Ambedkar College of Arts and Science, Wandoor

Class Test 1 — Answer Key

Data Structures and Algorithms — MAT5EJ302(1)

B.Sc. Mathematics Honours (CUFYUGP 2024) — Semester V

Maximum Marks: 20 **Time: 20 Minutes**

Part A — Short Answer ($5 \times 2 = 10$ marks)

1. Define an algorithm and state its two central questions. **[2]**

Answer: An algorithm is a finite, step-by-step procedure that solves a well-defined computational problem: it takes an input and produces an output. The two central questions asked of any algorithm are:

- **Correctness** — does it always produce the right answer?
- **Efficiency** — how fast does it run, and how much memory does it use?

2. Fibonacci recurrence and naive time complexity. **[2]**

Answer:

$$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2} \ (n \geq 2).$$

The naive recursive algorithm runs in $\mathcal{O}(\varphi^n) \approx \mathcal{O}(1.618^n)$ time — **exponential** in n, because it recomputes the same subproblems (e.g. F_{n-2}) many times over.

3. Formal definition of Big-O. **[2]**

Answer: $f(n) = \mathcal{O}(g(n))$ if there exist positive constants c and n_0 such that

$$f(n) \leq c \cdot g(n) \quad \text{for all } n \geq n_0.$$

Intuitively, $\mathcal{O}(g(n))$ is an upper bound on the growth rate of $f(n)$, ignoring constant factors.

4. Modular arithmetic computations. **[2]**

Answer:

- (a) $38 = 5 \times 7 + 3 \ \Rightarrow \ 38 \bmod 7 = 3$.
- (b) $-15 = (-2) \times 8 + 1 \ \Rightarrow \ -15 \bmod 8 = 1$.

5. Modular multiplication using the distributive property. **[2]**

Answer: $47 \bmod 11 = 3$ (since $47 = 4 \times 11 + 3$) and $53 \bmod 11 = 9$ (since $53 = 4 \times 11 + 9$).

$$47 \times 53 \bmod 11 = (3 \times 9) \bmod 11 = 27 \bmod 11 = 5.$$

Part B — Medium Answer ($2 \times 5 = 10$ marks)

6. Trace iterative Fibonacci for F_7; state complexity. **[5]**

Answer: Using `fib3`: initialise $a = 0, b = 1$, then for $i = 2$ to 7 set $c = a + b$, $a = b$, $b = c$.

i	a	b	c
1	0	1	—
2	1	1	1
3	1	2	2
4	2	3	3
5	3	5	5
6	5	8	8
7	8	13	13

$F_7 = 13$. Time complexity: $\mathcal{O}(n)$ (one loop iteration per step, $n = 7$). Space complexity: $\mathcal{O}(1)$ (only the variables a, b, c are kept, regardless of n).

7. Big-O complexity of the nested loop with inner bound i to n. **[5]**

Given code:

```
for i = 1 to n:
  for j = i to n:
    print(i + j)
```

Answer: For a fixed i, the inner loop runs for $j = i, i+1, \ldots, n$, i.e. $n - i + 1$ times. Summing over all i from 1 to n:

$$\sum_{i=1}^{n}(n - i + 1) = n + (n-1) + \cdots + 1 = \frac{n(n+1)}{2}.$$

Since $\frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$, dropping the constant factor and the lower-order term gives

$$\mathcal{O}(n^2).$$

— End of Solutions —