USCS303 – OS: Practical – 09: Page Replacement Algorithm LRU

Practical Aim: Page Replace Algorithm: LRU

[Practical Aim: 30th Aug, 2021 1](#_Toc81238838)

[Page Replace Algorithm 2](#_Toc81238839)

[Least Recently Used (LRU) 2](#_Toc81238840)

[Solved Example 2](#_Toc81238841)

[Question 3](#_Toc81238842)

[Implementation 3](#_Toc81238843)

[Input 7](#_Toc81238844)

[Output 7](#_Toc81238845)

[Sample Output 8](#_Toc81238846)

# Page Replace 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 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)

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

It is based on the observation that pages that have not been used for a long time will probably remain unused for the longest time and are to be replaced.

# Solved Example

**Question:**

Apply the LRU replacement algorithm 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 | 7 | 7 | 7 | 7 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | -1 | 1 | 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 |
| -1 | -1 | -1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

**Number of Hits:** count of no replacement = 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.

# Implementation

**Code**

// Name: Sumit Telawane

// Batch: B1

// PRN: 2020016400825777

// Date: 30th August 2021

// prac 09: Page Replacement Algorithm (LRU)

import java.io.\*;

import java.util.\*;

public class P9\_PR\_LRU\_ST {

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 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));

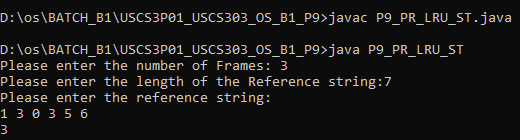
System.out.println("The number of Faults: " + fault);

}

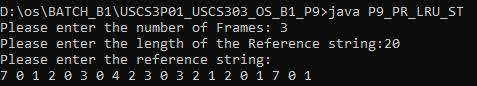
}

# Input

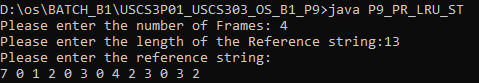
**Input 01:**

****

**Input 02:**

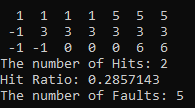
****

**Input 03:**

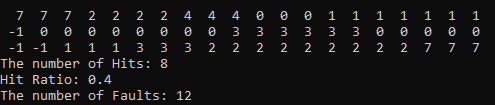
****

# Output

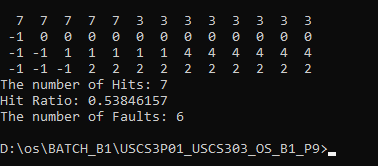
**Output 01:**

****

**Output 02:**

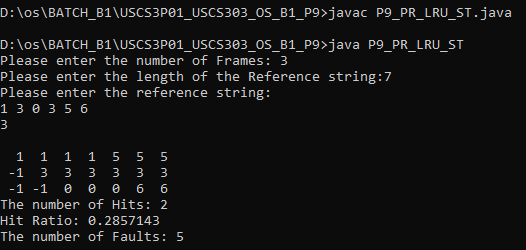
****

**Output 03:**

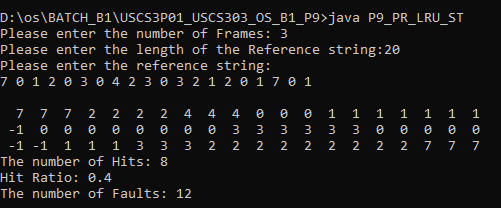
****

# Sample Output

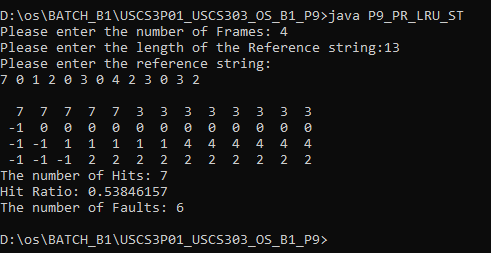
**Sample output 01:**

****

**Sample output 02:**

****

**Sample output 03:**

****