Contents

[USCSP3O1\_USCS303: OPERATING SYSTEM(OS) Practical 09 2](#_Toc81239003)

[Practical-09: Page Replacement Algorithm LRU 2](#_Toc81239004)

[Practical Date: 30th September 2021 2](#_Toc81239005)

[Practical Aim: Page Replacement Algorithm LRU 2](#_Toc81239006)

[ALGORITHM 2](#_Toc81239007)

[LEAST RECENTLY USED (LRU) 2](#_Toc81239008)

[SOLVED EXAMPLE: 2](#_Toc81239009)

[QUESTION: 3](#_Toc81239010)

[IMPLEMENTATION: 3](#_Toc81239011)

[INPUT 7](#_Toc81239012)

[OUTPUT 8](#_Toc81239013)

[SAMPLE OUTPUT 1 8](#_Toc81239014)

[SAMPLE OUTPUT 2 8](#_Toc81239015)

[SAMPLE OUTPUT 3 9](#_Toc81239016)

# USCSP3O1\_USCS303: OPERATING SYSTEM(OS) Practical 09

## Practical-09: Page Replacement Algorithm LRU

### Practical Date: 30th September 2021

### Practical Aim: Page Replacement Algorithm LRU

### ALGORITHM

1. In demand paging memory management technique, if a page demanded for execution is not present in main memory, then a page fault occurs.
2. To load the page in demand into main memory, a free page frame is searched in main memory and allocated.
3. If no 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.
4. To swap pages, many schemes or strategies are used.

### LEAST RECENTLY USED (LRU)

1. The Least recently used (LRU) algorithm replaces the page that has not nbeen used for the longest period of time.
2. It is based on the observation that pages that have not been used for long time will probably remained unused for the longest time and are to be replaced.

### SOLVED EXAMPLE:

Apply the LRU replacement algorithms for the following page-reference strings:7,0,1,2,0,3,0,4,2,3,0,3,2.

1. Indicate the number of page faults for LRU algorithm assuming demand paging with four frames.
2. 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** | **-1** | **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 no replacements = 7

Number of Faults: count of replacements = 6

Hit Ratio: Number of Hits/Len(Ref String) = 7/13 = 0.53846157

### QUESTION:

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

**Example 2:**

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

Number of Hits: count of no replacements = 2

Number of Faults: count of replacements = 7

Hit Ratio: Number of Hits/Len(Ref String) =2/9 = 0.2857143

**Example 3:**

(1)Consider the following example 3 frames with 7,0,1,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 page-reference strings.

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

Number of Hits: count of no replacements = 5

Number of Faults: count of replacements = 20

Hit Ratio: Number of Hits/Len(Ref String) = 5/20 = 0.25

### IMPLEMENTATION:

//Name: Ritika Sahu

//Batch : B1

//PRN:2020016400783543

//Date: 30 August, 2021.

//Practical 9: Page Repalcenment Algorithm LRU

import java.io.\*;

import java.util.\*;

public class P9\_PR\_LRU\_RS

{

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

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

{

reference[i] = scan.nextInt();

}

System.out.printIn();

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[i]))

{

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.printIn();

}

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

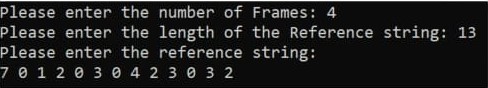
System.out.printIn("Hit Ratio: " + (float)((float)hit/ref\_len));

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

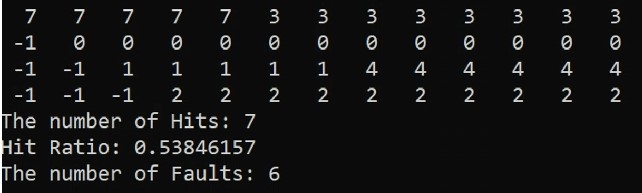
}

}

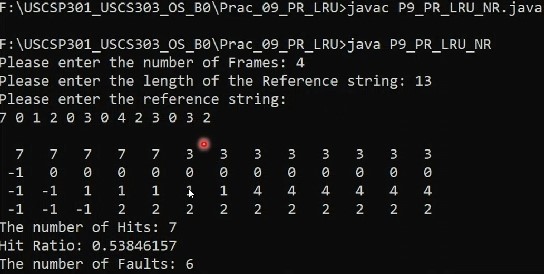
### INPUT



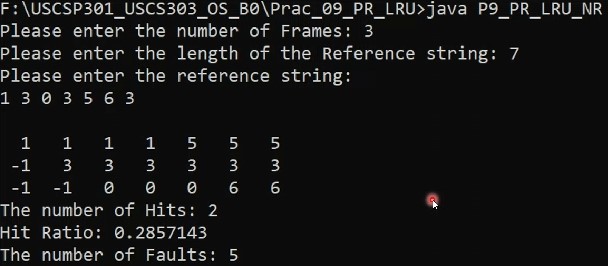
### OUTPUT



### SAMPLE OUTPUT 1



### SAMPLE OUTPUT 2



### SAMPLE OUTPUT 3

