

$$a) T(n) = 4T(n/4) + 5n$$

$$a=4 \quad b=4 \quad c=1 \quad d=1$$

$$n^1 < 5n \rightarrow \boxed{5n}$$

$$n = n \rightarrow \boxed{n \log n}$$

$$b) T(n) = 4T(n/5) + 5n$$

$$a=4 \quad b=5 \quad c=\log_5 4 \quad d=1$$

$$n^{\log_5 4} < n \rightarrow n$$

$$c) T(n) = 5T(n/4) + 4n$$

$$a=5 \quad b=4 \quad c=\log_4 5 \quad d=1$$

$$n^{\log_4 5} > n \rightarrow \boxed{n^c}$$

$$d) T(n) = 25T(n/5) + n^2$$

$$a=25 \quad b=5 \quad c=\log_5 25 = 2 \quad d=2$$

$$c=d \rightarrow \boxed{n^2 \log n}$$

$$e) T(n) = 4T(n/5) + \log n$$

$$a = 4 \quad b = 5 \quad c = \log_5 4$$

$$n^{\log_5 4} > \log n \rightarrow \boxed{n^c}$$

$$f) T(n) = 4T(n/5) + \log n \sqrt{n}$$

$$a = 4 \quad b = 5 \quad c = \log_5 4$$

$$n^{\log_5 4} < \log n \sqrt{n} \rightarrow \boxed{n^{\log_5 4} \cdot \log n \sqrt{n}}$$

$$g) T(n) = 4T(\sqrt{n}) + \log n$$

$$n = 2^m \rightarrow \textcircled{1}$$

$$T(2^m) = 4T(2^{m/2}) + m$$

$$T(2^m) = S(m) \rightarrow \textcircled{2}$$

$$S(m) = 4S(m/2) + m$$

$$a = 4 \quad b = 2 \quad c = \log_2 4 = 2$$

$$m^2 < m \rightarrow m \rightarrow \boxed{(\log n)^2}$$

$$H) T(n) = 4T(\sqrt{n}) + \log^2 n$$

$$n = 2^m \rightarrow \textcircled{1}$$

$$T(2^m) = 4T(2^{m/2}) + m^2$$

$$T(2^n) = S(m) \rightarrow \textcircled{2}$$

$$S(m) = 4S(m/2) + m^2$$

$$a = 4 \quad b = 2 \quad c = \log_2 4 = 2$$

$$m^2 \rightarrow m^2 \rightarrow m^2 \log m \rightarrow \boxed{(\log n)^2 \log(\log n)}$$

$$i) T(n) = T(\sqrt{n}) + 5$$

$$n = 2^m \rightarrow \textcircled{1}$$

$$T(2^m) = T(2^{m/2}) + 5$$

$$T(2^m) = S(m) \rightarrow \textcircled{2}$$

$$S(m) = S(m/2) + 5$$

$$a = 1 \quad b = 2 \quad c = \log_2 1 = 0$$

$$m^0 = 1 \rightarrow \log m \rightarrow \log(\log n)$$

Task 2: Code Analysis [40 Marks]

For each of the code snippets provided below, perform a detailed analysis. Specifically identify the key operations and write the time complexity using Big O notation.

1.

```
int secondLargest(int arr[], int n) {  
    int largest = INT_MIN;  
    secondLargest = INT_MIN;  
    for (int i = 0; i < n; i++) {  
        if (arr[i] > largest) {  
            secondLargest = largest;  
            largest = arr[i];  
        } else if (arr[i] >  
secondLargest && arr[i] != largest) {  
            secondLargest = arr[i];  
        }  
    }  
    return secondLargest;  
}
```

$2n+1$

$O(n)$

2.

```
int findMissingNumber(int arr[], int n)  
{  
    int totalSum = n * (n + 1) / 2;  
    int arrSum = 0;  
    for (int i = 0; i < n - 1; i++) {  
        arrSum += arr[i];  
    }  
    return totalSum - arrSum;  
}
```

n

$2n+2$

$O(n)$

3.

```
void dijkstra(int graph[V][V], int src)
{
    int dist[V];
    bool visited[V] = { false };

    for (int i = 0; i < V; i++)
        dist[i] = 99999;
    dist[src] = 0;

    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, visited);
        visited[u] = true;

        for (int v = 0; v < V; v++) {
            if (!visited[v] && graph[u][v] && dist[u] != 99999 && dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }
}
```

$$1$$
$$3n+2$$

$$2n+2$$

$$n$$

$$O(n)$$

4.

```
void multiplyMatrices(int A[3][3], int B[3][3], int C[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            C[i][j] = 0;
            for (int k = 0; k < 3; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}
```

$$2V$$

$$O(1)$$

5.

```
int countSetBits(int n) {
    int count = 0;
    while (n > 0) {
        count += n & 1;
        n >>= 1;
    }
    return count;
}
```

Handwritten analysis for problem 5:

- 1 (for the initial `count = 0`)
- $\log(n)$ (for the `while` loop)
- $\log(n)$ (for the `count += n & 1;` operation)
- $\log(n)$ (for the `n >>= 1;` operation)
- 1 (for the `return count;` statement)

$O(\log(n))$

6.

```
int lcs(string X, string Y, int m, int n) {
    if (m == 0 || n == 0) return 0;
    if (X[m - 1] == Y[n - 1]) return 1 + lcs(X, Y, m - 1, n - 1);
    else return max(lcs(X, Y, m - 1, n), lcs(X, Y, m, n - 1));
}
```

Handwritten analysis for problem 6:

- 1 (for the base case `if (m == 0 || n == 0)`)
- n (for the recursive call `lcs(X, Y, m - 1, n - 1)`)
- 2^n (for the recursive call `lcs(X, Y, m, n - 1)`)

7.

```
#define ll long long
#define vl vector<ll>
ll n, j;
vl weights;
ll knapsack(ll cap, ll i, vl &selected)
{
    if (i == 0 || cap == 0)
        return 0;
    if (weights[i - 1] > cap)
        return knapsack(cap, i - 1, selected);
    vl picked = selected;
    ll pick = knapsack(cap - weights[i - 1], i - 1, picked) + weights[i - 1];
    ll leave = knapsack(cap, i - 1, selected);
    if (pick > leave)
    {
        picked.push_back(weights[i - 1]);
        selected = picked;
        return pick;
    }
    return leave;
}
```

n
 2^n

8.

```
void permute(string s, int l, int r) {
    if (l == r) {
        cout << s << endl;
        return;
    }
    for (int i = l; i <= r; i++) {
        swap(s[l], s[i]);
        permute(s, l + 1, r);
        swap(s[l], s[i]);
    }
}
```

$2n + 1$
 $n! \Rightarrow \text{factorial}$

9.

```

int ternarySearch(int arr[], int l, int r, int x) {
    if (r >= l) {
        int mid1 = l + (r - l) / 3;
        int mid2 = r - (r - l) / 3;

        if (arr[mid1] == x) return mid1;
        if (arr[mid2] == x) return mid2;

        if (x < arr[mid1])
            return ternarySearch(arr, l, mid1 - 1, x);
        else if (x > arr[mid2])
            return ternarySearch(arr, mid2 + 1, r, x);
        else
            return ternarySearch(arr, mid1 + 1, mid2 - 1, x);
    }

    return -1;
}

```

$\log_3 n$

10.

```

int max(int a, int b) {
    return (a > b) ? a : b;
}

int max(int a, int b, int c) {
    return max(max(a, b), c);
}

int maxCrossingSum(int arr[], int l, int m, int h) {
    int sum = 0;
    int left_sum = INT_MIN;
    for (int i = m; i >= l; i--) {
        sum = sum + arr[i];
        if (sum > left_sum)
            left_sum = sum;
    }
    sum = 0;
    int right_sum = INT_MIN;
    for (int i = m; i <= h; i++) {
        sum = sum + arr[i];
        if (sum > right_sum)
            right_sum = sum;
    }
    return max(left_sum + right_sum - arr[m], left_sum, right_sum);
}

int maxSubArraySum(int arr[], int l, int h) {
    if (l > h)
        return INT_MIN;
    if (l == h)
        return arr[l];
    int m = (l + h) / 2;
    return max(maxSubArraySum(arr, l, m - 1), maxSubArraySum(arr, m + 1, h),
               maxCrossingSum(arr, l, m, h));
}

```

$n \log n$

11.

```
struct suffix {
    int index;
    string suffix;
};

bool compareSuffix(suffix a, suffix b) {
    return a.suffix < b.suffix;
}

void buildSuffixArray(string s) {
    suffix suffixes[s.length()];
    for (int i = 0; i < s.length(); i++) {
        suffixes[i].index = i;
        suffixes[i].suffix = s.substr(i);
    }
    sort(suffixes, suffixes + s.length(), compareSuffix);
    for (int i = 0; i < s.length(); i++) {
        cout << suffixes[i].index << " ";
    }
}
```

$n \log n$

12.

```
void bellmanFord(int graph[][3], int V, int E, int src) {
    int dist[V];
    for (int i = 0; i < V; i++) dist[i] = INT_MAX;
    dist[src] = 0;

    for (int i = 1; i <= V - 1; i++) {
        for (int j = 0; j < E; j++) {
            int u = graph[j][0], v = graph[j][1], weight = graph[j][2];
            if (dist[u] != INT_MAX && dist[u] + weight < dist[v]) {
                dist[v] = dist[u] + weight;
            }
        }
    }

    for (int j = 0; j < E; j++) {
        int u = graph[j][0], v = graph[j][1], weight = graph[j][2];
        if (dist[u] != INT_MAX && dist[u] + weight < dist[v]) {
            cout << "Negative weight cycle detected!";
            return;
        }
    }

    for (int i = 0; i < V; i++) cout << i << " " << dist[i] << endl;
}
```

n^2

n

13.

```
void countFrequency(int arr[], int n) {  
    cout << "Element Frequency" << endl;  
    for (int i = 0; i < n; i++) {  
        int count = 1;  
        for (int j = i + 1; j < n; j++) {  
            if (arr[i] == arr[j]) {  
                count++;  
                j++;  
            }  
        }  
        cout << arr[i] << " " << count << endl;  
    }  
}
```

Handwritten red annotations for complexity analysis:

- A vertical line with an arrow pointing to the first `for` loop, labeled n .
- A bracket spanning the inner `for` loop, labeled n^2 .
- A bracket spanning the `cout` statement, labeled n .

14.

```
bool isPalindrome(int arr[], int size) {  
    for (int i = 0; i < size / 2; i++) {  
        if (arr[i] != arr[size - i - 1]) {  
            return false;  
        }  
    }  
    return true;  
}
```

Handwritten red annotations for complexity analysis:

- A bracket spanning the `for` loop, labeled $n/2$.
- A vertical line with an arrow pointing to the `return false;` statement, labeled 1 .

15.

```
void findDuplicates(int arr[], int n) {  
    cout << "Duplicates: ";  
    for (int i = 0; i < n; i++) {  
        for (int j = i + 1; j < n; j++) {  
            if (arr[i] == arr[j]) {  
                cout << arr[i] << " ";  
                break;  
            }  
        }  
    }  
    cout << endl;  
}
```

Handwritten annotations for complexity analysis:

- A red line under the first parameter `arr[]` is labeled with a red 1 .
- A red line under the loop condition `i < n` is labeled with a red n .
- A red line under the loop condition `j < n` is labeled with a red n^2 .
- A red line under the `if` condition `arr[i] == arr[j]` is labeled with a red n^2 .
- A red line under the `cout` statement is labeled with a red 1 .
- A red line under the `break;` statement is labeled with a red 1 .
- A red line under the `cout << endl;` statement is labeled with a red 1 .

16-

```
int getsum(int n, int m){  
    int sum = 0;  
    for(int i=0; i<n; i++){  
        for(int j=m; j>=1; j/=2){  
            sum += i+j;  
        }  
    }  
    return sum;  
}
```

Handwritten annotations for complexity analysis:

- A red line under the first parameter `n` is labeled with a red 1 .
- A red line under the loop condition `i < n` is labeled with a red n .
- A red line under the loop condition `j >= 1` is labeled with a red $\log n$.
- A red line under the `sum += i+j;` statement is labeled with a red 1 .

17-

```
int count(int arr[],int n){
    const int fixed[] = {15,90,48,64,5};
    const int k = 5;
    int count = 0;
    for(int i =0;i<n;i++){
        for(int j=0;j<k;j++){
            if(arr[i]==fixed[j]){
                count++;
                break;
            }
        }
    }
    return count;
}
```

Handwritten annotations for problem 17:

 - Red lines under `fixed[]` and `k = 5` with a bracket and 5 written next to it.
 - Red line under `count = 0`.
 - Red line under the first `for` loop with n written next to it.
 - Red line under the second `for` loop with $5n$ written next to it.
 - Red line under `count++` with 1 written next to it.
 - Red line under `return count;` with 1 written next to it.

18-

```
void findPairsWithSum(int arr[], int n, int sum) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] + arr[j] == sum) {
                cout << "Pair with sum " << sum << " is: (" << arr[i] << ", " << arr[j] << ")" << endl;
            }
        }
    }
}
```

Handwritten annotations for problem 18:

 - Red line under `for (int i = 0; i < n; i++)` with n written next to it.
 - Red line under `for (int j = i + 1; j < n; j++)` with n^2 written next to it.
 - Red line under `cout << "Pair with sum ..."` with n^2 written next to it.

19.

```
long long power(long long base, int exp){  
    if(exp == 1) return base;  
    if(exp == 0) return 1;  
    else{  
        long long temp = base * base;  
        long long ans = power(temp, exp/2);  
        if(exp % 2 == 1) return base * ans;  
        return ans;  
    }  
}
```

Handwritten red annotations: A vertical line next to the first two return statements, and a line pointing to the recursive call with the text "log n".

20-

```
void prefix_sum(int arr[], int prefix[], int n) {  
    prefix[0] = arr[0];  
    for (int i = 1; i < n; i++) {  
        prefix[i] = prefix[i - 1] + arr[i];  
    }  
}
```

Handwritten red annotations: A checkmark next to the for loop, and an arrow pointing to the assignment statement inside the loop.