# **USCSP301-U****SCS303: Operating System (OS) Practical-09**

## 

## **Practical-09 : Page Replacement Algorithm LRU**

Contents

**U**[**SCS303: Operating System (OS) Practical-09** 1](#_Toc81314600)

[**Practical-09 : Page Replacement Algorithm LRU** 1](#_Toc81314601)

[**Practical Date** 2](#_Toc81314602)

[**Practical Aim** 2](#_Toc81314603)

[**Algorithm** 2](#_Toc81314604)

[**Solved Example** 3](#_Toc81314605)

[**Question - 01** 3](#_Toc81314606)

[**Question - 02** 4](#_Toc81314607)

[**Question - 03** 5](#_Toc81314608)

[**Question** 5](#_Toc81314609)

[**Implementation** 5](#_Toc81314610)

[**Input** 8](#_Toc81314611)

[**Output** 8](#_Toc81314612)

[**Sample Output** 9](#_Toc81314613)

[**Sample Output - 01** 9](#_Toc81314614)

[**Sample Output - 02** 9](#_Toc81314615)

[**Sample Output - 03** 10](#_Toc81314616)

### 

### **Practical Date :** 31-08-2021 (Tuesday)

### **Practical Aim :** Page Replacement Algorithm LRU

* **Content:**
* In LRU page replacement algorithm the page that has not been

used for the longest Period of the time is chosen and

replaced.

* **Process :**
* Implement LRU algorithm and find out page hits and page

faults.

### **Algorithm :**

* **Prior Knowledge :**
* Page Replacement Algorithm.

**Page Replacement Algorithm**

* In demand paging memory management technique, if a page demanded for execution is not present in main memory, then a page fault occurs.
* To load the page in demand into main memory, a free page frame is searching main memory and allocated.
* If no page frame is free, Memory Manager has to free a frame by swapping its content to secondary storage and thus make room for the required page.
* To swap pages many screens of strategies are used.

**Least Recently Used (LRU)**

* The Least Recently used (LRU) algorithm replaces the page that has not been used for the longest period of time.
* The Least Recently used (LRU) algorithm replaces the page that has not been used for the longest period of time.

### **Solved Example :**

### **Question – 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 you algorithm assuming demand paging with four frames.
* Find the number of hits, number of faults and hit ratio.

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

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

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

### 

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 |
| 🗶 | 🗶 | 🗶 | 🗶 | ✓ | 🗶 | ✓ | 🗶 | ✓ | ✓ | ✓ | ✓ | ✓ |

**Number of Hits:** count of number replacements = 7

**Number of faults:** count of replacements = 6

**Hit Ratio :** Number of hits / len( Ref string)= 7/13 = 0.53846157

**Question – 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 page using LRU Page Replacement Algorithm.

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

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

**Number of Hits:** 2

**Number of faults:** 5

**Hit Ratio :** 2/7 = 0.2857

### **Question – 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 page using LRU Page Replacement Algorithm

**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**

**Number of Hits:** 8

**Number of faults:** 12

**Hit Ratio :** 8/20 = 0.4

### **Question:**

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

### **Implementation:**

//Name:sahil jadhav

//Batch No:B2

//PRN:2020016400783091

//Date:31-08-2021

//Prac-09:Page Replacement Algorithm LRU

import java.io.\*;

import java.util.\*;

public class P9\_PR\_LRU\_SJ

{

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 Reference string: ");

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.println("Please enter the reference string:");

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:**

**Please enter the number of frames : 4**

**Please enter the length of Reference string : 13**

**Please enter the reference string** :

**7 0 1 2 0 3 0 4 2 3 0 3 2**

### **Output:**

### **7 7 7 7 7 3 3 3 3 3 3 3 3**

### **-1 0 0 0 0 0 0 0 0 0 0 0 0**

### **-1 -1 1 1 1 1 1 4 4 4 4 4 4**

### **-1 -1 2 2 2 2 2 2 2 2 2 2 2**

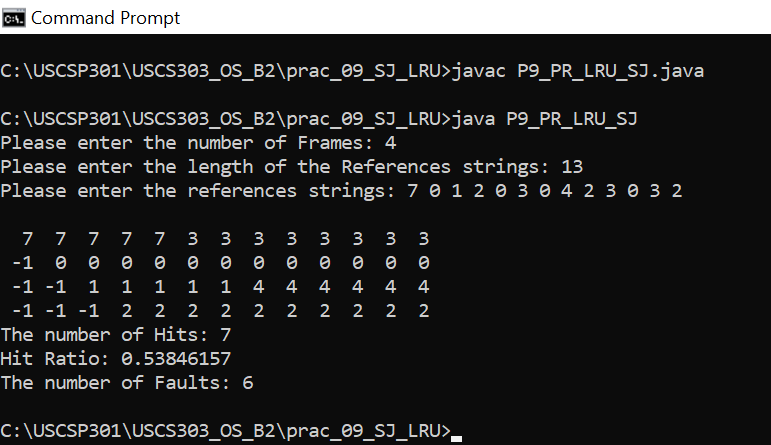
**Number of Hits:** 7

**Number of faults:** 6

**Hit Ratio :** 7/13 = 0.53846157

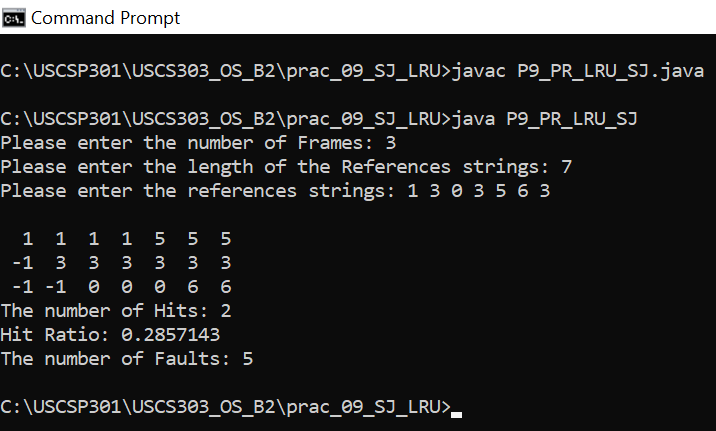
### **Sample Output:**

### **Sample Output – 01**



### 

### **Sample Output – 02**



### **Sample Output – 03**

