Syed Ukkashah

23k-0055

BAIS-A

Design Analysis and Algorithms

ASSIGNMENT 2

Q1)

```
partition (arr, st, end) {
    mid = (st + end)/2
    if (arr[st] > arr[mid]){
        swap(arr[st], arr[mid])
    }
    if (arr[st] > arr[end]){
        swap(arr[st], arr[end])
    }
    if (arr[mid] > arr[end]){
        swap(arr[mid], arr[end])
    }
// the process above ensures that arr[mid] holds
// the median value. This ensures the algorithm doesn't conform to worst case
swap(arr[mid], arr[end])
pivot = arr[end]

idx = st - 1
for (j = st; j < end; j++) {
    if (arr[j] > pivot) {
        idx++
        swap(arr[idx], arr[j])
    }
}
```

```
    swap (arr[i+1], arr[end])
    return i+1
}
```

Worst-case analysis : The pivot is the smallest/largest element

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

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

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

$T(n-3) = T(n-4) + c(n-3)$

Substituting in $T(n)$, we have

$T(N) = T(1) + c(2+3+\ldots\ldots N)$

$T(N) = 1 + c\left(\frac{n(n+1)}{2} - 1\right)$

$\therefore T(n) = O(n^2)$

Best-case analysis → pivot is in the middle

- Takes $O(n)$ time to perform partition
- Divides arrays into two subarrays of $n/2$ length.

So, the recurrence relation becomes.

$T(n) \Rightarrow 2T(n/2) + O(n)$

solving w/ master's theorem

$a=2, b=2, f(n) \in O(n), d=1$

$a = b^d$

so $T(n) \in \Theta(n \log n)$

Average case → avoiding worst case, so we will have

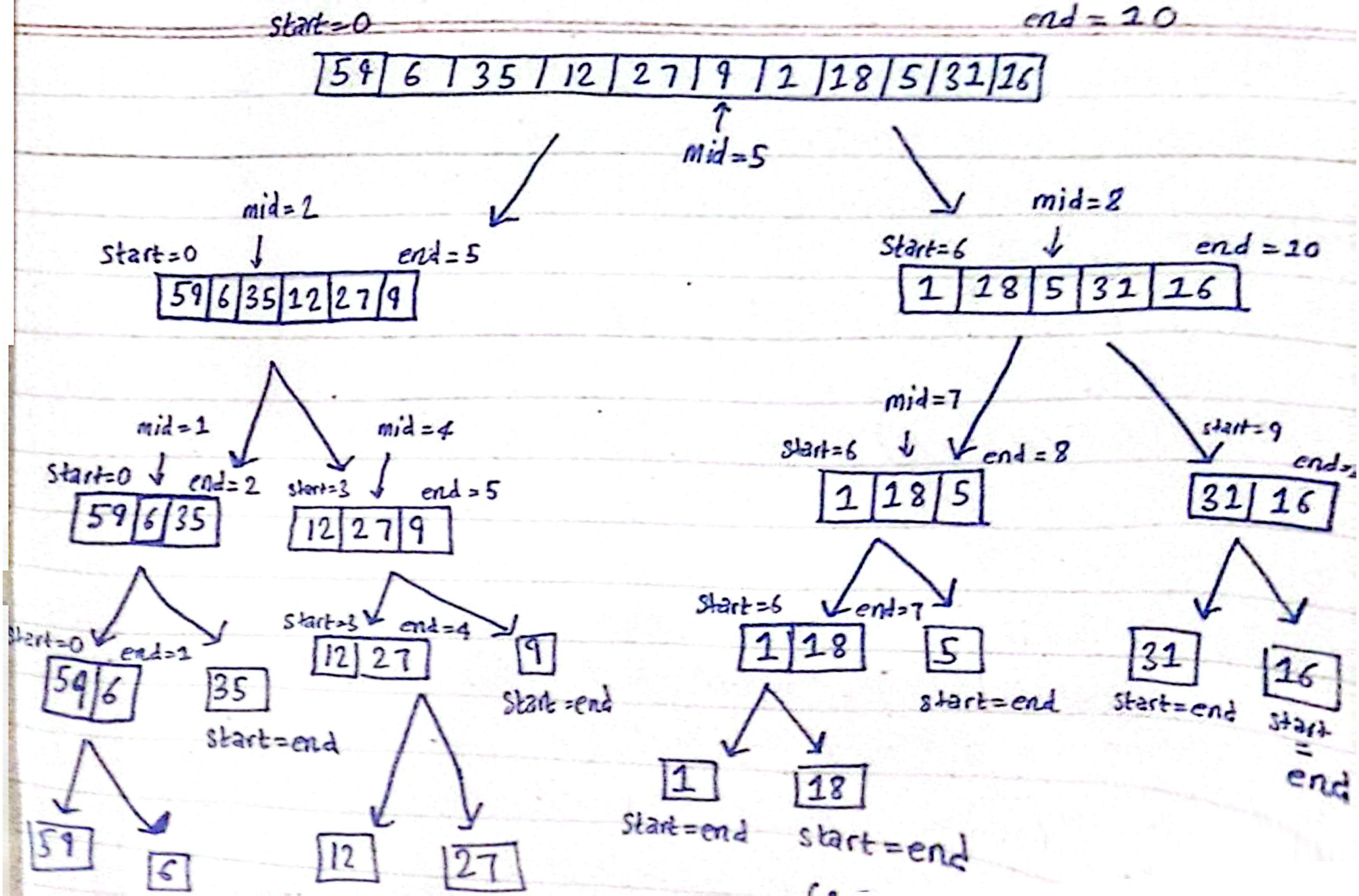
$O(n \log n)$

Q2)

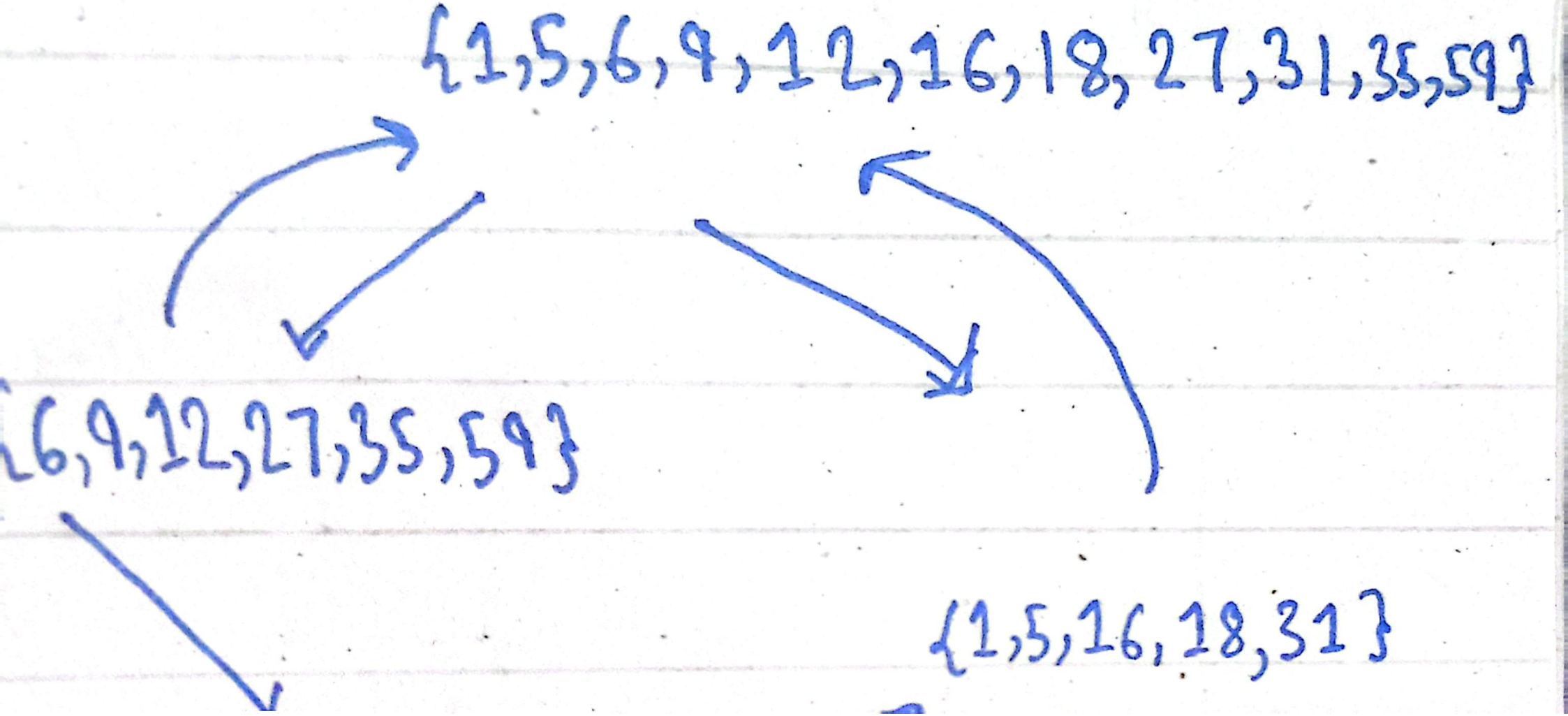
```
merge (arr, start, mid, end) {
    i = start, j = mid + 1, k = 0
    while (i <= mid && j <= end) {
        if (arr[i] <= arr[j]) {
            temp[k] = arr[i]
            k++
            i++
        }
        else {
            temp[k] = arr[j]
            k++
            j++
        }
    }
```

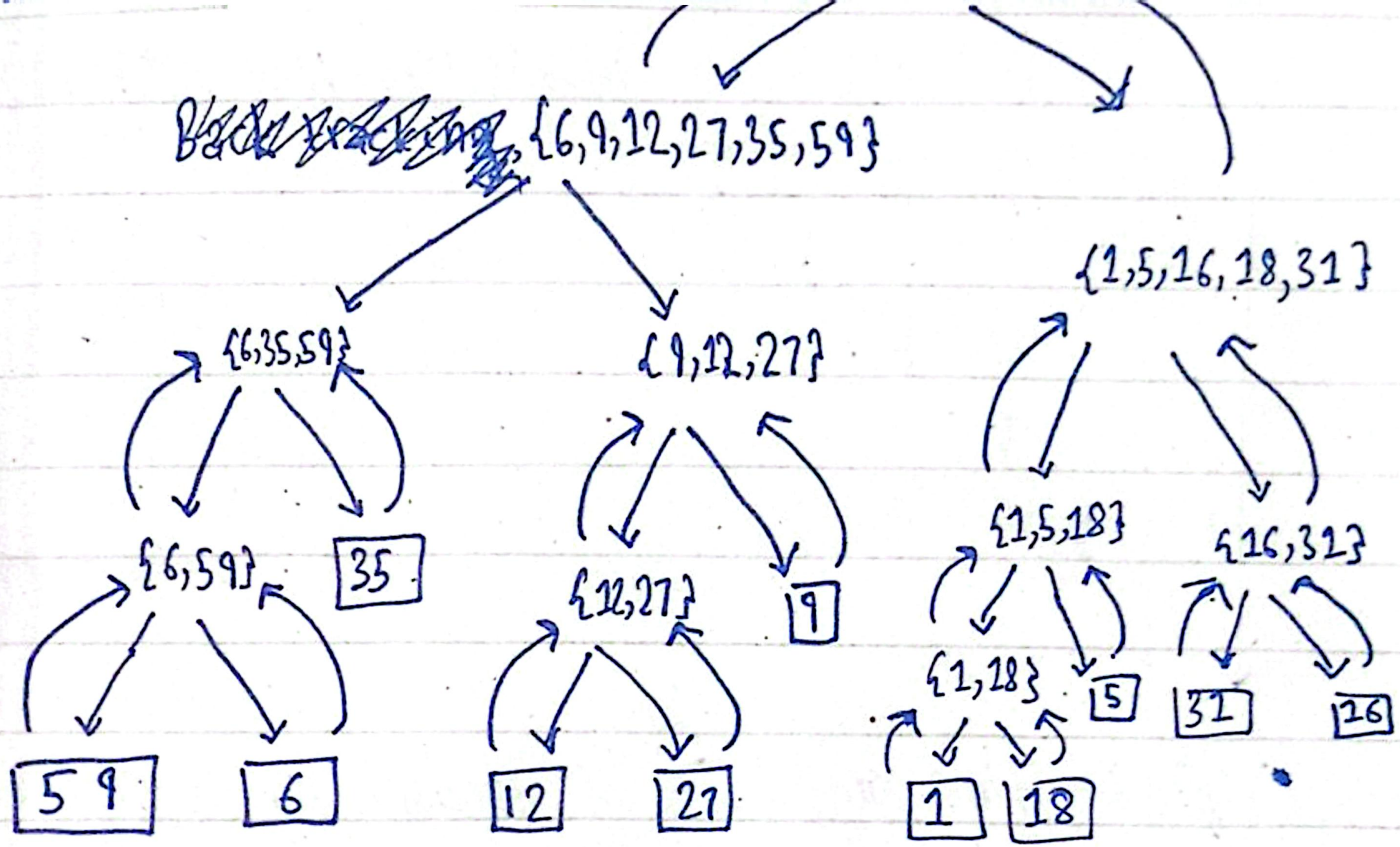
```
    while (i <= mid) {
        temp[k] = arr[i]
        i++
        k++
    }
    while (j <= end) {
        temp[k] = arr[j]
        j++
        k++
    }
}

mergeSort (arr, start, end) {
    if (start < end) {
        mid = (end - start) / 2
        mergeSort (arr, start, mid)
        mergeSort (arr, mid+1, high)
        merge (arr, start, mid, high)
    }
}
```


start=0
end=10
59 6 35 12 27 9 1 18 5 31 16
mid=5
mid=2
start=0
end=5
59 6 35 12 27 9
mid=8
start=6
end=10
1 18 5 31 16
mid=1
start=0
end=2
59 6 35
mid=4
start=3
end=5
12 27 9
mid=7
start=6
end=8
1 18 5
start=9
31 16
start=0
end=1
59 6
35
start=end
start=3
end=4
12 27
9
start=end
start=6
end=7
1 18
5
start=end
31
start=end
16
start=end
59
6
12
27
1
start=end
18
start=end

Backtracking....


{1,5,6,9,12,16,18,27,31,35,59}
{6,9,12,27,35,59}
{1,5,16,18,31}


{6,9,12,27,35,59}
{6,35,59}
{9,12,27}
{1,5,16,18,31}
{6,59}
35
{12,27}
9
{1,5,18}
{16,31}
{1,18}
5
31
26
59
6
12
27
1
18

Dividing the input into 4 parts instead of two, the recurrence relation will be

$T(n) = 4T(n/4) + \theta(n)$

$a = 4, b = 4, f(n) \in O(n^2), d = 1$

$a = b^d$

$\therefore T(n) \in O(n \log n)$

(Q3a) Standard matrix multiplication algorithm

```
matrixMultiply (rowA, colsA, rowB, colsB){
    if (colsA == rowB){
O(n) ←  for(i=0; i<rowA; i++){
O(n) ←      for(j=0; j<colsB; j++){
O(n) ←          for(k=0; k<colsA; k++){
                    result[i][j] += A[i][k] * B[k][j]
                }
            }
        }
    }
```

$O(n^3)$ ← Time complexity of standard matrix product algorithm

(93b) Strassen_Multiply(A, B, n) ← Step 1

1. If n == 1. // Base case: If matrix size is 1x1, multiply the single elements

```
C[0][0] = A[0][0] * B[0][0]
return C
```

2. Divide A & B into 4 submatrices of each size $(n/2 \times n/2)$

2. Divide A & B into 4 submatrices of each size $(n/2 \times n/2)$

A11, A12, A21, A22
B11, B12, B21, B22

3. Compute the 7 products

$M1 = (A11+A22)(B11+B22)$
$M2 = (A21+A22)(B11)$
$M3 = (A11)(B12-B22)$
$M4 = (A22)(B21-B11)$
$M5 = (A11+A12)(B22)$
$M6 = (A21-A11)(B11+B12)$
$M7 = (A12-A22)(B21+B22)$

4. Combine results to form C's submatrices

$C11 = M1+M4-M5+M7$
$C12 = M3+M5$
$C21 = M2+M4$
$C22 = M1-M2+M3+M6$

5. Combine submatrices C11, C12, C21, C22 into one resultant Matrix C

6. return C

Q3c). There are 7 recursive calls on $n/2$ size matrices
. $O(n^2)$ is the cost of additions and subtractions (double loops)

so recurrence relation becomes

$T(n) = 7T(n/2) + O(n^2)$

$a = 7, b = 2, d = 2$

$a < b^d$

so $T(n) \in O(n^{\log_2 7}) \approx O(n^{2.81})$

Q3e) Strassen's algorithm is asymptotically faster $O(n^{2.81})$ compared to the standard matrix multiplication algorithm $O(n^3)$

(Q4) Master's Theorem helps in finding time complexity of divide & conquer algorithms of the form

$$T(n) = aT(n/b) + f(n)$$

- $a \geq 1$ → num of recursive sub problems
- $b > 1$ → factor by which the problem size is divided
- $f(n)$ → cost of work done outside the recursive calls

$T(n)$

Case 1: $a < b^d$ where $T(n) \in \Theta(n^d)$

recursive part dominates here since the non-recursive work grows slower

Case 2: $a = b^d$ where $T(n) \in \Theta(n^d \log n)$

Non-recursive part grows just as fast as the recursive part

Case 3: $a > b^d$, $T(n) \in O(n^{\log_b a})$

Non-recursive part grows faster than the recursive part.

(Q5) $4T(n/4) + O(n)$

$a = 4, b = 4, f(n) \in O(n^d), d = 1$

$a = b^d$

so Case 2 applies here,

$O(n^d \log n)$

$\therefore T(n) \in O(n \log n)$

(Q6) 1. $T(n) = 2T(n/2) + n^2$

$T(n/2) = 2T(n/4) + (\frac{n}{2})^2 = 2T(n/4) + \frac{n^2}{4}$

$T(n/4) = 2T(n/8) + \frac{n^2}{16}$

$T(n/8) = 2T(n/16) + \frac{n^2}{64}$

Substituting, we have

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

$T(n) = 2[2T(n/4) + n^2/4] + n^2 \Rightarrow 4T(n/4) + n^2/2 + n^2 \Rightarrow 4T(n/4) + \frac{3n^2}{2}$

$T(n) \Rightarrow 4[2T(n/8) + \frac{n^2}{16}] + 3n^2/2 \Rightarrow 8T(n/8) + \frac{n^2}{4} + 3n^2/2 \Rightarrow 8T(n/8) + \frac{7n^2}{4}$

$T(n) \Rightarrow 8[2T(n/16) + n^2/64] + 7n^2/4 \Rightarrow 16T(\frac{n}{16}) + \frac{n^2}{8} + \frac{7n^2}{4} = 16T(\frac{n}{16}) + \frac{15n^2}{8}$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + n^2 \sum_{i=0}^{k-1} \left(\frac{1}{2}\right)^i$$

$$\frac{n}{2^k} = 1$$

$$2^k = n$$

$$k = \log_2 n$$

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

geometric series sum $\Rightarrow \frac{a(1-r^n)}{1-r}$

$$n + n^2\left(\frac{1-(\frac{1}{2})^k}{1-1/2}\right)$$

$$n + n^2\left(\frac{1-(1/2)^k}{1/2}\right)$$

$$n + 2n^2\left(1 - \frac{1}{2^k}\right)$$

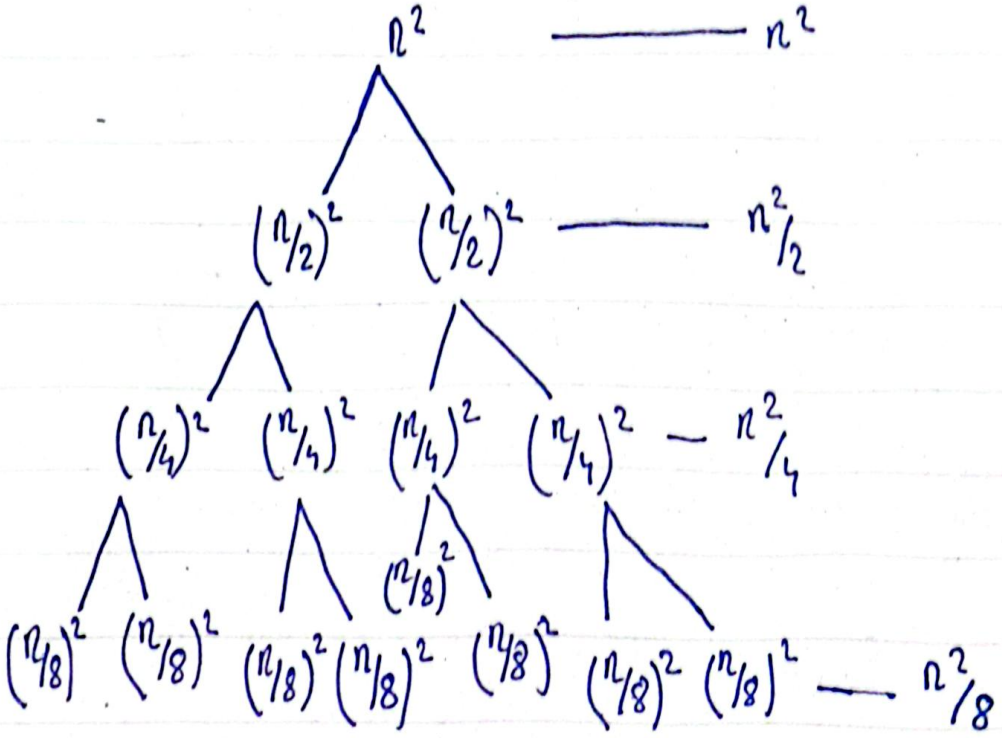
$$n + 2n^2\left(1 - \frac{1}{2^{\log_2 n}}\right)$$

$$n + 2n^2\left(1 - \frac{1}{n}\right)$$

$$n + 2n^2 - 2n$$

$$O(n) + 2n^2 - 2n$$

$$T(n) \Rightarrow O(n^2)$$



Total cost $\Rightarrow n^2\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots\ldots \frac{1}{n}\right)$

geometeic seeies

$\therefore T(n) \in O(n^2)$

11. $T(n) = 3T(n/3) + n\log^2 n$

$T(n/3) \Rightarrow 3T(n/9) + n/3 \log^2 n/3$

$T(n/9) \Rightarrow 3T(n/27) + n/9 \log^2 n/9$

$T(n/27) \Rightarrow 3T(n/81) + n/27 \log^2 n/27$

Substituting, we have

$T(n) \Rightarrow 3[3T(n/9) + n/3 \log^2 n/3] + n\log^2 n$

$\Rightarrow 3^2 T(n/9) + n\log^2 n/3 + n\log^2 n$

$T(n) \Rightarrow 3^2[3T(n/27) + n/9 \log^2 n/9] + n\log^2 n/3 + n\log^2 n$

$\Rightarrow 3^3 T(n/27) + n\log^2 n/9 + n\log^2 n/3 + n\log^2 n$

$T(n) \Rightarrow 3^3[3T(n/81) + n/27 \log^2 n/27] + n\log^2 \frac{n}{9} + n\log^2 \frac{n}{3} + n\log^2 n$

$\Rightarrow 3^4 T(\frac{n}{81}) + n\log^2 \frac{n}{27} + n\log^2 \frac{n}{9} + n\log^2 \frac{n}{3} + n\log^2 n$

$T(n) \Rightarrow 3^k T(\frac{n}{3^k}) + n(\log^2 \frac{n}{3^{k-1}} + \log^2 \frac{n}{3^{k-2}} + \log^2 \frac{n}{3^{k-3}} + \cdots \log^2 n)$

~~$T(n) = 3^k T(\frac{n}{3^k}) + n(\log^2 \frac{n}{\sum_{i=0}^{k-1} 3^i})$~~

~~$T(n) = 3^k T(\frac{n}{3^k}) + n\{\log^2 n - \log^2 3^i\}$~~

$\sum_{i=0}^{k-1}$

$T(n) = 3^k T(\frac{n}{3^k}) + n(\log^2 n - \sum_{i=0}^{k-1} i \log_2 3)$

$\sum_{i=0}^{k-1} i \Rightarrow \frac{k(k-1)}{2}$

$$\frac{n}{3^k} = 1$$

$$3^k = n$$

$$k = \log_3 n$$

$$\therefore \frac{\log_3 n(\log_3 k - 1)}{2} \Rightarrow \frac{(\log_3 n)^2 - \log_3 n}{2}$$

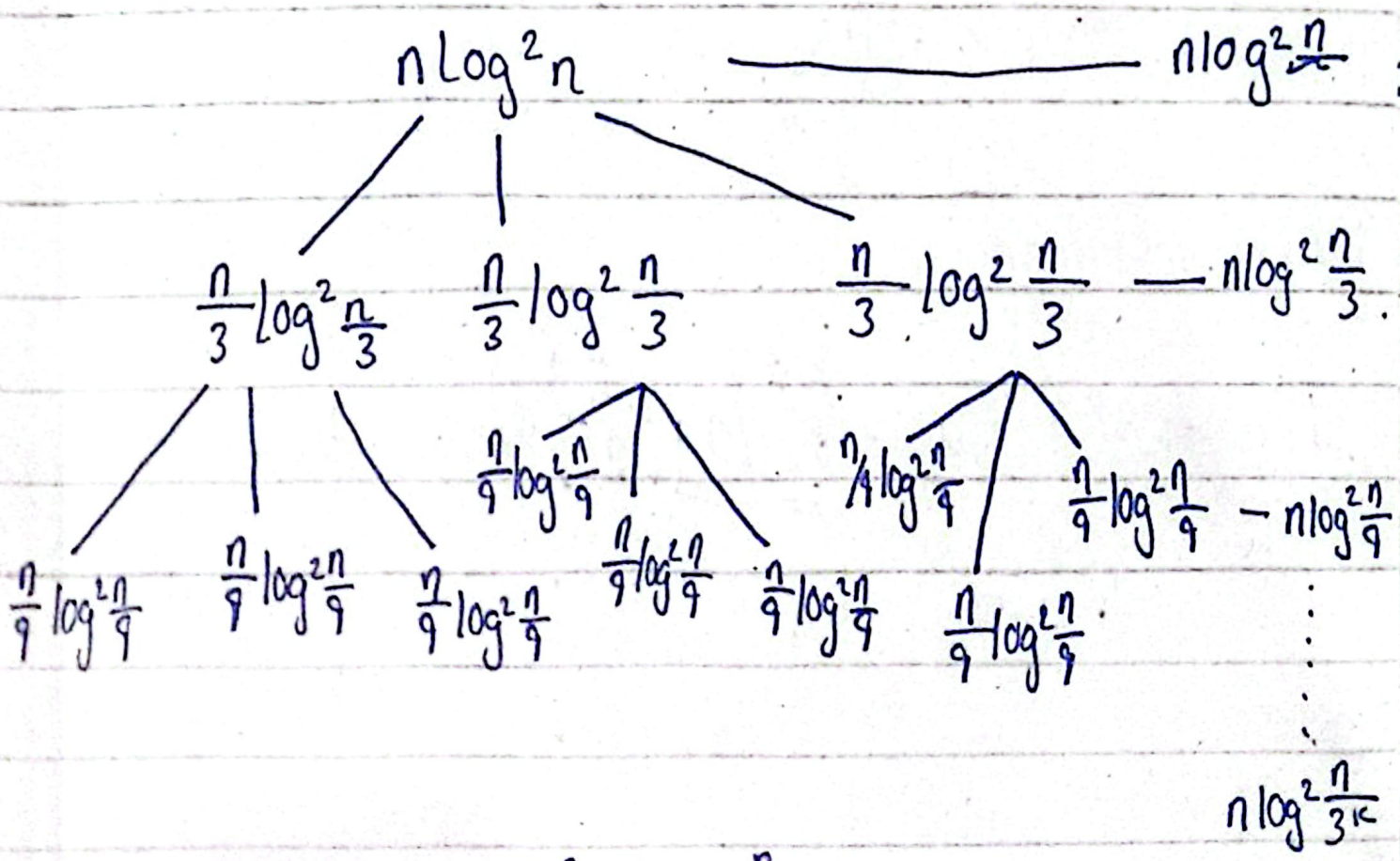
$$\log_3 n = \frac{\log_2 n}{\log_2 3}$$

$$= \frac{\frac{\log_2 k}{\log_2 3}\left(\frac{\log_2 n}{\log_2 3} - 1\right)}{2}$$

$$T(n) = 3^{\log_3 n} T(1) + n\left(\frac{(\log n)^2 - \log n}{2}\right)$$

$$T(n) \Rightarrow n + \frac{n(\log^2 n - \log n)}{2}$$

$$\therefore T(n) \in n\log^2 n$$



Total cost : $n\log^2 n + n\log^2\frac{n}{3} + n\log^2\frac{n}{9} + \ldots\ldots$

: $n(\log^2\frac{n}{3^k})$

$T(n) \in O(n\log^2 n)$

iii $T(n) = 2T(\frac{n}{2}) + \log n$

$T(\frac{n}{2}) = 2T(\frac{n}{4}) + \log \frac{n}{2}$

$T(\frac{n}{4}) = 2T(\frac{n}{8}) + \log \frac{n}{4}$

$T(\frac{n}{8}) = 2T(\frac{n}{16}) + \log \frac{n}{8}$

Substituting back, we have

$T(n) = 2[2T(\frac{n}{4}) + \log \frac{n}{2}] + \log n = 2^2 T(\frac{n}{4}) + 2(\log n - 1) + \log n$

$T(n) = 2^2[2T(\frac{n}{8}) + \log \frac{n}{4}] + 2(\log n - 1) + \log n = 2^3 T(\frac{n}{8}) + 2^2(\log n - 2) + 2(\log n - 1)$

$T(n) = 2^3[2T(\frac{n}{16}) + \log \frac{n}{8}] + 2^2(\log n - 2) + 2(\log n - 1)$

$\Rightarrow 2^4 T(\frac{n}{16}) + 2^3[\log n - 3] + 2^2(\log n - 2) + 2(\log n - 1)$

$$T(n) = 2^k T(\frac{n}{2^k}) + \sum_{i=0}^{k-1} 2^i (\log n - i) = 2^k T(\frac{n}{2^k}) + \log n \sum_{i=0}^{k-1} 2^i - \sum_{i=0}^{k-1} i \cdot 2^i$$

$$\Rightarrow \cancel{T(\frac{n}{2^k}) + \sum_{i=0}^{k-1} 2^i \cdot \sum_{i=0}^{k-1}}$$

1st sum: $\sum_{i=0}^{k-1} 2^i \Rightarrow 2^k - 1$ (Geometric series)

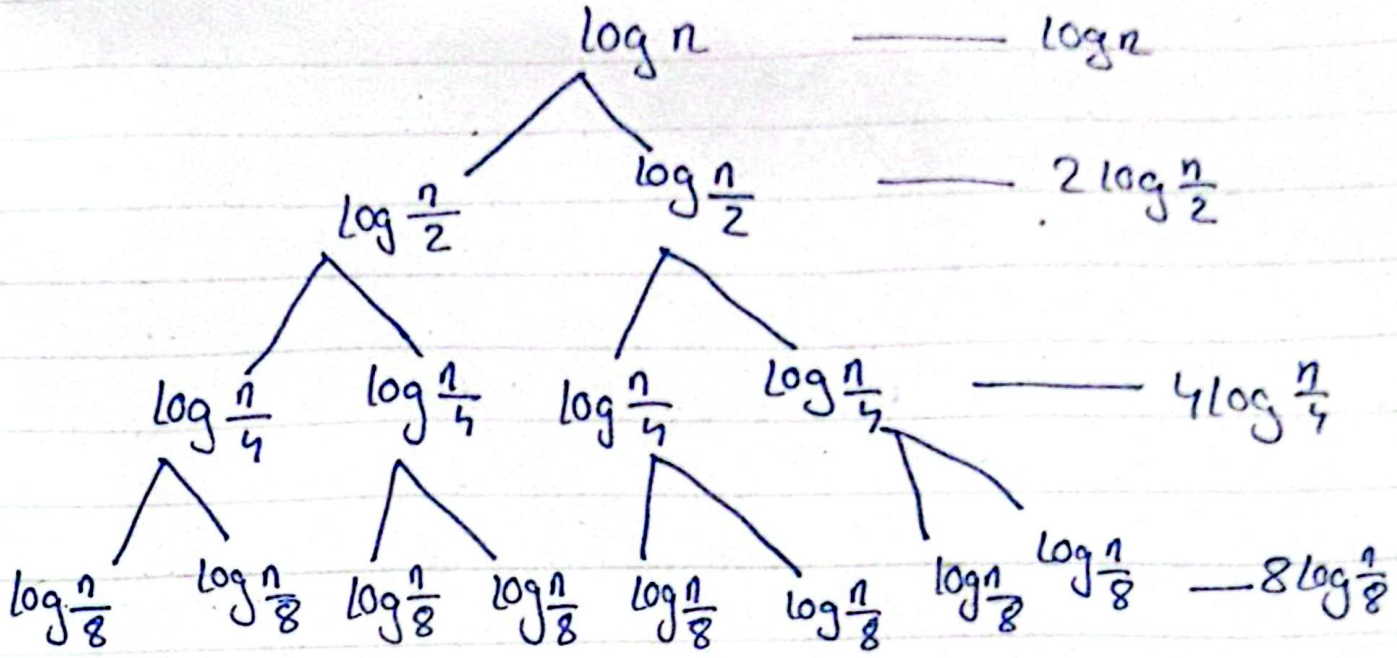
so $\log n \sum_{i=0}^{k-1} 2^i \Rightarrow \log n (2^k - 1)$

2nd sum: since $i=0$ term is 0, $\sum_{i=1}^{k-1} 2^i \cdot i$

$\frac{n}{2^k} = 1$, $k = \log_2 n$

$T(n) = 2n - \log n - 2$

$T(n) \in O(n)$



Cost: $T(n) = \log n + 2\log\frac{n}{2} + 4\log\frac{n}{4} + 8\log\frac{n}{8} + \dots$

$\Rightarrow \log n + 2(\log n - \log 2) + 4(\log n - \log 4) + 8(\log n - \log 8) + \dots$

$\Rightarrow \log n + 2(\log n - 1) + 4(\log n - 2) + 8(\log n - 3) + \dots$

$T(n) = O(n)$

Q7) $T(n) = 4T(n/2) + n^2$

$a = 4, b = 2, f(n) = n^2, d = 2$

$a = b^d$

$\therefore T(n) \in \Theta(n^d \log n)$

$T(n) \in \Theta(n^2 \log n)$

11. $T(n) = 2T(n/4) + \sqrt{n}$

$a = 2, b = 4, d = 1/2$

$a = b^d$

$\therefore T(n) \in \Theta(n^d \log n)$

$T(n) \in \Theta(\sqrt{n} \log n)$

11. $T(n) = 4T(n/2) + n^2 \log n$

$a = 4, b = 2, f(n) = n^2 \log n$

$f(n) = \Theta(n^{\log_b a} \log^k n)$

$\Rightarrow \Theta(n^{\log_2 4} \log^1 n)$

$\therefore T(n) \in \Theta(n^{\log_b a} \log^{k+1} n)$

$T(n) \in \Theta(n^{\log_2 4} \log^2 n) \Rightarrow T(n) \in \Theta(n^2 \log^2 n)$

(Q8) I. $T(n) = 4T(n/2) + n^2$

Guess 1: $T(n) = O(n)$

$T(n) \leq c \cdot g(n)$

$T(n) = 4T(n/2) + n^2 \leq 4 \cdot c(n/2) + n^2$

$4T(n/2) + n^2 \leq n^2 + 2cn$

for $T(n) \leq cn$:

$2cn + n^2 \leq cn$

$n^2 + nc \leq 0$

Guess 1 is false

Guess 2: $T(n) = O(n^2)$

$T(n) \leq c \cdot g(n)$

$T(n) = 4T(n/2) + n^2 \leq 4c(n/2)^2 + n^2$

$4T(n/2) + n^2 \leq cn^2 + n^2$

$n^2 + cn^2 \leq cn^2$

$n^2 \leq 0$

Guess 2 is also false