60004210210
Amartya Mishra
COMPS - EC31

AA Experiment 1 (C)

Aim: Amortized analysis (Potential method) of binary counter.

Theory:

The potential method is a technique in amortized analysis that uses a potential function to account for difference between the actual cost & the amortized cost of a operation. A potential function denoted as ϕ is defined to represent the accumulated potential or stored energy in the Data structure. The change in potential $\Delta\phi$ due to an operation is added to the actual cost to obtain amortized cost.

For dynamic table

$$\phi = (2 \times \text{no. of times}) - \text{Size of array}$$

For Binary counter:

$$\phi = \text{No. of Set Bits in given counter.}$$

Amortized cost using potential method

$$\hat{c}_i = c_i + \Delta\phi_i$$

Conclusion: We performed Amortized analysis using Potential Method on Binary Counter.

AA – Experiment 1(C)

Amartya Mishra

60004210210

COMPS C31

Code:

```cpp
#include<iostream>
#include<vector>
#include<bitset>

using namespace std;

int calculatecost( string number){
    int cost=0;
    for(auto a:number){
        if(a == '1') cost += 1;
    }
    return cost;
}

int main(){
    cout<<"number"<<"\t\t"<<"ActualCost"<<"\t\t"<<"potential"<<"\t\t"<<"Amortized cost"<<endl;
    for(int i =1;i<10; i++){
        int prev = i-1;
        int exor = prev^i;
        string binary = bitset<8>(exor).to_string();
        int potentialdiff = calculatecost(bitset<8>(i).to_string()) - calculatecost(bitset<8>(prev).to_string()) ;
        int cost =calculatecost(binary);


        cout<<i<<"\t\t"<<cost<<"\t\t\t"<<calculatecost(bitset<8>(i).to_string())<<"\t\t\t\t"<<cost + potentialdiff<<endl;
    }
    return 0;
}
```

Output:

Output

Clear

```
/tmp/K1AWDwJaOI.o
number      ActualCost      potential       Amortized cost
1           1               1               2
2           2               1               2
3           1               2               2
4           3               1               2
5           1               2               2
6           2               2               2
7           1               3               2
8           4               1               2
9           1               2               2
```