Subject Name Solutions

4331601 – Winter 2023

Semester 1 Study Material

Detailed Solutions and Explanations

Question 1(a) [3 marks]

Define best case, worst case and average case for time complexity.

Solution

Table 1: Time Complexity Cases

Case Type	Definition	Example
Best Case	Minimum time needed for algorithm execution	Linear search finds element at first position
Worst Case	Maximum time needed for algorithm execution	Linear search finds element at last position
Average Case	Expected time for typical input scenarios	Linear search finds element in middle

- **Best Case**: Algorithm performs optimally with ideal input conditions
- **Worst Case**: Algorithm takes maximum possible time with unfavorable input
- **Average Case**: Mathematical expectation of execution time across all possible inputs

Mnemonic

"BWA - Best, Worst, Average"

Question 1(b) [4 marks]

What is Class and Object in OOP? Give suitable example.

Solution

Table 2: Class vs Object

Aspect	Class	Object
Definition	Blueprint/template for creating objects	Instance of a class
Memory	No memory allocated	Memory allocated when created
Example	Car (template)	my_car = Car()

```
\# Class definition
class Student:
    def \_\_init\_\_(self, name, age):
        self.name = name
        self.age = age

    def display(self):
        print(f"Name: \{self.name\}, Age: \{self.age\}")

\# Object creation
student1 = Student("John", 20)
student1.display()
```

- **Class**: Template defining attributes and methods
- **Object**: Real instance with actual values

Mnemonic

"Class = Cookie Cutter, Object = Actual Cookie"

Question 1(c) [7 marks]

Write a program for two matrix multiplication using simple nested loop and numpy module.

Solution

```
\# Method 1: Using Simple Nested Loop
def matrix\_multiply\_nested(A, B):
    rows\_A, cols\_A = len(A), len(A[0])
    rows\_B, cols\_B = len(B), len(B[0])

    \# Initialize result matrix
    result = [[0 for \_ in range(cols\_B)] for \_ in range(rows\_A)]

    \# Matrix multiplication
    for i in range(rows\_A):
        for j in range(cols\_B):
            for k in range(cols\_A):
                result[i][j] += A[i][k] * B[k][j]

    return result

\# Method 2: Using NumPy
import numpy as np

def matrix\_multiply\_numpy(A, B):
    A\_np = np.array(A)
    B\_np = np.array(B)
    return np.dot(A\_np, B\_np)

\# Example usage
A = [[1, 2], [3, 4]]
B = [[5, 6], [7, 8]]

print("Nested Loop Result:", matrix\_multiply\_nested(A, B))
print("NumPy Result:", matrix\_multiply\_numpy(A, B))
```

- **Nested Loop**: Three loops for row, column, and multiplication
- **NumPy**: Built-in dot() function for efficient multiplication