|  |  |
| --- | --- |
| Name: | Prerna Sunil Jadhav |
| Sap Id: | 60004220127 |
| Class: | T. Y. B. Tech (Computer Engineering) |
| Course: | Advance Algorithm Laboratory |
| Course Code: | DJ19CEL602 |
| Experiment No.: | 01-B |

**AIM: Perform Amortized Analysis of Multipop / Dynamic Tables / Binary Counter using**

**Aggregate, Accounting and Potential method. (Amortized Analysis)**

1B) Amortized Analysis (Accounting method)

**CODE:**

def accounting(n):

    size=1

    total=0

    dcost=0

    icost=0

    bank=0

    print("Elements\tDoubling Copying Cost\tInsertion Cost\tTotal Cost\tBank\t\tSize")

    for i in range(1,n+1):

        icost=1

        if i>size:

            size\*=2

            dcost=i-1

        total=icost+dcost

        bank+=(3-total)

        print(i,"\t\t\t",dcost,"\t\t\t",icost,"\t",total,"\t\t",bank,"\t\t",size)

        icost=0

        dcost=0

n=int(input("Enter number of elements:"))

print("Accounting method")

accounting(n)

class AccountingStack:

    def \_\_init\_\_(self):

        self.stack=[]

        self.cost=0

        self.balance=0

    def push(self,item):

        self.stack.append(item)

        self.cost+=1

        self.balance+=1

        self.printstack()

    def pop(self):

        self.stack.pop()

        self.cost+=1

        self.balance-=1

        self.printstack()

    def multipop(self,k):

        for i in range(k):

            self.pop()

    def printstack(self):

        print(self.stack,"\nBalance",self.balance,"\n")

s=AccountingStack()

s.push(1)

s.push(2)

s.push(3)

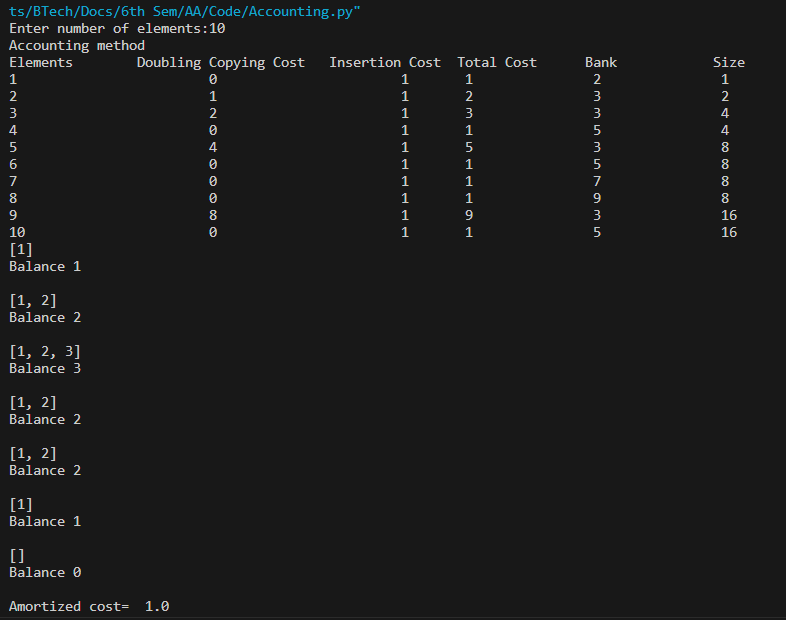
s.pop()

s.printstack()

s.multipop(2)

print("Amortized cost= ",s.cost/6)

**OUTPUT:**

****

**CONCLUSION:**  Hence we studied amortized analysis-Accounting method.