Jigar Siddhpura
60004210155
C22

DATE
PAGE NO

Experiment 1B

Amortized Analysis (Accounding method)

Aim: To implement amotized analysis - accounting method

Theory: Accounting method — It involveds assigning amortized cost to operations, creating a global "account" to track these costs, & redistributing credits to offset expensive operations.

Steps to implement accounting method:

1. Define operations - Define basic operations peformed by algorithm & data structure.
2. Assigning amortized cost — Assign cost to each operation. This cost may differ from actual cost of performing operation.
3. Create global account — Have global account that track cumulative amortized costs. Account starts with initial balance of zero.
4. Perform operation — While performing operation, update global account based on difference between actual & amortized cost for each operation.
5. Redistribute credits — If actual cost of an operation is less than its amortized cost, Surplus represents "credits". Distribute these to global account. Conversely, if actual cost exceed amortized cost, charge deficit to global account.

→ The code below implements dynamic table using an amortized analysis technique called accounting method. Here, first initialization is done by creating dynamic table class which maintain size of table & actual table to store elements. Insert method is used to insert values, if table gets full, it doubles its capacity. The cost of each iteration is accounted for & a separate variable is used to account for doubling.

Resize method creates a new table with double capacity & copies the existing element into it. Cost calculation performs amortized analysis using accounting approach, where it tracks various costs : insertion cost, doubling cost, total cost, amortized cost, bank balance is used to ensure that it covers all operations.

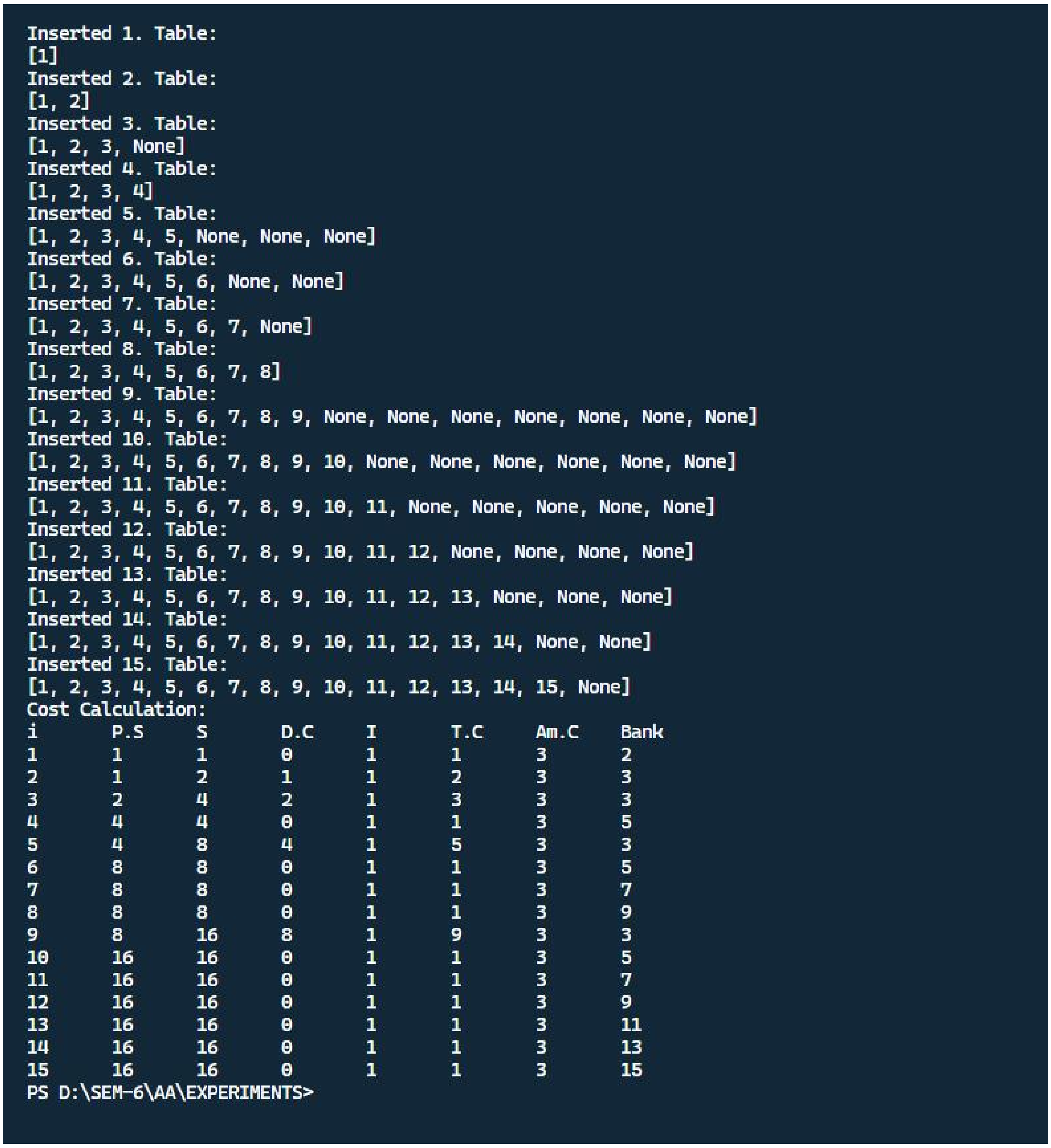
When table is resized, doubling cost is accounted to maintain amortized cost, bank balance helps to cover diff btw actual & amortized cost ensuring that over time avg cost remain constant.

Conclusion : We implement amortized analysis using accounting methods where tracking actual cost & allocating cost, during resizing code ensures that am. cost / op. remains constant

Code :

```
        class DynamicTable:
            def __init__(self):
                self.capacity = 1
                self.size = 0
                self.table = [None] * self.capacity

            def insert(self, value):
                if self.size == self.capacity:
                    self._resize(self.capacity * 2)
                else:
                    self.doubling_copy = 0
                self.table[self.size] = value
                self.size += 1

            def _resize(self, new_capacity):
                new_table = [None] * new_capacity
                for i in range(self.size):
                    new_table[i] = self.table[i]
                self.table = new_table
                self.capacity = new_capacity

            def print_table(self):
                print(self.table)

            def cost_calculation(self):
                insertion_cost = 1
                amortized_cost = 3
                doubling_copying = 0
                total_cost = insertion_cost + doubling_copying
                bank_balance = amortized_cost - total_cost
                size = 1
                previous_size = 1
                print("i\tP.S\tS\tD.C\tI\tT.C\tAm.C\tBank")
                for i in range(1, self.size + 1):
                    if i < size:
                        print(f"{i}\t{previous_size}\t{size}\t{doubling_copying}\t{insertion_cost}\t{total_cost}
\t{amortized_cost}\t{bank_balance}")
                        previous_size = size
                        doubling_copying = previous_size - size
                        total_cost = insertion_cost + doubling_copying
                        bank_balance = amortized_cost - total_cost + bank_balance
                    else:
                        print(f"{i}\t{previous_size}\t{size}\t{doubling_copying}\t{insertion_cost}\t{total_cost}
\t{amortized_cost}\t{bank_balance}")
                        previous_size = size
                        size = size * 2
                        doubling_copying = size - previous_size
                        total_cost = insertion_cost + doubling_copying
                        bank_balance = amortized_cost - total_cost + bank_balance

        table = DynamicTable()
        for i in range(1, 16):
            table.insert(i)
            print(f"Inserted {i}. Table:")
            table.print_table()
        print("Cost Calculation:")
        table.cost_calculation()
```

Output :



```
Inserted 1. Table:
[1]
Inserted 2. Table:
[1, 2]
Inserted 3. Table:
[1, 2, 3, None]
Inserted 4. Table:
[1, 2, 3, 4]
Inserted 5. Table:
[1, 2, 3, 4, 5, None, None, None]
Inserted 6. Table:
[1, 2, 3, 4, 5, 6, None, None]
Inserted 7. Table:
[1, 2, 3, 4, 5, 6, 7, None]
Inserted 8. Table:
[1, 2, 3, 4, 5, 6, 7, 8]
Inserted 9. Table:
[1, 2, 3, 4, 5, 6, 7, 8, 9, None, None, None, None, None, None, None]
Inserted 10. Table:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, None, None, None, None, None, None]
Inserted 11. Table:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, None, None, None, None, None]
Inserted 12. Table:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, None, None, None, None]
Inserted 13. Table:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, None, None, None]
Inserted 14. Table:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, None, None]
Inserted 15. Table:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, None]
Cost Calculation:
i       P.S     S       D.C     I       T.C     Am.C    Bank
1       1       1       0       1       1       3       2
2       1       2       1       1       2       3       3
3       2       4       2       1       3       3       3
4       4       4       0       1       1       3       5
5       4       8       4       1       5       3       3
6       8       8       0       1       1       3       5
7       8       8       0       1       1       3       7
8       8       8       0       1       1       3       9
9       8       16      8       1       9       3       3
10      16      16      0       1       1       3       5
11      16      16      0       1       1       3       7
12      16      16      0       1       1       3       9
13      16      16      0       1       1       3       11
14      16      16      0       1       1       3       13
15      16      16      0       1       1       3       15
PS D:\SEM-6\AA\EXPERIMENTS>
```