



# LRU-Implementation

14.04.2019

---

Submitted to: Sateesh Kumar Peddoju

Mansi Agarwal: 17114048

Hritvi Bhandari: 17114039

Harshit Maurya: 17114037

Abhijeet Shakya: 17114002

Pulkit Jeph: 17114060

Atul Sinha: 17114016

Rahul Gome: 17114062

Gaurav Dewat: 17114032


Github link: [https://github.com/hritvi/lru\\_implementation](https://github.com/hritvi/lru_implementation)

## Introduction

LRU or Least Recently Used algorithm as the name suggests mainly discards the least recently used items (in a memory cache) first.

We have analyzed the following variations of the algorithm as needed and implemented them as per the specified requirements of the project:

1. LRU-Counter Method
2. LRU-Stack Method
3. LRU-Aging Register Method
4. Approx. LRU; Clock or Second Chance Method



In order to infer how different inputs to the above implementations generate varying results and why they do so, we came up with certain corner cases as mentioned below and used the number of Frames coupled with the number of Page Faults generated as an analysis metric along with the execution times.

## What are some *Corner Cases*?

### I. Same Input

All the pages have the same value stored, e.g. in our analysis, we used a continuous input of 20 integers of value 7.

### II. Repeating input

The input consists of a repeating pattern of say, integers. Our implementation uses a repeating pattern of the type '7 6 5 4 3 2 1 7 6 5 4 3 2 1 7 6...' up to 20 values.

### III. Non-repeating input

The input consists of a non-repeating pattern of integers where all the values differ. For example, in our analysis, we have made use of a sequence of integers of the form '9 5 7 8 4 2 6 0 3 1'.

## Graphs for *No. of Frames vs No. of Faults*

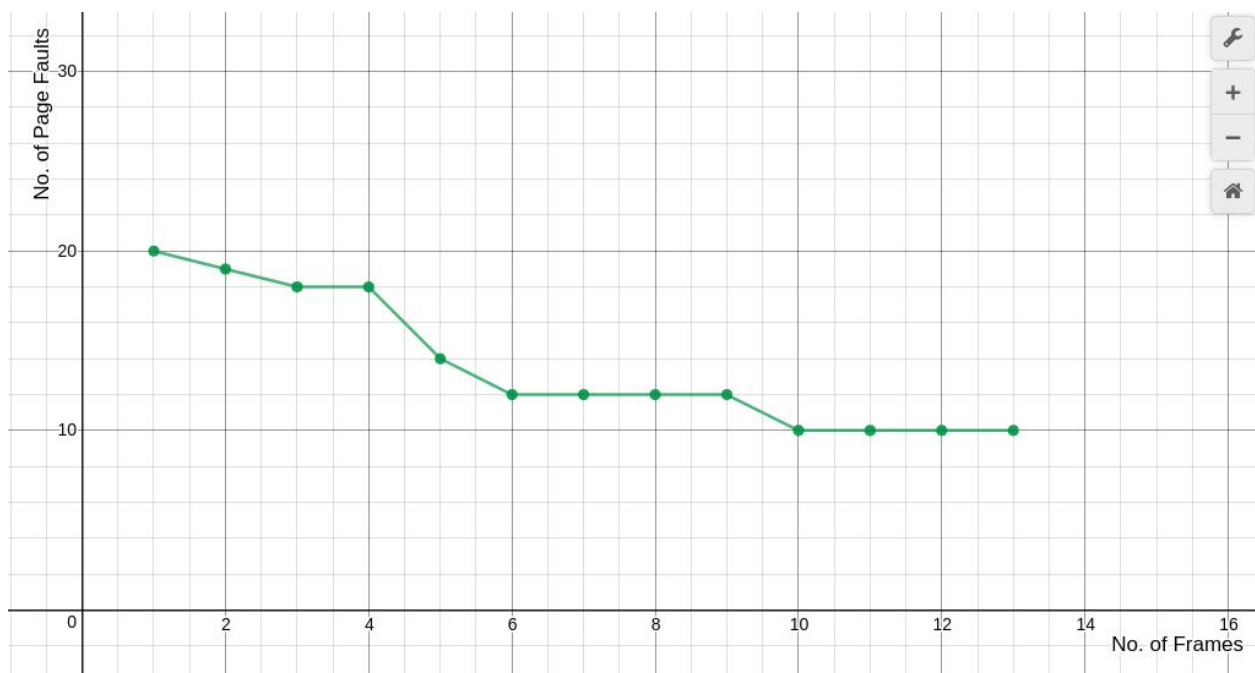
### Input Sequence:

Size : 20

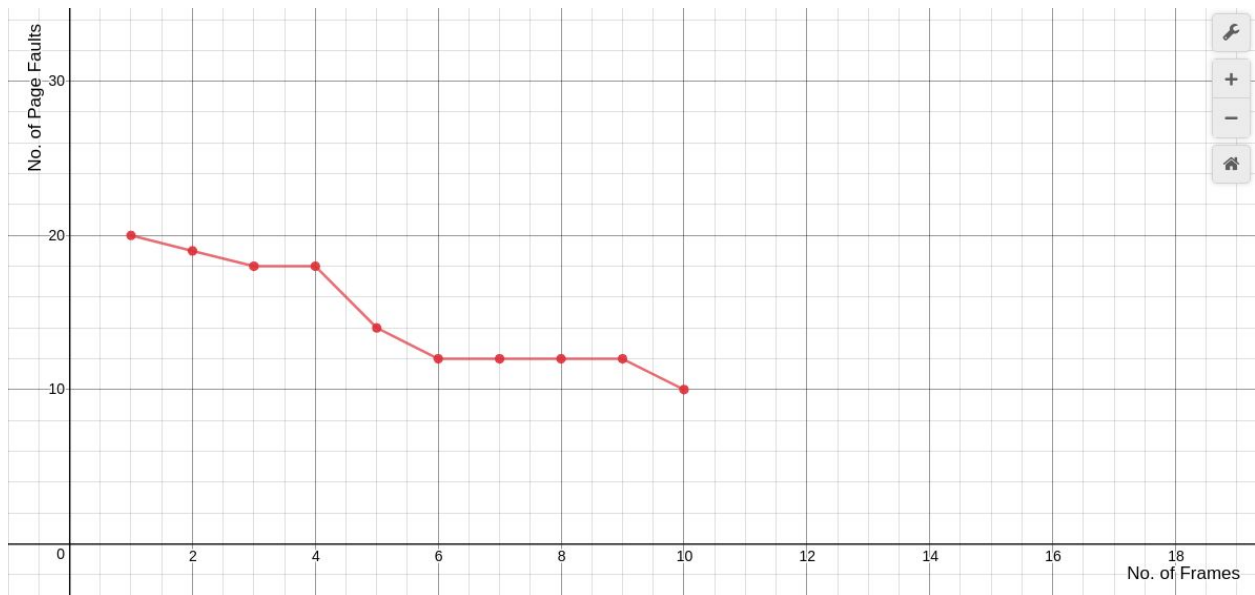
Sequence : 7 0 1 2 0 3 5 4 2 6 9 3 8 3 2 7 9 8 6 1

For this sequence, number of distinct integers are 10 ( 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ), so for any value of number of frames  $\geq 10$  the value of number of page faults will remain 10.

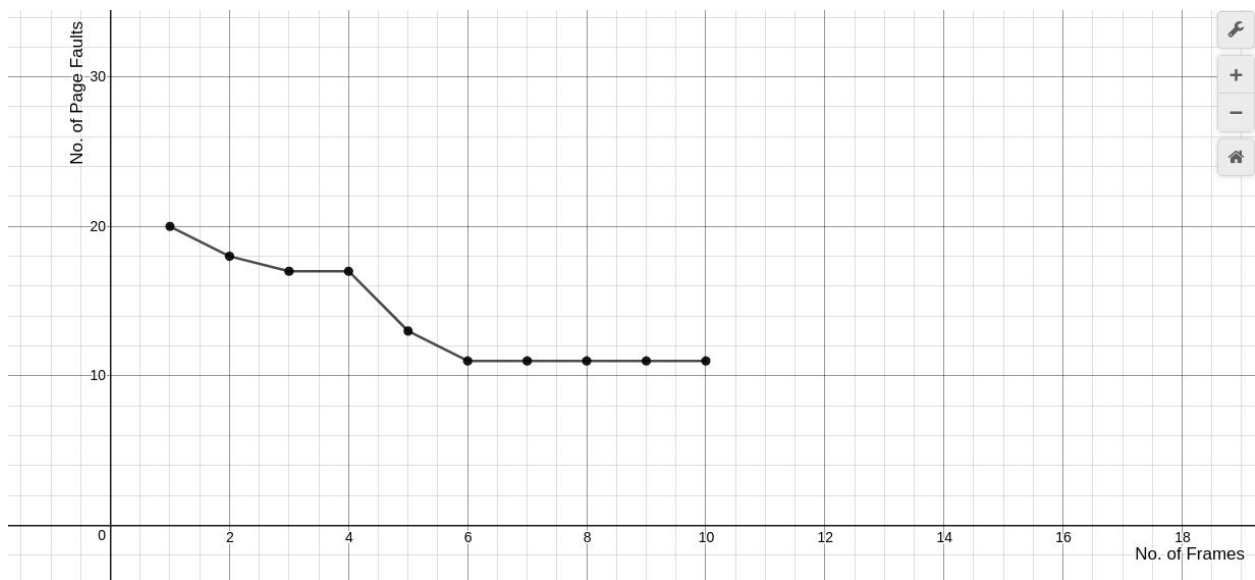
### I. LRU (counter implementation):



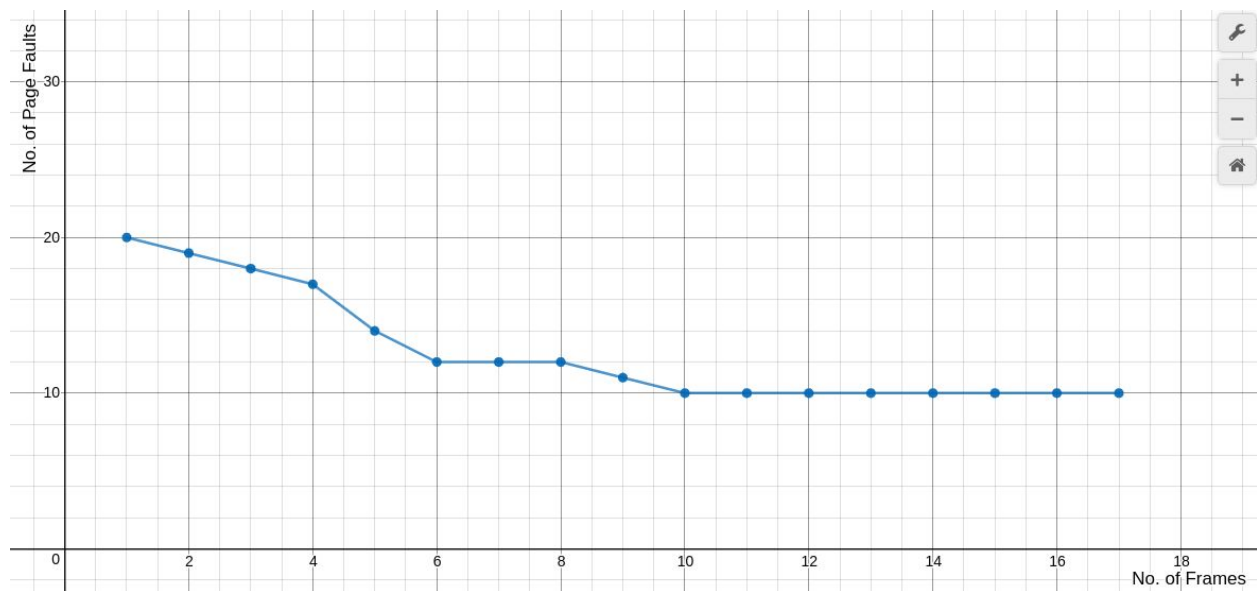
## II. LRU (stack implementation):



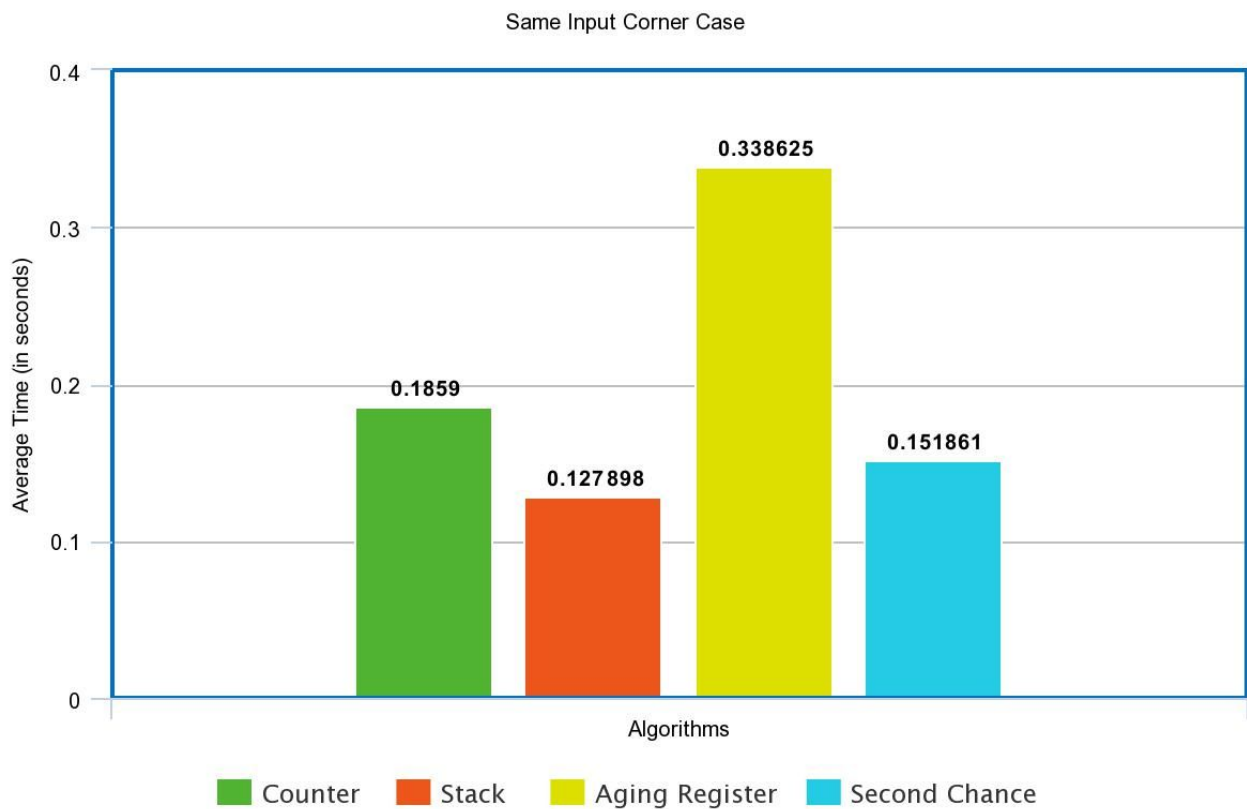
## III. LRU (Aging Register Method):

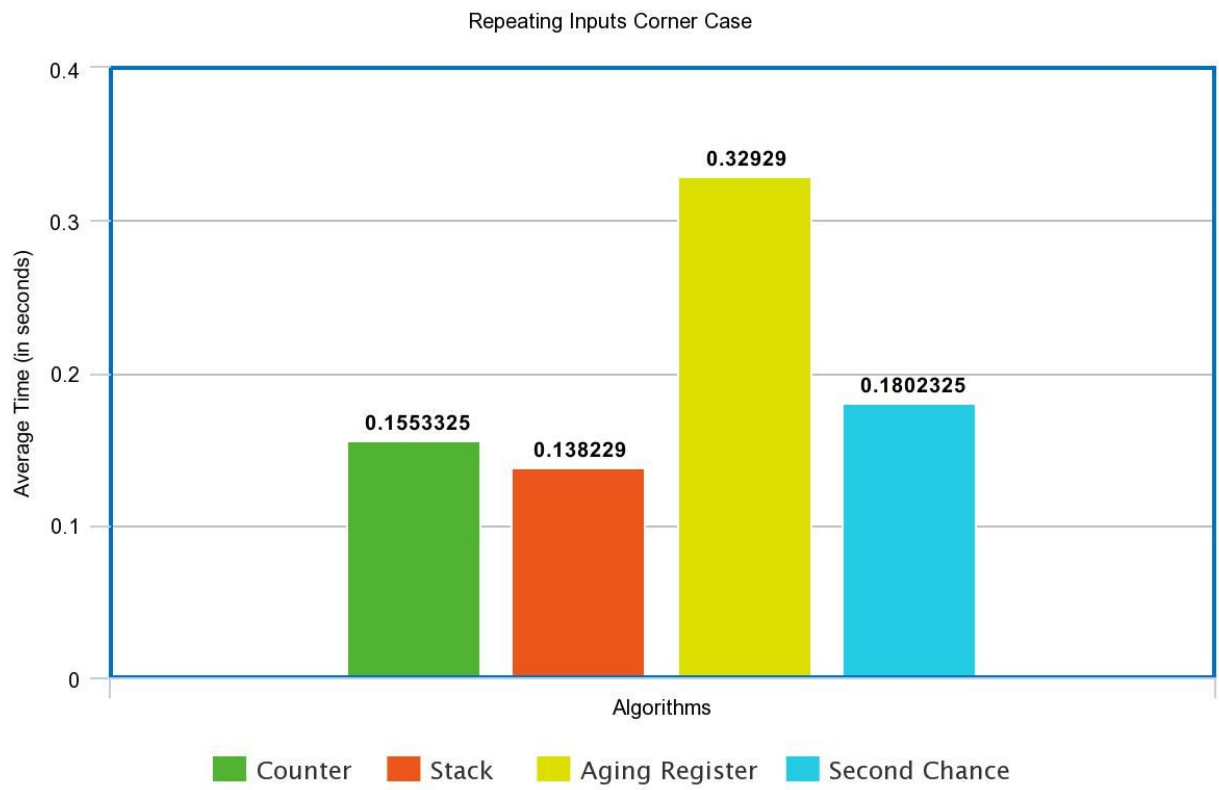


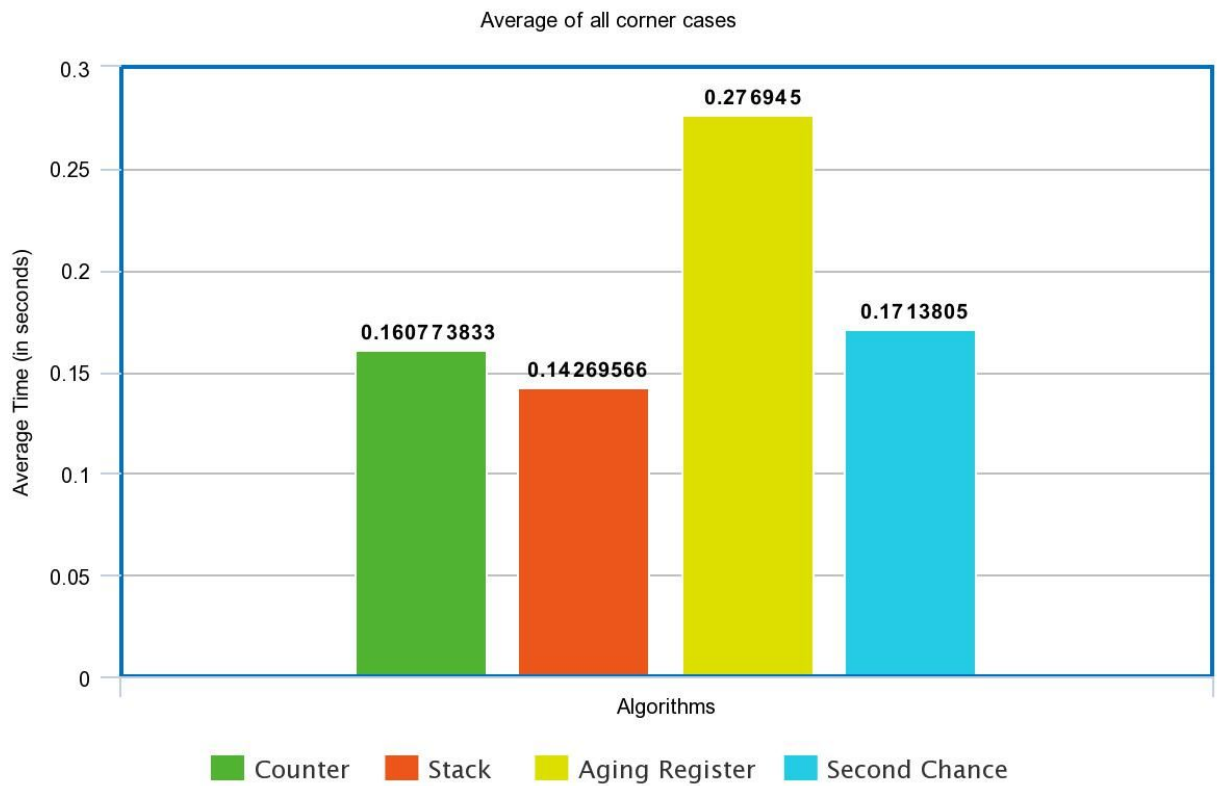
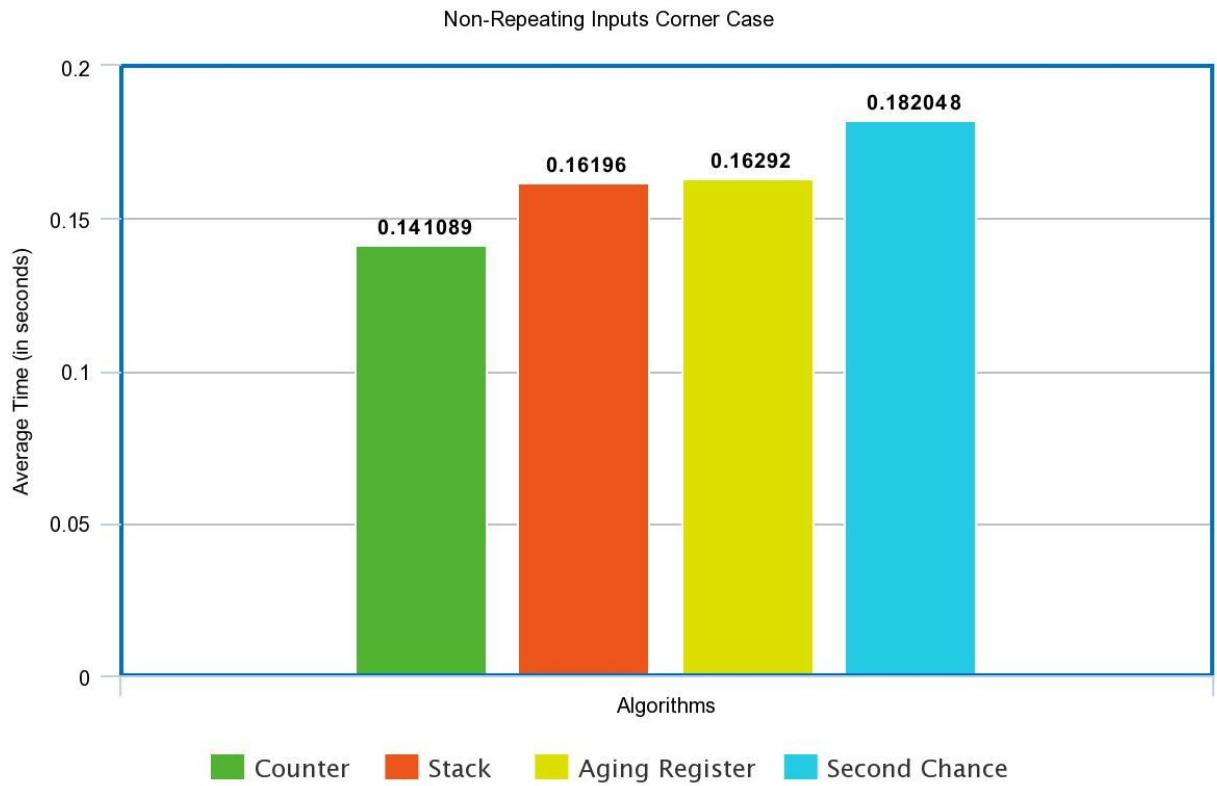
#### IV. Clock (Approximate LRU) Algorithm or Second Chance Algorithm:



#### Graphs for *Average Execution Time* for different *Corner Cases*







## Complexity Analysis

Note -  $T(m,n)$  denotes the time complexity dependent on  $m$  and  $n$ .

- LRU (counter implementation)

Let no. of pages be  $m$ .

Let no. of frames be  $n$ .

$$T(m,n) = O(n) + O(m) + O(m*n) + O(m*n) + O(n) + O(n)$$

$$\Rightarrow T(m,n) = O(m*n)$$

- LRU (stack implementation)

Let no. of pages be  $m$ .

$$T(m) = O(m)$$

- LRU (Aging Register Method)

Let no. of frames be  $n$ .

Let no. of requests be  $m$ .

$$T(n) = O(m*(n+n+n))$$

$$\Rightarrow T(m,n) = O(m*n)$$

- Clock (Approximate LRU) Algorithm or Second Chance Algorithm

Let the number of input elements =  $n$

Let the variable full =  $m$

$$T(n) = O(n) + O(n*m)$$

$$\Rightarrow T(n) = O(n*m)$$



## Critical Analysis of achieved results and complexity

1. Let us first analyze the page-fault vs no-of-frames graphs.
  - The general trend is observed that with the increasing **number of frames**, the number of **page faults** decreases monotonically
  - Its terminal value is the number of different type of frames.
2. The graph of our major interest is **Average time** vs **Algorithm** graph for different corner cases.
  - The general observation is that The **aging register** implementation took the most amount of time and the **stack** implementation took the least amount of CPU time.
  - In the **Same input corner case**, the order of finishing times was:  
Aging > Counter > Clock > Stack
  - In the **Repeating input corner case**, the order was as follows:  
Aging > Clock > Counter > Stack
  - In the **Non Repeating input corner case**, the order was as follows:  
Clock > Aging > Stack > Counter
  - In the **Average input corner case**, the order was as follows:  
Aging > Clock > Counter > Stack
3. Analysis of the time complexity
  - The time complexity primarily depends upon two major variables, viz. **Number of pages** and **number of frames**.
  - Stack implementation does not depend upon the number of frames, being the fastest among the four.
  - The other three have comparable performance wherein clock/second\_chance has a slight edge over the others.