Uzair Akram

Mini Assignment 1

In My initial Approach i used an algorithm of O(kn) complexity

```
//Pseudo Code

OpenBoxes(N)
    int boxes[N] = {0}  //initialize all boxes to 0
    int count = 0
    for (int i = 1; i <=N; i++)
        for (int j = 0; j <=N; j+=i)
                if(boxes[j] == 0)
                    boxes[j] = 1, count++
                else
                    boxes[j] = 0, count--

    return count

main()
    int N = 0
    print "Enter the number of boxes:"
    input >> N
    print "Number of open boxes " << OpenBoxes(N)


```

Explanation:

To simulate the binary state of boxes: open/close, I used an int array of size N initialized at 0. The function OpenBoxes returns the Number of open boxes after N iterations by iterating 1 - N for each iteration it loops through the ids of boxes with increments of i (the ith number of iterations) so that it only loops through ids that are multiples of i and it achieves O(kN) time complexity.

I implemented the algorithm in C++, Here is the output:

Uzair Akram

OpenBoxes.cpp 〉 OpenBoxes(N)

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

using namespace std;

//Prototypes
int OpenBoxes(int N);
void PrintVector(vector<int> &V);
void InitVector(vector<int> &V, int N);

int main(){
    int N = 0;  //Number of boxes

    //get input
    cout << "Enter the Number of Boxes: ";
    cin >> N;

    //display output
    cout << "The Number of open boxes is " << OpenBoxes(N) << endl;

    return 0;
}

int OpenBoxes(int N){
    vector<int> boxes;
    InitVector(boxes, N);
    cout << endl << "Initial Vector" << endl;
    PrintVector(boxes);
    int count = 0;

    for (int i = 1; i <= N; i++){
        cout << "itteration:" << i << endl;
        for (int j = i; j <=N; j+=i){
                if (boxes[j-1] == 0){
                    boxes[j-1] = 1; count++;
                }else if (boxes[j-1] == 1){
                    boxes[j-1] = 0; count--;
                }
        }
        PrintVector(boxes);
    }

    cout << "Count:";
    return count;
}
```

```
itteration:4
1 0 0 1 1 1 1
itteration:5
1 0 0 1 0 1 1
itteration:6
1 0 0 1 0 0 1
itteration:7
1 0 0 1 0 0 0
Count:2
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 8
The Number of open boxes is
Initial Vector
0 0 0 0 0 0 0 0
itteration:1
1 1 1 1 1 1 1 1
itteration:2
1 0 1 0 1 0 1 0
itteration:3
1 0 0 0 1 1 1 0
itteration:4
1 0 0 1 1 1 1 1
itteration:5
1 0 0 1 0 1 1 1
itteration:6
1 0 0 1 0 0 1 1
itteration:7
1 0 0 1 0 0 0 1
itteration:8
1 0 0 1 0 0 0 0
Count:2
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 9
The Number of open boxes is
Initial Vector
0 0 0 0 0 0 0 0 0
itteration:1
1 1 1 1 1 1 1 1 1
itteration:2
1 0 1 0 1 0 1 0 1
itteration:3
1 0 0 0 1 1 1 0 0
itteration:4
1 0 0 1 1 1 1 1 0
itteration:5
1 0 0 1 0 1 1 1 0
itteration:6
1 0 0 1 0 0 1 1 0
itteration:7
1 0 0 1 0 0 0 1 0
itteration:8
1 0 0 1 0 0 0 0 0
itteration:9
1 0 0 1 0 0 0 0 1
Count:3
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$
```

There is a more efficient approach, The Number of opened boxes is equal to the number of perfect squares in the N number of boxes.
This Algorithm runs with of O(M(k)) complexity.
Here is the pseudo code:

```
//Pseudo Code

OpenBoxes(N)
    int count = 0
    for (int i = 1; i*i <=N; i++)
        if (i*i <= N)
            count++

    return count

main()
    int N = 0
    print "Enter the number of boxes:"
    input >> N
    print "Number of open boxes " << OpenBoxes(N)

```

Uzair Akram

There seems to be a mathematical relationship. The Number of opened boxes in the N number of boxes equals the floor square root of N.

EvalOpenBoxes.cpp 〉 OpenBoxes(N)

```cpp
#include <iostream>
#include <math.h>

using namespace std;

//prototypes
int OpenBoxes(int N);

int main(){
    int N = 0;  //Number of boxes

    //get input
    cout << "Enter the Number of Boxes: ";
    cin >> N;

    //display output
    cout << "The Number of open boxes is " << OpenBoxes(N) << endl;

    return 0;
}

int OpenBoxes(int N){
    int openned = floor(sqrt(N));
    return openned;
}
```

```
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ g++ EvalOpenBoxes.cpp
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 4
The Number of open boxes is 2
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 5
The Number of open boxes is 2
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 6
The Number of open boxes is 2
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 7
The Number of open boxes is 2
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 8
The Number of open boxes is 2
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 9
The Number of open boxes is 3
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 10
The Number of open boxes is 3
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 11
The Number of open boxes is 3
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$ ./a.out
Enter the Number of Boxes: 16
The Number of open boxes is 4
(base) Uzairs-MacBook-Pro:CSCE4110 uzairakram$
```

This runs in O(c) which is the cost of evaluating the expression.