Mid-Term Exam Solutions - Analysis of Algorithms

Page 1 Solutions (Second Image Provided First)

1. Worst-case Time Complexities

Selection Sort: $O(n^2)$

Merge Sort: $O(n \log n)$

Quick Sort: $O(n^2)$ (when pivot selection is poor)

Insertion Sort: $O(n^2)$

Counting Sort: $O(n + k)$ (where k is the range of numbers)

Radix Sort: $O(nk)$ (where k is the number of digits in the largest number)

Binary Search: $O(\log n)$

2. Useful Criterion for Algorithm Efficiency

Efficiency is based on both time and memory usage.

Thus, the correct answer is: (c) Both Time and Memory.

3. Measuring Time Complexity

The best way to measure time complexity is by counting the number of primitive operations in an algorithm.

Correct answer: (b) Counting primitive operations.

4. Stable Sorting Algorithm

A sorting algorithm is stable if it preserves the relative order of equal elements.

- Counting Sort is stable.
- Selection Sort and Heap Sort are not stable.

Correct answer: (c) Counting Sort.

5. Time Complexity of Nested Loop Algorithm

```
int a = 0;
for(int i = 0; i < N; i++)
    for(int j = 1; j < N; j++)
        a = a + 1;
```

- The outer loop runs N times.
- The inner loop runs N times for each outer iteration.
- Total iterations = $O(N * N) = O(N^2)$.

Correct answer: $O(N^2)$.

6. Time Complexity of Logarithmic Algorithm

```
int i, j, k = 0;
for(i = 1; i < n; i *= 2)
    for(j = 1; j < n; j++)
        k++;
```

- The outer loop runs O(log n) times (as `i` doubles each time).
- The inner loop runs O(n) times.
- Total time complexity = O(n log n).

Correct answer: O(n log n).

7. Time Complexity of While Loop Algorithm

```
int a = 0, i = 1;
while(i < N) {
    i *= 2;
    a += 2;
}
```

- Since `i` doubles in each iteration, the loop runs O(log N) times.

Correct answer: O(log N).

Page 2 Solutions (First Image Provided Second)

8. Time Complexity of Given Loop

```
int k = 2;
for(int i = 2; i <= n; i *= k) {
    // some operation
}
```

- The loop starts at `i = 2` and multiplies by `k` each iteration.
- The number of times the loop runs is log_k(n).

Correct answer: O(log_k n).

9. Time & Space Complexity of Given Code

```
int a = 0, b = 0;
for(int i = 0; i < N; i++) a = a + rand();
for(int j = 0; j < M; j++) b = b + rand();
```

- The first loop runs N times -> O(N).
- The second loop runs M times -> O(M).
- Space complexity is O(1) (constant variables).

Correct answer: O(n + m) time, O(1) space.

10. Binary Search Complexity Analysis

Binary Search follows the recurrence relation:

$T(n) = T(n/2) + O(1)$

Using Master Theorem:

$T(n) = O(\log n)$

11. Solving Recurrence Relations

a) $T(n) = 2T(n/2) + n \rightarrow O(n \log n)$

b) $T(n) = 2T(n/2) + n \log n \rightarrow O(n \log^2 n)$

c) $T(n) = 2T(n/2) + n^2 \rightarrow O(n^2)$

d) $T(n) = 4T(n/2) + n \log n \rightarrow O(n^2)$

12. Best Sorting Algorithm for Given List

Array: 127, 324, 173, 4, 38, 217, 134

- QuickSort is the best choice for average-case O(n log n).
- MergeSort is also O(n log n) but requires additional space.

Thus, the best sorting algorithm for this case is QuickSort with O(n log n) complexity.