Subject Name (Gujarati)

4331601 -- Winter 2023

Semester 1 Study Material

Detailed Solutions and Explanations

પ્રશ્ન 1(a) [3 ગુણ]

Time Complexity માટે best case, worst case અને average case વ્યાખ્યાયિત કરો.

જવાબ

ટેબલ: Time Complexity Cases

કેસનો પ્રકાર	વ્યાખ્યા	ઉદાહરણ
Best Case	એલ્ગોરિધમ execution માટે લઘુત્તમ સમય	Linear search માં એલિમેન્ટ પહેલી પોઝિશન પર મળે
Worst Case	એલ્ગોરિધમ execution માટે મહત્તમ સમય	Linear search માં એલિમેન્ટ છેલ્લી પોઝિશન પર મળે
Average Case	સામાન્ય input scenarios માટે અપેક્ષિત સમય	Linear search માં એલિમેન્ટ મધ્યમાં મળે

- **Best Case**: આદર્શ input conditions સાથે એલ્ગોરિધમ optimal પ્રદર્શન આપે
- **Worst Case**: પ્રતિકૂળ input સાથે એલ્ગોરિધમ મહત્તમ સમય લે
- **Average Case**: બધા શક્ય inputs માં execution time ની ગાણિતિક અપેક્ષા

મેમરી ટ્રીક

``BWA - Best, Worst, Average''

પ્રશ્ન 1(b) [4 ગુણ]

OOP માં Class અને Object શું છે? યોગ્ય ઉદાહરણ આપો.

જવાબ

ટેબલ: Class vs Object

પાસું	Class	Object
વ્યાખ્યા	Objects બનાવવા માટે blueprint/template	Class નું instance
મેમરી	કોઈ મેમરી allocate નથી થતી	બનાવવામાં આવે ત્યારે મેમરી allocate થાય
ઉદાહરણ	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**: Attributes અને methods વ્યાખ્યાયિત કરતું template
- **Object**: વાસ્તવિક values સાથેનું instance

મેમરી ટ્રીક

``Class = Cookie Cutter, Object = વાસ્તવિક Cookie''

પ્રશ્ન 1(c) [7 ગુણ]

Simple nested loop અને numpy module નો ઉપયોગ કરીને બે matrix multiplication માટે પ્રોગ્રામ લખો.

જવાબ

```
\# Method 1: 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])

    \# Result matrix initialize
    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: 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)

\#
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**: Row, column અને multiplication માટે ત્રણ loops
- **NumPy**: કાર્યક્ષમ multiplication માટે built-in dot() function