Contents

[USCSP301 – USCS303: Operating System (OS) Practical – 09 2](#_Toc81313525)

[Practical – 09: Page Replacement Algorithm Least Recently Used (LRU) 2](#_Toc81313526)

[Practical Date: 30th August 2021 2](#_Toc81313527)

[ Algorithm 2](#_Toc81313528)

[ Page Replacement Algorithm 2](#_Toc81313529)

[ Least Recently Used (LRU) Algorithm 2](#_Toc81313530)

[ Solved Example 2](#_Toc81313531)

[ Example – 01 2](#_Toc81313532)

[▪ Example – 02 4](#_Toc81313533)

[▪ Example – 03 5](#_Toc81313534)

[ Question 5](#_Toc81313535)

[ Implementation 6](#_Toc81313536)

[File Name: P9\_PR\_LRU\_YP.java 6](#_Toc81313537)

[ Input Of Example – 01 8](#_Toc81313538)

[ Output Of Example – 01 8](#_Toc81313539)

[ Input Of Example – 02 8](#_Toc81313540)

[ Output Of Example – 02 9](#_Toc81313541)

[ Input Of Example – 03 9](#_Toc81313542)

[ Output Of Example – 03 9](#_Toc81313543)

[ Sample Output Of Example - 01 10](#_Toc81313544)

[ Sample Output Of Example – 02 10](#_Toc81313545)

[ Sample Output Of Example – 03 10](#_Toc81313546)

**USCSP301 – USCS303: Operating System (OS) Practical – 09**

## **Practical – 09: Page Replacement Algorithm Least Recently Used** (LRU)

# Practical Date: 30th August 2021

**Practical Aim:** Page Replacement Algorithm (LRU)

# Algorithm

# Page Replacement Algorithm

* + - In demand paging memory management technique, if a page demanded for execution is present in main memory, then a page fault occurs.
    - To load the page in demand into main memory, a free page frame is searched in main memory and allocated.
    - If no page frame is free, memory manager has to free a frame by swapping its contents to secondary storage and thus make room for the required page.
    - To swap pages, many schemes or strategies are used.

# Least Recently Used (LRU) Algorithm

* + - The **Least Recently Used (LRU) algorithm** replaces the page that has not been used for the longest period of time.
    - It is based on the observation that pages that have not been used for long time will probably remain unused for the longest time and are to be replaced.

# Solved Example

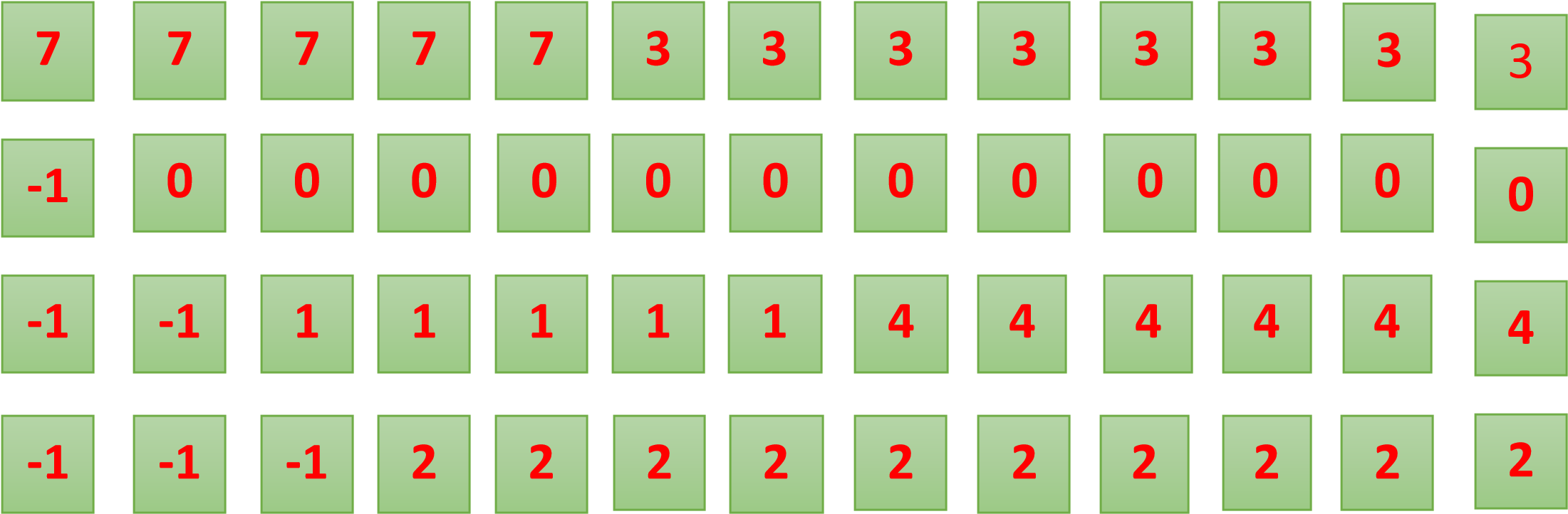
#  Example – 01

* + Apply the LRU replacement algorithms for the following page-reference strings: 7,0,1,2,0,3,0,4,2,3,0,3,2
  + Indicate the number of page faults for LRU algorithm assuming demand paging with four frames.
  + Find the number of hits, number of faults and hit ratio

**Solution**

**Page Reference String:** 7,0,1,2,0,3,0,4,2,3,0,3,2

**Demand Paging or Number of Frames:** 4



|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** | **0** | **3** | **2** |
|  | | | | | |  | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

**Number of Hits:** Count of no replacements = 7 

**Number of Faults:** Count of replacements = 6

**Hit Ratio:** Number of Hits/Len (Ref String) = 7/13 = 0.53

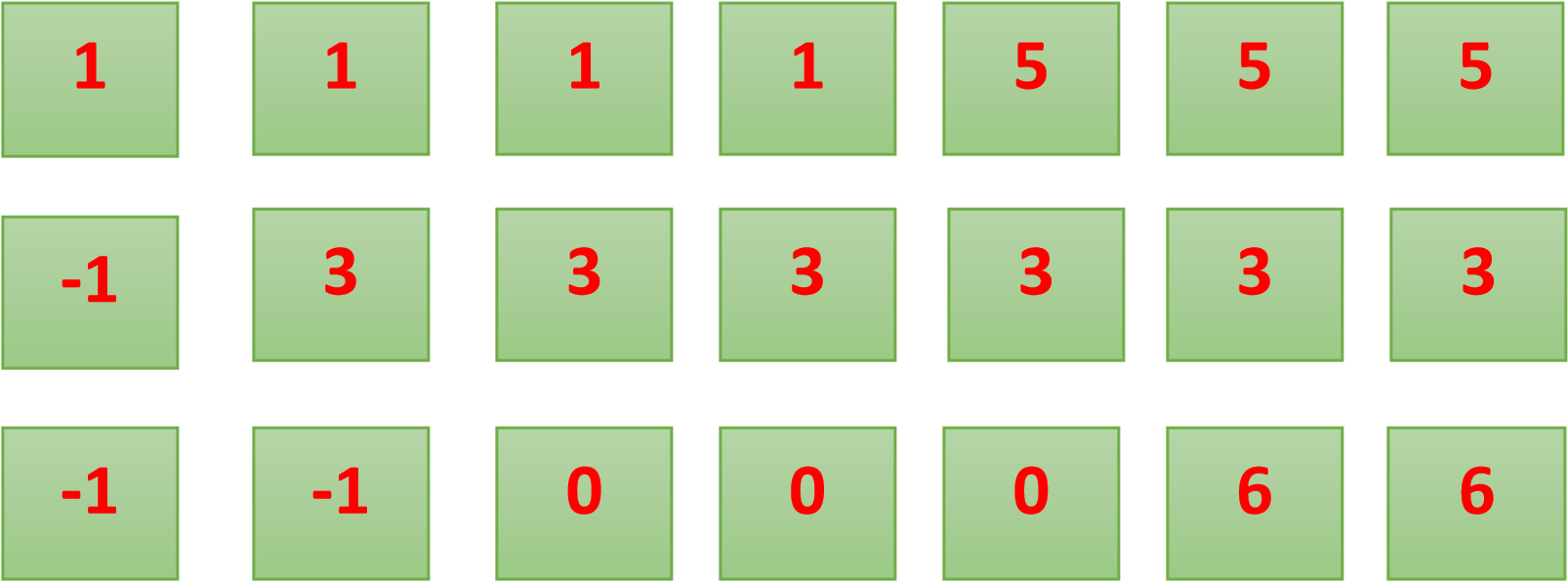
# Example – 02

* Consider the following example 3 frames with 1,3,0,3,5,6,3 page-reference strings.
* Find the number of hits, number of faults and hit ratio using LRU Page Replacement Algorithm.

**Solution:**

**Page Reference String:** 1,3,0,3,5,6,3

**Demand Paging or Number of Frames:** 3



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **3** | **0** | **3** | **5** | **6** | **3** |
|  | | | | | |  |
|  |  |  |  |  |  |  |

**Number of Hits:** Count of no replacements = 2 

**Number of Faults:** Count of replacements = 5

**Hit Ratio:** Number of Hits/Len (Ref String) = 2/7= 0.28

# Example – 03

* Consider the following example 3 frames with

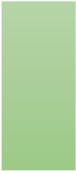
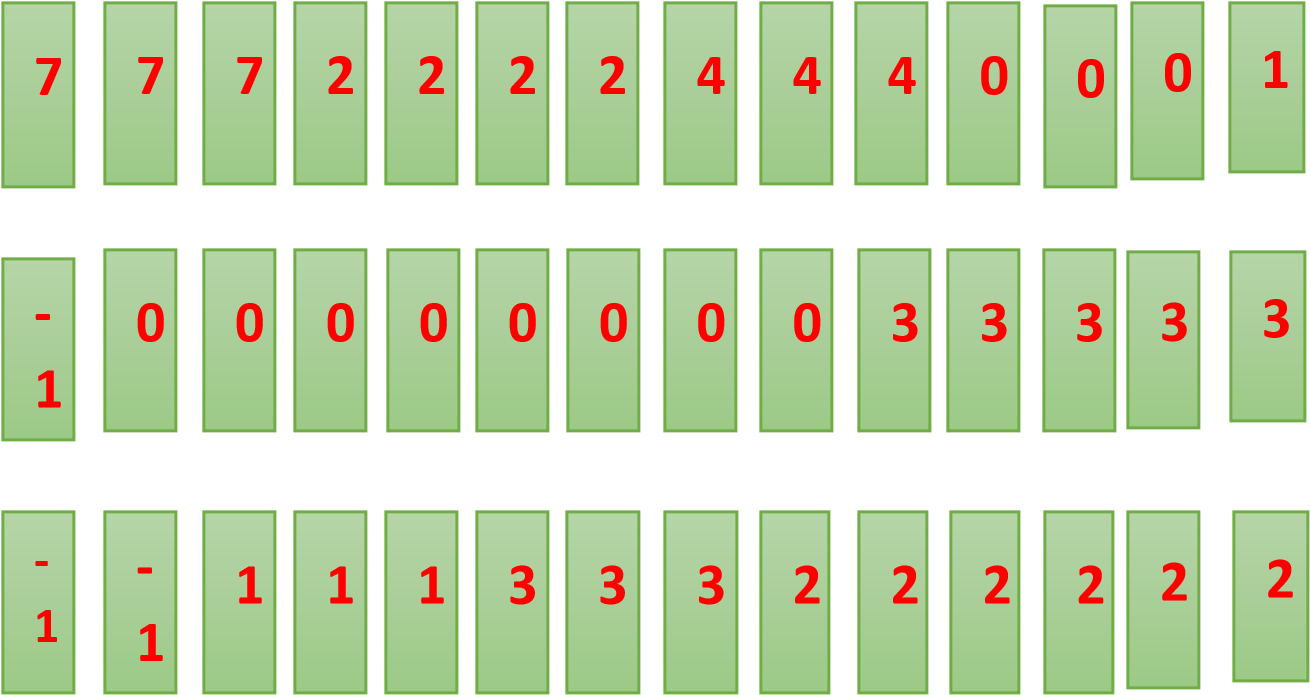
7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 page-reference strings.

* Find the number of hits, number of faults and hit ratio using LRU Page Replacement Algorithm.

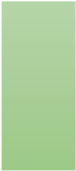
**Solution:**

**Page Reference String:** 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

**Demand Paging or Number of Frames:** 3



**1**



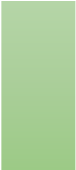
**3**



**2**



**1**



**0**



**2**



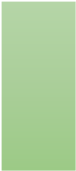
**1**



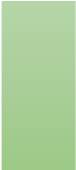
**0**



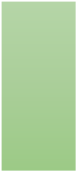
**2**



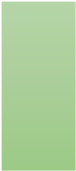
**1**



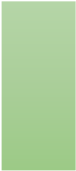
**0**



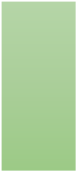
**7**



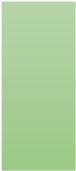
**1**



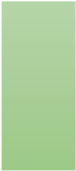
**0**



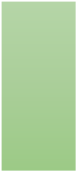
**7**



**1**



**0**



**7**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **7** | **0** | | **1** | | **2** | | | **0** | | **3** | | **0** | | **4** | | **2** | | **3** | | **0** | | **3** | | **2** | | **1** | | **2** | | **0** | | **1** | | **7** | **0** | | **1** | |
|  | |  | |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | |

**Number of Hits:** Count of no replacements = 8 

**Number of Faults:** Count of replacements = 12

**Hit Ratio:** Number of Hits/Len (Ref String) = 8/20 = 0.4

# Question

* Write a Java program that implements the LRU page-replacement algorithm.

# **Implementation**

# File Name: P9\_PR\_LRU\_YP.java

// Name: Yash Parab

// Batch: B1

// PRN: 2020016400922513

// Date: 30 August,2021

// Prac-09: Page Replacement Algorithm LRU

import java.io.\*;

import java.util.\*;

public class P9\_PR\_LRU\_YP

{

public static void main(String[] args) throws IOException

{

Scanner scan = new Scanner(System.in);

int frames,pointer = 0, hit = 0, fault = 0,ref\_len;

Boolean isFull = false;

int buffer[];

ArrayList<Integer>stack = new ArrayList<Integer>();

int reference[];

int mem\_layout[][];

System.out.print("Please enter the number of Frames: ");

frames = scan.nextInt();

System.out.print("Please enter the length of the References strings: ");

ref\_len = scan.nextInt();

reference = new int[ref\_len];

mem\_layout = new int[ref\_len][frames];

buffer = new int[frames];

for(int j = 0; j < frames; j++)

buffer[j] = -1;

System.out.print("Please enter the references strings: ");

for(int i = 0; i < ref\_len; i++)

{

reference[i] = scan.nextInt();

}

System.out.println();

for(int i = 0; i < ref\_len; i++)

{

if(stack.contains(reference[i]))

{

stack.remove(stack.indexOf(reference[i]));

}

stack.add(reference[i]);

int search = -1;

for(int j = 0; j < frames; j++)

{

if(buffer[j] == reference[i])

{

search = j;

hit++;

break;

}

}

if(search == -1)

{

if(isFull)

{

int min\_loc = ref\_len;

for(int j = 0; j < frames; j++)

{

if(stack.contains(buffer[j]))

{

int temp = stack.indexOf(buffer[j]);

if(temp < min\_loc)

{

min\_loc = temp;

pointer = j;

}

}

}

}

buffer[pointer] = reference[i];

fault++;

pointer++;

if(pointer == frames)

{

pointer = 0;

isFull = true;

}

}

for(int j = 0; j < frames; j++)

mem\_layout[i][j] = buffer[j];

}

for(int i = 0; i < frames; i++)

{

for(int j = 0; j < ref\_len; j++)

System.out.printf("%3d",mem\_layout[j][i]);

System.out.println();

}

System.out.println("The number of Hits: " + hit);

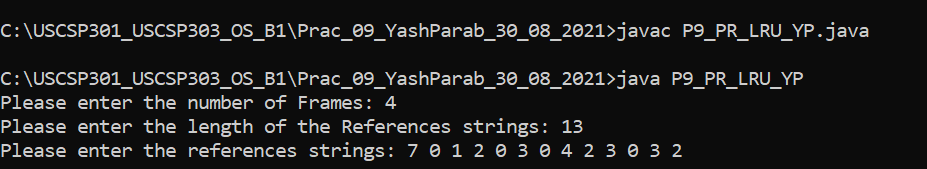
System.out.println("Hit Ratio: " +(float)((float)hit/ref\_len));

System.out.println("The number of Faults: " + fault);

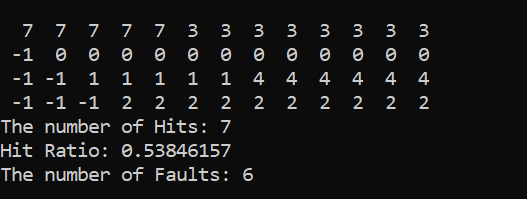
}

}

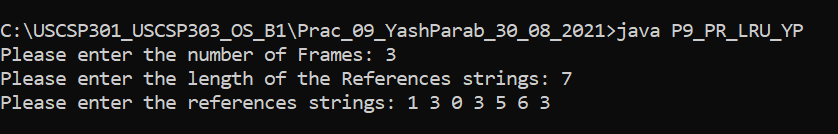
# Input Of Example – 01



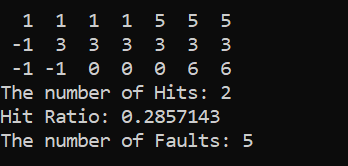
# Output Of Example – 01



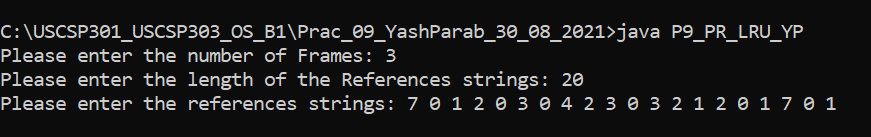
# Input Of Example – 02



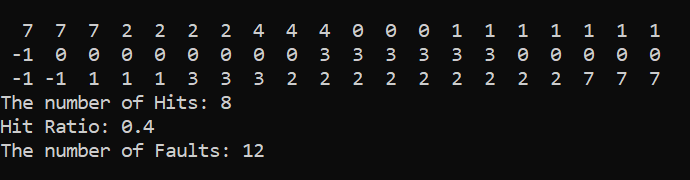
# Output Of Example – 02



# Input Of Example – 03

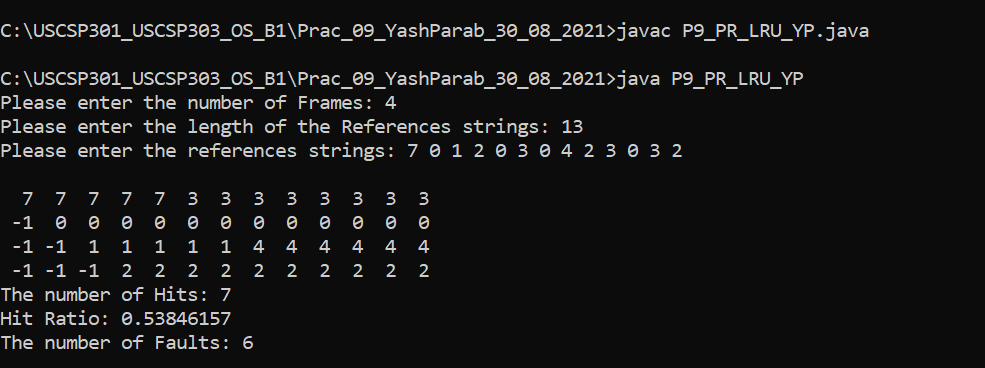


# Output Of Example – 03

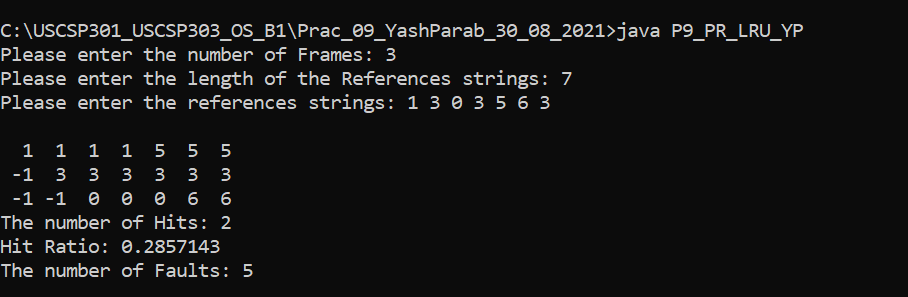


# 

# Sample Output Of Example - 01



# Sample Output Of Example – 02



# Sample Output Of Example – 03

