DATE:

Name: Prerna Sunil Jadhav
SapID: 60009220127
Batch: C22
Course: Advance Algorithm Lab

EXP1A

AIM: Perform Amortized Analysis using Aggregate Method

THEORY:

Amortized analysis is a technique used in computer science to analyse the average case time complexity of algorithm that perform a sequence of operation.
In aggregate analysis, we compute the total cost of a sequence of operations and divide it by the number of operations to get the average cost per operation.
Using aggregate analysis we can obtain a better upper bound that considers the entire sequence of n operations.
In aggregate analysis, we assign the amortized cost of each operation to be the average cost.
Each object can be POP only once for each time it is PUSHed. POP is at most PUSH, which is at most n.
Thus the average cost of an operation is $O(n)/n = O(1)$

DATE:

CONCLUSION: Thus we studied about the aggregate method in Amortized ~~Analysis~~.

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

AIM: Perform Amortized Analysis of Multipop / Dynamic Tables / Binary Counter using Aggregate, Accounting and Potential method. (Amortized Analysis)

1A) Amortized Analysis (Aggregate method)

CODE:

```
class AggregateStack:
    def __init__(self):
        self.stack=[]
        self.cost=0
    def push(self,item):
        self.stack.append(item)
        self.cost+=1
        self.printstack()
        print("\tCost: ",self.cost)
    def pop(self):
        self.stack.pop()
        self.cost+=1
        self.printstack()
        print("\tCost: ",self.cost)

    def multipop(self,k):
        for i in range(k):
            self.pop()

    def printstack(self):
        print(self.stack,end='')
s=AggregateStack()
s.push(10)
s.push(10)
s.push(10)
s.push(10)
s.multipop(2)

print("\n_____________________________")

def aggregate_dynamic(n):
    size=1
```

```
    icost=0
    dcost=0
    totalcost=0
    total=0

    print("Element\tDoubling Cost\tInsertion cost\tTotal cost")
    for i in range(1,n+1):
        icost=1
        if i > size:
            size*=2
            dcost=i-1
        totalcost=dcost+icost
        total=total+totalcost
        print(i,"\t\t",dcost,"\t\t",icost,"\t\t",totalcost,"")
        icost=0
        dcost=0
    return total/n

n=int(input("Enter the number of elemnets: "))
print("Aggregate method")
a=aggregate_dynamic(n)
print("Amortized cost= ",a)
```

OUTPUT:

```
PS C:\Users\Jadhav\Documents\BTech\Docs\6th Sem\AA\Code> & C:/msys64/mingw64/bin/python.exe "c:/Users/Jadhav/Doc
uments/BTech/Docs/6th Sem/AA/Code/Aggregate.py"
[10]    Cost:  1
[10, 10]        Cost:  2
[10, 10, 10]    Cost:  3
[10, 10, 10, 10]        Cost:  4
[10, 10, 10]    Cost:  5
[10, 10]        Cost:  6


_______________________________
Enter the number of elemnets: 6
Aggregate method
Element Doubling Cost   Insertion cost  Total cost
1               0               1               1
2               1               1               2
3               2               1               3
4               0               1               1
5               4               1               5
6               0               1               1
Amortized cost=  2.1666666666666665
PS C:\Users\Jadhav\Documents\BTech\Docs\6th Sem\AA\Code>
```