

11

a) What is the worst case running time of Quicksort? Can you improve the worst case running time of quicksort? If so, describe how.

the running time of Quick sort is $O(n^2)$ It can be improved and became $O(n \log n)$. You can improve the running time of quick sort by choosing a good pivot. a good pivot is less than $3n/4$ of elements.

but how ?

-1

b. (i) Explain with an example what is meant by "Mathematics is not sound".

Mathematics is not sound. Actually the only true statement should be proved but in mathematics we can prove also the false assertion

Example

P	$\overline{P}$	$P \wedge \overline{P}$
1	0	0
0	1	0

(ii) Is Mathematics consistent? Explain with a proof.

Mathematics is not consistent, an assertion can not be prove both true and false. For instance $2^0 \to 1$

$2^1 \to 2$

$2^2 \to 4$

$\vdots$

$2^\infty \to$ undefined ?

Question 3: (continued)

c. Use RadixSort, with two bucket arrays and radix = 11, to sort the following array: [63, 1, 48, 53, 24, 10, 12, 30, 100, 115, 17]. Show all steps of the sorting procedure. Then explain why the running time is O(n).

r[ ]

0	1	2	3	4	5	6	7	8	9	10
	100 12 1	24		48	115	17		30 53 63		10

① Take each array element and find its mod 11 and put it in array r[ ]

q[ ]

0	1	2	3	4	5	6	7	8	9	10
10 1	17 12	30 24		53 48	63				100	115

② move elements from array r[ ] to q[ ] but by considering its quotient by dividing with radix

③ write the return array from left to right bottom to up

Sorted array : [1, 10, 12, 17, 24, 30, 48, 53, 63, 100, 111]

the running time is O(n) because it takes

O(n) for initializing arrays
O(n) to copy from array remainder to quotient
O(n) to return the result
O(n)+O(n)+O(n) = 3n which is O(n)

11 ⇒ 9

Question 3: 11 points (5 + 3 + 3)

(a) Assume you are creating an array data structure that has a fixed size of n. You want to backup this array after every so many insertion operations. Unfortunately, the backup operation is quite expensive, it takes n time to do the backup. Insertions without a backup just take 1 time unit.

(i) How frequently can you do a backup and still guarantee that the amortized cost of insertion is O(1)?

You can generate a total time of backup without assigning every element of array a fixed time. For instance you can make a total time of backup of whole array T and time of every operation will be $T/_{\#\,operation} = O(1)$. Another way is to cost more time for insertion to cover the loss of backup time.

-1

(ii) Prove that you can do backups in O(1) amortized time. Use the accounting method for your proof.

Let's say $c(add) = 1$ and $\hat{c}(add) = 1$
$c(clear) = 2$ $\hat{c}(clear) = 6$

For n operations we gain as the profit is $k + 4m$ (k for add, m for clear

Extra profit ?

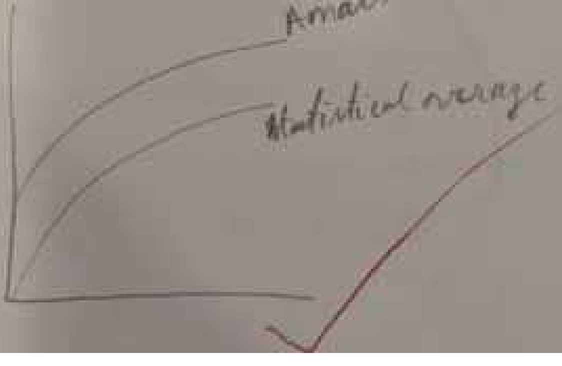
and $\sum_{i=1}^{n} c(s_i) \leq \sum_{i=1}^{n} \hat{c}(s_i)$

$\sum_{i=1}^{n} \hat{c}(s_i) = \sum_{i=1}^{n} 6n = 6 \quad O(n)$

-1

(b) Explain why Amortized analysis is better than Average Case analysis using probabilistic method.

Amortized analysis is better than Average case analysis using probabilistic method because amortized analysis is simple and faster while average case analysis is very complex dealing with big data. Additional to that amortized running time is upper bound of statistical average running time



Question 2: (continued)

c. **Use Induction to show that**

$$D(n) = \begin{cases} 0 & \text{if } n = 1 , \\ D(n/2) + \lg n & \text{if } n = 2^k \text{ and } k \geq 1 , \end{cases}$$

has the solution $D(n) = (\lg n)(\lg n + 1)/2$.

① let's prove Base case

$n = 1 \qquad D(n) = \dfrac{(\log n)(\log n + 1)}{2}$

$D(1) = \dfrac{\log 1 (\log 1 + 1)}{2}$

$D(1) = 0$ Proved

② Assume $n = 2^k$ is proved let's prove for $n = 2^{k+1}$

$D(\frac{n}{2}) + \log n = D(\frac{2^{k+1}}{2}) + \log 2^{k+1}$

$= D(2^{k+1} \cdot 2^{-1}) + \log 2^{k+1}$

$= D(2^k) + \log 2^{k+1} \quad (1)$

let's replace $D(2^k)$ with its equivalent because we know t

$D(n) = \dfrac{(\log n)(\log n + 1)}{2}$

(1) becomes : $\dfrac{(\log 2^k)(\log 2^k + 1)}{2} + \log 2^{k+1}$

$= \dfrac{(\log 2^k)(\log 2^k) + \log 2^k + 2 \log 2^{k+1}}{2}$

$= \dfrac{(\log 2^{k+1})(\log 2^{k+1} + 1)}{2}$ Proved

Question 1: (continued)

c. Give a Big O estimate for $f(x) = (x^3 + 4)\log(x^2 + 1) + 4x^3$

① since $\lim_{x \to \infty} \frac{x^3+4}{x^3} = \lim_{x \to \infty} \frac{x^3(1+\frac{4}{x^3})}{x^3} = 1 > 0$

x^3+4 is $O(x^3)$

② $x^2+1 \leq 2x^2$ when $x > 1$

and $\log x^2 = 2\log x$ which means that

$\log(x^2+1)$ is $O(\log x)$

③ $\lim_{x \to \infty} \frac{4x^3}{x^3} = 4 > 0$

x^3 is $O(x^3)$

$f(x)$ is $O(\max(x^3 \log x, x^3))$

$f(x)$ is $O(x^3 \log x)$

Question 2: 12 points (4 + 3 + 5)

12

For each of the following recurrences, derive an expression for the running time using iterative, substitution or Master Theorem.

a. Consider the following recurrence algorithm **[Use Master Theorem – See LAST Page]**

```
long power(long x, long n)
   if (n==0) return 1;
   if (n==1) return x;
   if ((n % 2) == 0)
      return power(x*x, n/2);
   else
      return power(x*x, n/2) * x;
```

Assume n is power of 2.

i. (2 points) Write a recurrence equation for T(n)

$$\begin{cases} T(n) = 1 & n = 0 \\ T(n) = T(n/2) + 3 \end{cases}$$

ii. (2 points) Solve recurrence equation using Master's method i.e. give an expression for the runtime T(n).

$a = 1$
$b = 2$
$c = 3$
$k = 0$

$\Rightarrow$ $b^k = 2^0 = 1$
$a = b^k$ so $T(n) = \Theta(n^0 \log n) = \Theta(\log n)$

b. Use **Iterative method**

$$\begin{cases} T(n) = 3T(n-1) + 1 \\ T(1) = 0 \end{cases}$$

$T(n) = 3T(n-1) + 1$

$T(n) = 3[3T(n-1-1)+1] + 1 = 3^2 T(n-2) + 3 + 1$

$T(n) = 3^2(3T(n-2-1)+1) + 3 + 1 = 3^3 T(n-3) + 3^2 + 3 + 1$

⋮

k^{th} step $T(n) = 3^k T(n-k) + 1 + 3 + 3^2 + 3^3 \ldots 3^{k-1}$

$T(n) = 3^k T(n-k) + \sum_{i=1}^{k-1} 3^i$

Assume $n - k = 1$
$k = n - 1$

$T(n) = \frac{3^{n-1-1+1} - 1}{3-1}$

$T(n) = \frac{3^{n-1} - 1}{2} = \frac{3^n \cdot 3^{-1} - 1}{2} \Leftrightarrow \underline{O(3^n)}$ running time is O

8

Question 1: 10 points (3 + 3 + 4)

a.

Show that $n^2 + 2n$ is $o(2^n)$

$$\lim_{n\to\infty} \frac{n^2+2n}{2^n} \overset{\lim}{=} \frac{2n+2}{c_1 \cdot 2^n}$$ "by applying L'Hôpital's rule"

$$= \lim_{n\to\infty} \frac{2}{c_2 \cdot 2^{2n}} = \lim_{n\to\infty} \frac{0}{c_3\, 2^{3n}} = 0$$

$\therefore f(n) = n^2+2n$ is $o(2^n)$

b. Determine whether f is O, o, Big omega or small omega of g where

f(n) = n ^ lg m and g(n) = m ^ lg n; Show your reasoning / work.

$F(n) = n^{\lg m}$, $g(n) = m^{\lg n}$

$$\lim_{n\to\infty} \frac{n^{\lg m}}{m^{\lg n}} = \lim_{n\to\infty} \frac{\lg m \cdot n^{(\lg m)-1}}{m^{\lg n} \cdot C_1 \cdot \frac{1}{n}}$$

$\boxed{\text{let } \log m = k \rightarrow 2^k = m}$

$$= \lim_{n\to\infty} \frac{k n^{k-1}}{2^{k \lg n} \cdot C/n}$$? taking L'Hôpital's rule k times:

$$\cancel{= \lim_{n\to\infty} \frac{k!}{C_2^{\lg n}}} = \lim_{n\to\infty} \frac{k!}{2^{c_2 \lg n} \cdot C_3/n^k}$$

$f(n) = n^k$, $g(n) = 2^{k \cdot \lg n}$

-2 Since $\lg_a n < n$, $a \geq 2$ (Logarithms laws)

$\therefore g(n) = 2^{k \lg n} < 2^{kn}$ $\therefore g(n)$ is $o(2^{kn})$ let $2^{kn} = h(n)$

-Take $\lim_{n\to\infty} \frac{n^k}{2^{kn}}$: apply L'Hôpital's rule k-times:

$$= \lim_{n\to\infty} \frac{k!}{c \cdot 2^{k^2 n}} = 0 \quad \therefore f(n) \text{ is } o(h(n))$$?

$\therefore f(n)$ is $O(g(n))$

Question 1: (continued)

c. Give a Big O estimate for f(x) = (x^3 + 2) log(x^2 + 1) + 4x^3

1- Since $\lim_{x \to \infty} \frac{x^3+2}{x^3} = \lim_{x \to \infty} \frac{1 + 2/x^3}{1} = 1 \therefore f_1(x) = x^3 + 2$ is $O(x^3) \to$ 1

2- Since x^2+1 eventually equals x^2

$\therefore \log(x^2+1) \approx \log x^2 = 2 \log x$

since $2 \log x \leq c \log x \to c \geq 2 \quad \therefore f_2(x) = \log(x^2+1)$ is $O(\log x)$ $\to$ 2

3- Since $\lim_{x \to \infty} \frac{4x^3}{x^3} = 4 \to \therefore f_3(x) = 4x^3$ is $O(x^3) \to$ 3

since $f(x) = f_1(x) \cdot f_2(x) + f_3(x)$, from 1, 2, 3 :

~~$\therefore f(x)$ is $O(x^3$~~

$\therefore f(x)$ is $\max(O(x^3 \cdot \log x), O(x^3))$

$\therefore f(x)$ is $O(x^3 \cdot \log x)$

9.5

Question 2: 12 points (4 + 3 + 5)

For each of the following recurrences, derive an expression for the running time using iterative, substitution or Master Theorem.

a. Consider the following recurrence algorithm **[Use Master Theorem – See LAST Page]**

Code		Operations performed
`Procedure (Array A, int n) {`		
`If (n == 0) return True;`	2	Comparison + return
`for i = 1...n {`	1+n+2n	initialize counter + "n" comparisons + "n" increments + "n" assignment of increment
`A[i] = A[i] + 1;`	4 n	
`}`		
`if (n>1) Procedure(A, n/2);`	2+ T(n/2)	Comparison + Call + running time of call
`} // end procedure`		

i. (2 points) Write a recurrence equation for T(n)

$$T(n) = \begin{cases} 2 & n=0 \\ T(n/2) + 7n + 3 \end{cases}$$

ii. (2 points) Solve recurrence equation using Master's method i.e. give an expression for the runtime T(n).

From the equation above:

$a = 1,\ b = 2,\ c = 7,\ k = 1,\ d = 2 \Rightarrow a = 1 < b^k = 2^1 = 2$

$\therefore T(n)$ is $\Theta(n)$

b. Use **Iterative method**

$$\begin{cases} T(n) = 3T(n-1) + 1 \\ T(1) = 0 \end{cases}$$

$T(n) = 3T(n-1) + 1$

$= 3(3T(n-2)+1)+1$

$= 3(3(3T(n-3)+1)+1)+1$

$= 3^3 \cdot T(n-3) + 3^2 + 3^1 + 3^0$ Observing the pattern:

$\therefore T(n) = 3^k \cdot T(n-(k)) + 3^{k-1} + 3^{k-2} + 3^{k-3} + \dots + 3^1 + 3^0$

$= 3^k\, T(n-(k)) + \sum_{i=0}^{k-1} 3^i$

$= 3^k\, T(n-(k)) + \dfrac{3^k - 1}{3-1}$ Plug in $k = n-1$:

$\therefore T(n) = 3^{n-1}\, T(1) + \dfrac{3^{n-1}-1}{3-1}$ Since $T(1) = 0$:

$\therefore T(n) = \dfrac{3^{n-1}-1}{2}$

$\boxed{\therefore T(n) \text{ is } O(3^n)}$

Question 2: (continued)

c. Use Induction to show that

If $T(n) = rT(n-1) + a$, $T(0) = b$, *and* $r \neq 1$ *then*

$$T(n) = r^n b + a\frac{1-r^n}{1-r} \quad \text{—} 1$$

For all nonnegative integer a.

1 Base case: Plug in $n=0$ into equation 1 above:

$T(0) = r^0 b + a\frac{1-r^0}{1-r} = b + a\left(\frac{1-1}{1-r}\right) = b \rightarrow$ Base case proven.

2 Induction case: assume equation 1 is true for "n" and try to prove it for $n+1$. That is, assume

$r \cdot T(n-1) + a = r^{n} \cdot b + a\frac{1-r^{n}}{1-r}$? / 2.

and try to prove:

$r \cdot T(n) + a = r^{n+1} \cdot b + a\frac{1-r^{n+1}}{1-r}$

R.H.S $= r^{n+1} \cdot b + a\frac{1-r^{n+1}}{1-r}$

?

Needs to be derived this using original equation with mathematical manipulations!

-2.5

9

Question 3: 11 points (4 + 3 + 4)

a. Suppose we perform a sequence of stack operations on a stack whose size never exceeds k. After every k operations, we make a copy of the entire stack for backup purposes. Show that the cost of n stack operations, including copying the stack, is O(n) by assigning suitable amortized costs to the various stack operations

- Let actual cost be as follows: $C_{pop} = 1$, $C_{push} = 1$, $C_{copy} = k$, since each cell is copied at cost of 1.
- Let amortize cost be as follows: $\hat{C}_{pop} = 3$, $\hat{C}_{push} = 3$, $\hat{C}_{copy} = 0$ $=n$
- After "n" operations (Pop/push) the total cost will be $\left(\sum_{i=1}^{n} C_i + C_{copy} * \frac{n}{k}\right)$ because every $\frac{n}{k}$ times a copy will be made
- After "n" operation, the amortized cost is $3n = \sum_{i=1}^{n} \hat{C}_i$
- Amortized cost - Actual cost $= \sum_{i=1}^{n} \hat{C}_i - \sum_{i=1}^{n} C_i + \frac{n}{k} C_{copy}$

$= 3n - \cancel{2}n - \frac{n}{k}K = 3n - n - n = n$?

∴ Cost of "n" operations is $n \rightarrow$ is $O(n)$

∴ Cost per operation $= O(n/n) \rightarrow O(1)$

-1

b. Explain why Amortized analysis is better than Average Case analysis using probabilistic method.

Because probabilistic approach needs a complicated analysis and good estimation of the expected inputs and involves a lot of math. Amortized analysis covers all cases with a lot simpler effort. Its results may be not as solid as probabilities, but considering the simplicity, it is very good.

✓

Question 3: 11 points (4 + 3 + 4)

c. Use RadixSort, with two bucket arrays and radix = 11, to sort the following array: [63, 1, 48, 53, 24, 10, 12, 30, 100, 115, 17]. Show all steps of the sorting procedure. Then explain why the running time is O(*n*).

Remainder

R[ ]

	100 12 1	24		48	115		17	30 63	53	10
0	1	2	3	4	5	6	7	8	9	10

The table above is filled as follows:
R[A[i] mod 11] = A[i]   where A[i] is the i^{th} element of the input array and R[ ] is the element of the remainder bucket array.

Quotient

Q[ ]

10 1	17 12	30 24		53 48	63				100	115
0	1	2	3	4	5	6	7	8	9	10

Each ~~quo~~ element of the quotient array above "Q[i]" is filled using the formula: Q[A[i]/11] = A[i]
But here A[i] is collected from the ~~buckets in~~ remainder bucket array from left to right, and ~~down~~ bottom to top.

Finally, we obtain the sorted array by collecting the numbers from the quotient bucket; left to right & bottom ~~down~~ to top:

A_{sorted} = [1, 10, 12, 17, 24, 30, 48, 53, 63, 100, 115]

Running time analysis?

11.5

Question 4: 13 points (4 + 4 + 5)

a. Show how Quicksort is not stable by using in-place random partitioning algorithm and the following 4 numbers {4a, 4b, 4c, 4d} (show all steps).

1- Pick Pivot = 4b

2- Swap it with last element:

A[ ]

4a	4_d	4_c	4_b
i		j	

3- Set pointers i, j as shown above.

4- A[i] & A[j] are swapped because ~~they are both ≤~~ A[i] ≥ pivot and A[j] ≤ pivot, thus they are both stuck.

After swapping, j is decremented and i is incremented. Array is as below:

A[ ]

4_c	4_d	4a	4b
	i j		

5- 4_d is swapped with itself and i is incremented, j is decremented, but now i > j. So, we stop and swap back the pivot with A[i]:

4_c	4_d	4_a	4_b
	j	i	

→

4c	4d	4b	4a

please see back of paper

b. (i) Is mathematics decidable? Explain the Halting problem in your own words (no need to prove).

Mathematics in NOT decidable because given a Program P, we cannot tell if this program will halt, run finitely and stop with a result, or will be stuck in an infinite loop. The previous was the description of the halting problem, and it is the reason why math is undecidable. We have an algorithm to describe the halting problem, but none for solving it.

(ii) Is Mathematics Consistent? Explain your answer with an example.

Mathematics is not consistent.

Example ?

-1

Question 4: (continued)

c. Use Decision tree and binary tree basic ideas to prove the following theorem:

"Every comparison based sorting algorithm has, for each n, running on input of size n, a worst case in which its running time is Ω(nlog n)".

How does comparison based sorting achieves Ω(nlog n) compared to O(n^2) running time of inversion bound sorts like insertion sort and bubble sort? Explain your answer.

- A decision tree has $n!$ leaves (n: input size)
- A binary tree has max 2^h leaves (h: tree height)
- We know that a decision tree is $\subseteq$ binary tree

$\therefore$ No. of leaves in decision tree is $\leq$ that of binary tree

$$\therefore n! \leq 2^h \quad \text{take Log :}$$

$$\log(n!) \leq \log 2^h$$

From Stirling's approximation: $n! > \left(\frac{n}{e}\right)^n$

$$\therefore \log\left(\frac{n}{e}\right)^n \leq h$$

$$\therefore h \geq n\log n - n\log e \quad \therefore \boxed{h \text{ is } \Omega(n.\log n)}$$

* Since "h" is the height, with represents the max. depth of any leave, which also represents the no. of comparisons in a decision tree, then no. of comparisons is $\Omega(n \log n)$.

- Inversion sorting makes no. of comparisons $\geq$ no. of inversions in the array. No. of inversions in array with size(n) is $\binom{n}{2} = \frac{n(n-1)}{2}$, which is $O(n^2)$. That's why inversion sorting has running time of $O(n^2)$.

→ (Divide & conquer)

40 - 0.5

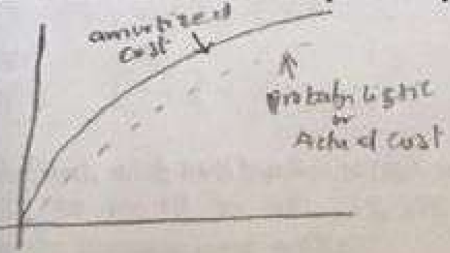
Question 3: 11 points (3 + 2 + 3 + 3) 8,5 + 0,5 = 9

a. (i) Explain why Amortized analysis is better than Average Case analysis using probabilistic method.

Amortized analysis is better than Average case analysis using probabilistic method because Amortized analysis simpler and faster to compute whereas probabilistic method is difficult and expensive. Amortized analysis covers all possible range of inputs but probabilistic method may not have all possible inputs since generating input data is expensive.

−1,5 → −1

(ii) From Average case analysis standpoint, does Amortized analysis provide upper bound or lower bound? Explain why.



From the graph above, amortized analysis provides upper bound to the actual or probabilistic method and also has a lower bound of the actual or probabilistic analysis. Thus amortized cost can't go below the probabilistic cost

$\sum_{i=0}^{n} c \leq \sum_{i=0}^{n} \hat{c}$

Need to explain/compare with $\sum x p(x)$

Question 1: 10 points (3 + 4 + 3) 10

a.

Show that $n^2 + 2n$ is $o(2^n)$

let $f(n) = n^2+2n$ and $g(n) = 2^n$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^2+2n}{2^n}$

Applying L'Hopital's rule

$\lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{2n+2}{2^n c}$

$\lim_{n\to\infty} \frac{f''(n)}{g''(n)} = \lim_{n\to\infty} \frac{2+0}{2^n c^2} = \lim_{n\to\infty} \frac{2}{2^n c^2} = 0$

Hence $f(n)$ is $o(g(n))$

b. Determine whether f is O(g) or not. Show your work.

i) f = 2^(n + 1), g = 2^n

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2^{(n+1)}}{2^n} = \lim_{n\to\infty} \frac{2^n \cdot 2^1}{2^n} = \lim_{n\to\infty} \frac{2}{1} = 2$

Since $2 > 0$, hence f is $O(g)$

ii) f = 2^(2n), g = 2^n

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2^{2n}}{2^n} = \lim_{n\to\infty} \left(\frac{2^2}{2}\right)^n = \lim_{n\to\infty} \left(\frac{4}{2}\right)^n = \infty$

Hence f is not $O(g)$

Question 1: (continued)

c. Show that $n^2 + 2n$ is O(n3). Do NOT use f/g; use n_0 and c.

let $f(n) = n^2+2n$ and $g(n) = n^3$

$f(n)$ is $O(g(n))$ iff there is a positive integer c such that $f(n) \leq c\, g(n)$

let $c = 1$

$\Rightarrow n^2+2n \leq n^3 \Rightarrow n+2 \leq n^2 \Rightarrow n^2 \geq n+2$

hence $n_0 \geq 2$.

$\therefore f(n)$ is $O(g(n))$ for $n \geq n_0$ ✓

Question 2: (continued)

c. **Use Induction to show that**

$$\textit{If } T(n) = rT(n-1) + a,\ T(0) = b,\ \textit{and } r \neq 1 \textit{ then}$$

$$T(n) = r^n b + a\frac{1-r^n}{1-r}$$

For all nonnegative integer a.

Base case

$T(0) = r^0 b + a\left(\frac{1-r^0}{1-r}\right) = b + a\left(\frac{1-1}{1-r}\right) = b + a(0) = b$ hence base case is true

Inductive case

Assuming $T(k) = r^k b + a\left(\frac{1-r^k}{1-r}\right)$ is true then prove for $T(k+1)$

By definition,

$T(k+1) = rT(k) + a$ since $T(k)$ is already assumed to be true,

$T(k+1) = r\left(r^k b + a\left(\frac{1-r^k}{1-r}\right)\right) + a$

$= r^{k+1} b + a\left(\frac{r - r^{k+1}}{1-r}\right) + a$

$= r^{k+1} b + \frac{a(r - r^{k+1}) + a(1-r)}{1-r}$

$= r^{k+1} b + a\left(\frac{r - r^{k+1} + 1 - r}{1-r}\right)$

$= r^{k+1} b + a\left(\frac{1 - r^{k+1}}{1-r}\right)$ which is expected

Hence the equation is true

Question 3: (continued)

b. Use the QuickSelect algorithm to manually compute the 5th smallest element of the array [1, 5, 23, 0, 8, 4, 33]. Assume that the rightmost element is used as the pivot in each case. Show what happens in each self-call, indicating the new input array and the current value of k.

S = (1, 5, 23, 0, 8, 4, 33), k = 5, E = 33

L [1, 5, 23, 0, 8, 4]   E [8]   G [ ]   k < |L|

S = (1, 5, 23, 0, 8, 4), k = 5, E = 4

L [1, 0]   E [4]   G [5, 23, 8]

k ≥ |L| + |E|

k' = 5 - 2 - 1 = 2

S = (5, 23, 8), k = 2, E = 8

L [5]   E [8]   G [23]

k = |L| + |E|

since |L| < k ≤ |L| + |E|

return 8

c. Use RadixSort, with two bucket arrays and radix = 11, to sort the following array: [63, 1, 48, 53, 24, 10, 12, 30, 100, 115, 17]. Show all steps of the sorting procedure. Then explain why the running time is O(n).

① Remainder (A[i] % 11) → take mod 11 of each element and store the element in the remainder bucket at the position of A[i] % 11

Remainder bucket

0	1	2	3	4	5	6	7	8	9	10
	100, 12, 1	24		48	115	17		30, 63	53	10

② Quotient → Take mod 11 of each element in the remainder bucket and store the element in the quotient bucket at position of the quotient

Quotient bucket

0	1	2	3	4	5	6	7	8	9	10
10, 1	17, 12	30, 24		53, 48	63				100	115

return [1, 10, 12, 17, 24, 30, 48, 53, 63, 100, 115]

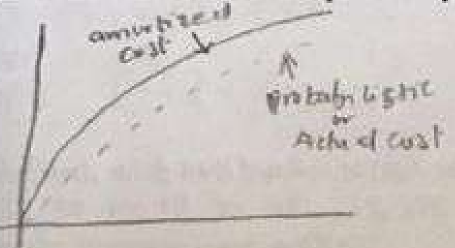
Question 3: 11 points (3 + 2 + 3 + 3) 8,5 + 0,5 = 9

a. (i) Explain why Amortized analysis is better than Average Case analysis using probabilistic method.

Amortized analysis is better than Average case analysis using probabilistic method because Amortized analysis ~~simpler~~ and faster to compute whereas probabilistic method is difficult and expensive. Amortized analysis covers all possible range of inputs but probabilistic method may not have all possible inputs since generating input data is expensive.

−1,5 → −1

(ii) From Average case analysis standpoint, does Amortized analysis provide upper bound or lower bound? Explain why.



From the graph above, amortized analysis provides upper bound to the actual or probabilistic method and also has a lower bound of the actual or probabilistic analysis. Thus amortized cost can't go below the probabilistic cost.

$\sum_{i=0}^{n} c \leq \sum_{i=0}^{n} \hat{c}$

Need to explain/compare with $\sum x p(x)$

Question 4: 13 points (4 + 4 + 5) 13

a. Show how Quicksort is not stable by using in-place random partitioning algorithm and the following 4 numbers {4a, 4b, 4c, 4d} (show all steps).

4a, 4b, 4c, 4d let 4d be the pivot

i j

Since 4a = 4c = 4d hence i and j are stuck so we swap and advance one step each

4c, 4b, 4a, 4d

i j

Both i and j are stuck so we swap and advance one step each.

4c, 4b, 4a, 4d

j i Finally we swap i and the pivot

4c, 4b, 4d, 4a

Since 4a which was first is now last it shows that quicksort is not stable

b. (i) Is mathematics decidable? Explain the Halting problem in your own words (no need to prove).

NO, mathematics is not decidable.

For a running program, we can't say whether the program will halt, terminate successfully or will run infinitely. There is an algorithm for the halting problem but no algorithm to solve it.

(ii) Is Mathematics Sound? Explain your answer with an example.

No, mathematics is not sound.

There are some equations which are true but their proof shows as false

Eg. the proof for P^P shows false.

Question 4: (continued)

c. Use Decision tree and binary tree basic ideas to prove the following theorem:

"Every comparison based sorting algorithm has, for each n, running on input of size n, a worst case in which its running time is Ω(nlog n)".

How does comparison based sorting achieves Ω(nlog n) compared to O(n^2) running time of inversion bound sorts like insertion sort and bubble sort? Explain your answer.

The max number of leaves for binary tree is 2^h, where h is the height of the tree

The number of leaves for a decision tree is $n!$

Since decision tree is a subset of binary tree,

$\Rightarrow n! \leq 2^h$

From Stirling's theorem $n! \leq \left(\frac{n}{e}\right)^n$

$\Rightarrow \left(\frac{n}{e}\right)^n \leq 2^h$ taking log on both sides.

$n\log(n/e) \leq h\log 2$

$\Rightarrow n\log n - n\log e \leq h$

$\therefore h \geq n\log n - n\log e$

$\Rightarrow h$ is $O(n\log n)$

Since the depth of a leave is the maximum height which also the number of decisions to reach the leave it shows that the running time is $\Omega(n\log n)$

Comparison based sorting achieves $\Omega(n\log n)$ running time because it uses divide and conquer which reduces the number of comparisons hence reduced running time

with divide and conquer the height of the tree is logn therefore for n operations $T(n) = O(n\log n)$

Question 2: 12 points (4 + 3 + 5) 12

For each of the following recurrences, derive an expression for the running time using iterative, substitution or Master Theorem.

a. Consider the following recurrence algorithm **[Use Master Theorem – See LAST Page]**

```
    Procedure (Array A, int n) {

If (n == 0) return True;                1+1 = 2

    for i = 1...n {                     1+n+n-1 = 2n
        A[i] = A[i] + 1;                1+1+1+1 = 4
    }
    if (n>1) Procedure(A, n/2);         1+1+T(n/2) = 2+T(n/2)
}  // end procedure
```

i. (2 points) Write a recurrence equation for T(n)

$T(n) = T(n/2) + 2n + 8$

ii. (2 points) Solve recurrence equation using Master's method i.e. give an expression for the runtime T(n).

$a = 1,\ b = 2,\ k = 1$

since $a < b^k$ thus $1 < 2^1$

$T(n) = \Theta(n)$

b. Use **Iterative method**

$$\begin{cases} T(n) = 3T(n-1) + 1 \\ T(1) = 0 \end{cases}$$

$T(n) = 3T(n-1) + 1$

$f(2) = 3(3T(n-2) + 1) + 1$

$= 3^2 T(n-2) + 3 + 1$

$f(3) = 3^2(3T(n-3) + 1) + 3 + 1$

$= 3^3 T(n-3) + 3^2 + 3 + 1$

$\Rightarrow f(k) = 3^k T(n-k) + 3^{k-1} + 3^{k-2} + \dots + 3^0$

$= 3^k T(n-k) + \sum_{i=1}^{k-1} 3^i$

To reach the base case, $n - k = 1 \Rightarrow n = k+1$

$T(k) = 3^k T(k+1-k) + \sum_{i=1}^{k-1} 3^i$

$= 3^k T(1) + \sum_{i=1}^{k-1} 3^i = 3^k(0) + \sum_{i=1}^{k-1} 3^i$

$= \sum_{i=1}^{k-1} 3^i$

$= \frac{3^{k-1+1} - 1}{3-1} = \frac{3^k - 1}{2}$

Since $k = n-1$

$\Rightarrow T(n) = \frac{3^{n-1} - 1}{2} = \frac{3^n \cdot 3^{-1} - 1}{2}$ Hence $T(n)$ is $O(3^n)$

Question 4: (continued)

c.

(i) Compare Mergesort and Quicksort using all key features and their advantages and disadvantages?

when the input is not too big Quick sort is quicker and faster than merge sort but in large data Mergesort is better because it's worse time is O(n logn) while Quick sort is $O(n^2)$. Merge Sort is stable but Quicksort is not stable in general. Mergesort is not in place but Quick sort it is in place when its partition is in place.

(ii) Why Quicksort, in general performs better than Mergesort. Explain with an example using solutions for running time T(n) for both methods.

Quicksort in general perform better than Mergesort because mergesort takes much time to merge elements while Quick sort doesn't merge element in final step instead it makes only union because elements is already sorted in different nodes.

-1