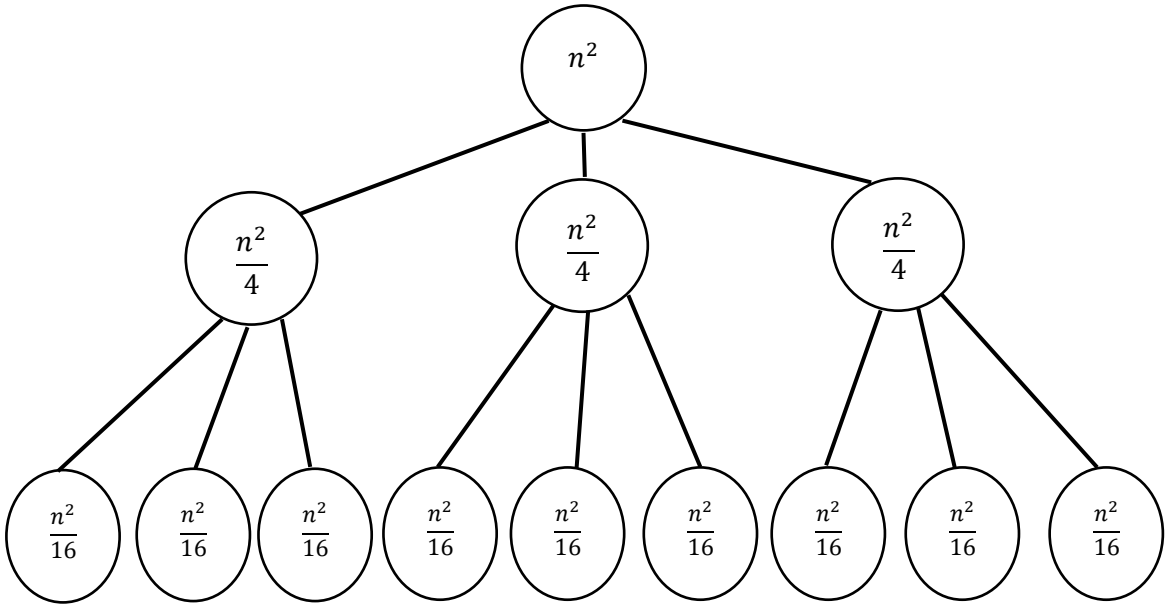
PART I

1. $T(n) = 3 \cdot T\left(\frac{n}{2}\right) + n^2$

- A sketch of tree



- A chart

depth	size	#nodes	Work per node	Total work
0	n	1	n^2	n^2
1	$\frac{n}{2}$	3^1	$(\frac{n}{2})^2$	$3^1 \cdot (\frac{n}{2})^2$
2	$\frac{n}{4}$	3^2	$(\frac{n}{4})^2$	$3^2 \cdot (\frac{n}{4})^2$
…	…	…	…	…
i	$\frac{n}{2^i}$	3^i	$(\frac{n}{2^i})^2$	$3^i \cdot (\frac{n}{2^i})^2$
$\log_2 n$	1	n	d	nd

- A summation giving the total work of the recurrence

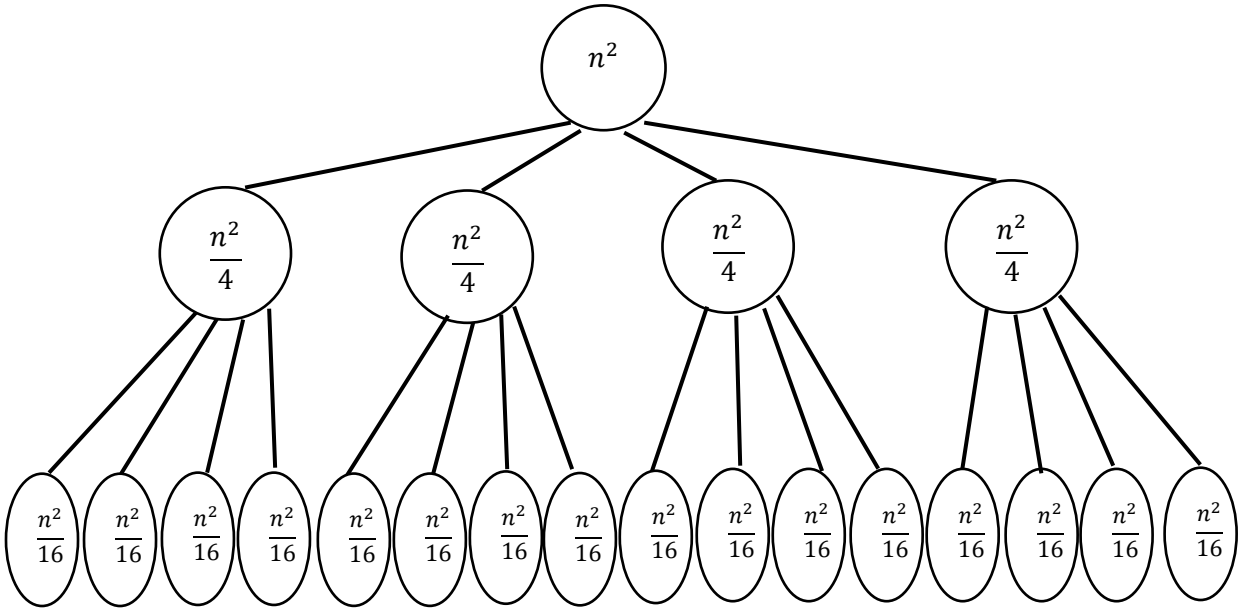
$$dn + \sum_{i=0}^{\log_2 n - 1} (\frac{3}{4})^i n^2$$

- A closed-form asymptotic solution

$$dn + \sum_{i=0}^{\log_2 n - 1} (\frac{3}{4})^i n^2 = \Theta(n^2)$$

2. $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n^2$

- A sketch of tree



- A chart

depth	size	#nodes	Work per node	Total work
0	n	1	n^2	n^2
1	$\frac{n}{2}$	4^1	$(\frac{n}{2})^2$	$4^1 \cdot (\frac{n}{2})^2 = n^2$
2	$\frac{n}{4}$	4^2	$(\frac{n}{4})^2$	$4^2 \cdot (\frac{n}{4})^2 = n^2$
…	…	…	…	…
i	$\frac{n}{2^i}$	4^i	$(\frac{n}{2^i})^2$	$4^i \cdot \left(\frac{n}{2^i}\right)^2 = n^2$
$\log_2 n$	1	n	d	nd

- A summation giving the total work of the recurrence

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

- A closed-form asymptotic solution

$$dn + \sum_{i=0}^{\log_2 n - 1} n^2 = dn + n^2 \log_2 n = \Theta(n^2 \log n)$$

3. $T(n) = T\left(\frac{n}{3}\right) + \log n$

$a = 1, b = 3, f(n) = \log n$

$c_{crit} = \log_b a = \log_3 1 = 0$

$c = 0 \ \because \ f(n) = n^0 \log n$

$case\ \#2 \ \because c = \ c_{crit}$

$f(n) = \Theta\left(n^{c_{crit}} \log^k n\right)\ where\ k = 1 \geq 0$

$\therefore T(n) = \Theta\left(n^{c_{crit}} \log^{k+1} n\right) = \Theta(\log^2 n)$

4. $T(n) = 7T\left(\frac{n}{2}\right) + n^2$

$a = 7, b = 2, f(n) = n^2$

$c_{crit} = \log_b a = \log_2 7$

$c = 2, f(n) = n^2$

$case\ \#1 \ \because c < \ c_{crit}$

$\therefore T(n) = \Theta(n^{c_{crit}}) = \Theta\left(n^{\log_2 7}\right)$

5. $T(n) = 26T\left(\frac{n}{5}\right) + 12n^2$

$a = 26, b = 5, f(n) = n^2$

$c_{crit} = \log_b a = \log_5 26$

$c = 2, f(n) = 12n^2$

$case\ \#1 \ \because c < \ c_{crit}$

$\therefore T(n) = \Theta(n^{c_{crit}}) = \Theta\left(n^{\log_5 26}\right)$

6. $T(n) = 9T\left(\frac{n}{3}\right) - 6n^2 \log^2 n$

$a = 9, b = 3, f(n) = -6n^2 \log^2 n$

$\therefore Inadmissible\ equation!\ \ because\ f(n)\ is\ not\ positive$

7. $T(n) = 9T\left(\frac{n}{3}\right) + \frac{n^2}{\log n}$

$a = 9, b = 3, f(n) = \frac{n^2}{\log n}$

$c_{crit} = \log_b a = \log_3 9 = 2$

$c = 2, k = -1 \because f(n) = \Theta(n^2 \log^{-1} n)$

$case\ \#2 - b \because c = c_{crit}\ \&\ k = -1$

$\therefore T(n) = \Theta(n^{c_{crit}} \log\log n) = \Theta(n^2 \log\log n)$

8. $T(n) = 11T\left(\frac{n}{4}\right) + 2n^{\log_3 11}$

$a = 11, b = 4, f(n) = 2n^{\log_3 11}$

$c_{crit} = \log_b a = \log_4 11$

$c = \log_3 11, f(n) = 2n^{\log_3 11}$

$case\ \#3 \because c > c_{crit}$

$\therefore T(n) = \Theta(f(n)) = \Theta\left(2n^{\log_3 11}\right)$

9. $T(n) = 3T\left(\frac{n}{2}\right) + n^2 \log\log n$

$a = 3, b = 2, f(n) = n^2 \log\log n$

$c_{crit} = \log_b a = \log_2 3$

$c = 2 \because f(n) = n^2 \log\log n$

$case\ \#3 \because c > c_{crit}$

$\therefore T(n) = \Theta(f(n)) = \Theta(n^2 \log\log n)$

PART II

10.

FOO

i++

if i mod b = 0

READ(X, i, A)

BAR

j++

if j mod b = 0

READ(Y, j, B)

11.

Needed WRITE function call for n size is $n \div b = \frac{n}{b}$

Needed READ function call for $\frac{n}{2}$ size is $\frac{n}{2} \div b = \frac{n}{2b}$ + 1(one more read is needed at the end of array)

Since there are two $\frac{n}{2}$ so the total is $\left(\frac{n}{2b} + 1\right) \cdot 2 = \frac{n}{b} + 2$

Therefore, total WRITE function call is $\frac{n}{b}$ and total READ function call is $\frac{n}{b} + 2$

12.

* Case #1: An array A of size n stored in the cloud

$T(n) = 2T\left(\frac{n}{2}\right) + \frac{2n}{b} + 2$

$a = 2, b = 2, f(n) = \frac{2n}{b} + 2$

$c_{crit} = \log_b a = \log_2 2 = 1$

$c = 1 \ \because f(n) = \frac{2}{b}n^1 + 2$

$case\ \#2 \ \because c = \ c_{crit}$

$f(n) = \Theta\left(n^{c_{crit}} \log^k n\right)\ where\ c_{crit} = \ 1, k = 0$

$\therefore T(n) = \Theta\left(n^{c_{crit}} \log^{k+1} n\right) = \Theta(n \log n)$

* Case #2: An array of size n held entirely in your computer's memory

If there is enough memory space and then we can assume that chunk size is n. Just 2 times (constant time) will be needed to read and write memory.

$T(n) = 2T\left(\frac{n}{2}\right) + 2$

$a = 2, b = 2, f(n) = 2$

$c_{crit} = \log_b a = \log_2 2 = 1$

$c = 0, f(n) = 2n^0$

$case\ \#1 \ \because c < \ c_{crit}$

$\therefore T(n) = \Theta(n^{c_{crit}}) = \Theta(n)$

13.

In the case of insufficient memory, a divide-and-conquer algorithm should be selected. Since MergeSort is a divide-and-conquer algorithm, it can be sorted using partial memory, whereas heapsort is not divide-and-conquer, so using partial memory is impossible. Therefore, when sorting in cloud sorting model, merge sort should be used.

PART III

14.

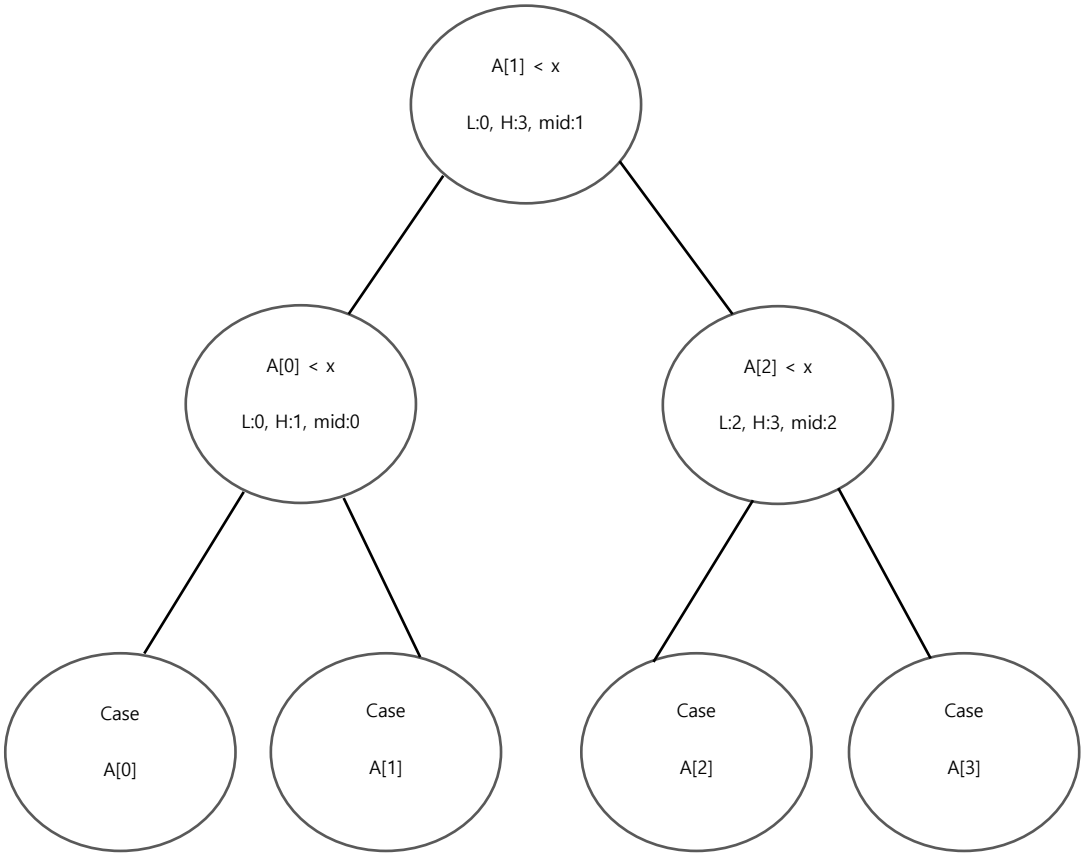
The possible outcome is n + 1. Because when the BinarySearch fails to search of value x, it will return ‘Not found’. If it is found it will return the value between 1 and n. Therefore, different outcome is n+1.

15.

The answer is 2.

One possible outcome is A[i]<x and the other outcome is A[i]>=x. Therefore, the number of different outcomes is 2.

16.



17.

When searching for values, one comparison will divide the whole array into 2 parts. And repeating this process until the value is found. It is like binary search tree and the number of comparisons is related to depth of the tree. So, the height of decision tree provides the number of comparisons.

In a binary search tree, the height of the tree of number of n nodes is $h = \log_2 n$

Therefore, the asymptotic lower bound for searching in a sorted array in comparison model is $\Omega(\log n)$

PART IV

18.

Base case: j=1. The numbers will be sorted on the last 1 digit because sort algorithm used in the problem is stable sort. Stable sort can sort the numbers by its least significant digit. Therefore, for j = 1, it is true that the numbers are sorted.

19.

Inductive step: Assume for j, prove for j+1

Let's assume that there is two number X, Y. And X_j means jth digit of X and Y_j means jth digit of Y.

And then there will be three cases.

Case#1, $X_{j+1} > Y_{j+1}$: Bucket sort will put them in order

Case#2, $X_{j+1} < Y_{j+1}$: Again, bucket sort will put them in order

Case#3, $X_{j+1} = Y_{j+1}$: In this case, order depends on previous digit j. Invariant claims that they are already sorted.

Conclusion: By the principle of induction, radix sort algorithm can sort an array of n d-digit integers,

with each digit in base k.

20. Why does the invariant imply that the radix sort as a whole is correct?

'invariant: after j passes through the loop, the input is sorted according to the integers formed by each element's j least-significant'

According to the invariant, it is sure that after d loops, the numbers will be sorted based on their d least significant digits. If the numbers have d digits each, it will be sorted according to the value of all of their digits. Therefore, the number will be sorted correctly according to their values after d loops. As a result, radix sort as a whole is correct.