Name: Pritam Mogal
Div:- ~~B~~ TE ~~(B)~~ comp
Assignment No.12 Roll No:17

Page Replacement Algorithm.

Assignment D-I.

Aim :- Implementing page replacement algorithm
1) LRU
2) Optimal.

Problem Statement:- To write a java program (using OOP feature) to implement LRU & optimal algorithm for page Replacement.

Pre-requisite:-
1. Explain the concept of virtual meomory.
2. Define page replacement algorithm: LRU & optimal
3. Explain address translation in paging system.
4. Explain Belady's Anomaly.

Theory:-
Whenever there is a page reference for which the page needed in meomory, that erent is called page fault or page fetch or page failure situation. In such case we have to make space in meomory for this new page by replacing any existing page. But we cannot replace any page. We have to replace a page which is not used currently.

There are sereral algorithms to achiere.
1) Last recently used (LRU)
2) optimal.

1) LRU Page replacement:
The main difference between FIFO and optimal

page replacement is that the FIFO algorithms uses the time when the page was brought in to meamory and optimal algorithm uses the time when a page is to be used.
Now, consider reference string 7,0,1,2,0,3,4,2, 3,0,3 with three meamory frames or blocks.

1	2	3	4	5	6	7	8	9	10	11	12
	7	0	1	2	0	3	0	4	2	3	0
	7	7	7	2	2	2	2	4	4	4	0
		0	0	0	0	0	0	0	0	3	3
		1	1	1	3	a	3	2	2	2	2
	+	+	+	+		+		+			+

2) Optimal page replacement:
The algorithm has lowest page fault rate of all algorithm. This algorithm state that: Replace the page which will not be used for longest period of time i.e future Knowledge of reference string is required.

- Consider the following reference string
0,2,1,6,4,0,1,0,3,1,2,1

Compulsory misses X X X, X X

$$\begin{array}{|c|}\hline 0\\ 2\\ 1\\ 6\\ \hline\end{array} \xrightarrow{4} \begin{array}{|c|}\hline 0\\ 2\\ 1\\ 4\\ \hline\end{array} \xrightarrow{3} \begin{array}{|c|}\hline 3\\ 2\\ 1\\ 4\\ \hline\end{array}$$

- Fault rate = 6/12 = 0.50
- With the above reference string this is the best we can hope to do.

Algorithm For LRU:-

```
1- Start traversing the pages.
i) if set holds less pages than capacity.
    a) Insert page into the set one by one until the
       size of set reaches capacity or all page
       requests are processed.
    b) Simultaneously maintain the recent occured
       index of each page in a map called indexes.
    c) Increment page fault
ii) Else
    If current page is present in set, do nothing
    Else
    a) find the page in the set that was least
       recently used.
    b) Replace the found page with current page
    c) Increment page faults.
    d) Update index of current page.
```

Algorithm For optimal

```
1. Start the process

2. Delare the size
3. Get the number of pages to be inserted.
4. Get the value
5. Compare counter label and stack.
6. Select the optimal page by counter value.
7. Stack them according the selection
8. Print pages with fault pages.
9. Stop the process.
```

Name : Pritam Gotiram Mogal **DivRoll**: TE-B-17

Assignment No. 12

Problem Satement: To write a java program (using OOP feature) to implement LRU & Optimal algorithm for Page Replacement.

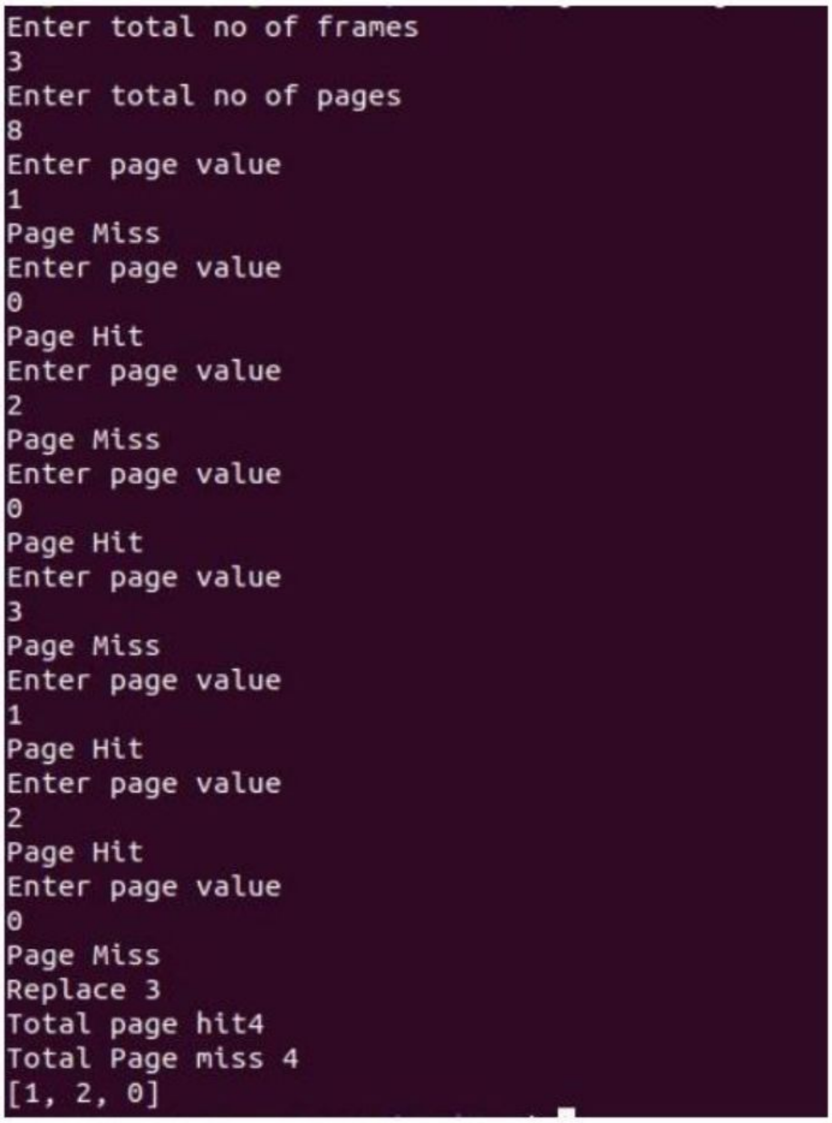
1. LRU (Last Recently Used) Program:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;

public class LRU
{


                public static void main(String[] args)throws Exception {
                        int hit=0;
                        int miss=0;

                        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));

                        System.out.println("Enter total no of frames");
                        int noFrames=Integer.parseInt(br.readLine());

                        int[] frames=new int[noFrames];
                        int[] lruTime=new int[noFrames];

                        System.out.println("Enter total no of pages");
                        int totalPages = Integer.parseInt(br.readLine());

                        for(int i=0;i<totalPages;i++){
                                        System.out.println("Enter page value");
                                        int page= Integer.parseInt(br.readLine());
                                        int searchIndex=isPresent(frames, page );

                                          if(searchIndex!=-1){
//        page fonud
                                                        hit++; lruTime[searchIndex]=i;
                                                        System.out.println("Page
                                                        Hit");
                                        }
                                        else{
                                                System.out.println("Page Miss");
                                                miss++;
//          page not found
                                        int emptyindex=isEmpty(frames);
                                             if(emptyindex!=-1){
//          if frame is empty
                                                       frames[emptyindex]=page;

                                                       lruTime[emptyindex]=i;
                                                 }
                                                 else{
            //user lru algo to find replace location int minLocationIndex=lru(lruTime);

                                                        System.out.println("Replace "+
frames[minLocationIndex]);

                                                       frames[minLocationIndex]=page;
                                                       lruTime[minLocationIndex]=i;

                                                     }

                                          }

                              }


                                    System.out.println("Total page hit" + hit);
                                  System.out.println("Total Page miss " + miss);
                                   System.out.println(Arrays.toString(frames));

                        }

                        public static int lru(int[] lruTime){ int min = 9999; int
                                          index = -1; for(int
                                          i=0;i<lruTime.length;i++){

                                                 if(min>lruTime[i]){
                                                       min=lruTime[i];
                                                       index=i;
                                                     }

                                          }

                                                return index;
                        }

                        public static int isEmpty(int[] frames){

                                        for(int i=0;i<frames.length;i++)
                                    { if(frames[i]==0){
                                          return i;
                                          }
                                    }
                                                  return -1;
```

```
                }

                public static int isPresent(int[] frames, int search){

                        for(int i=0;i<frames.length;i++){
                                if(frames[i]==search)
                                return i;
                        }

                        return -1;
                }


}
```

OUTPUT:

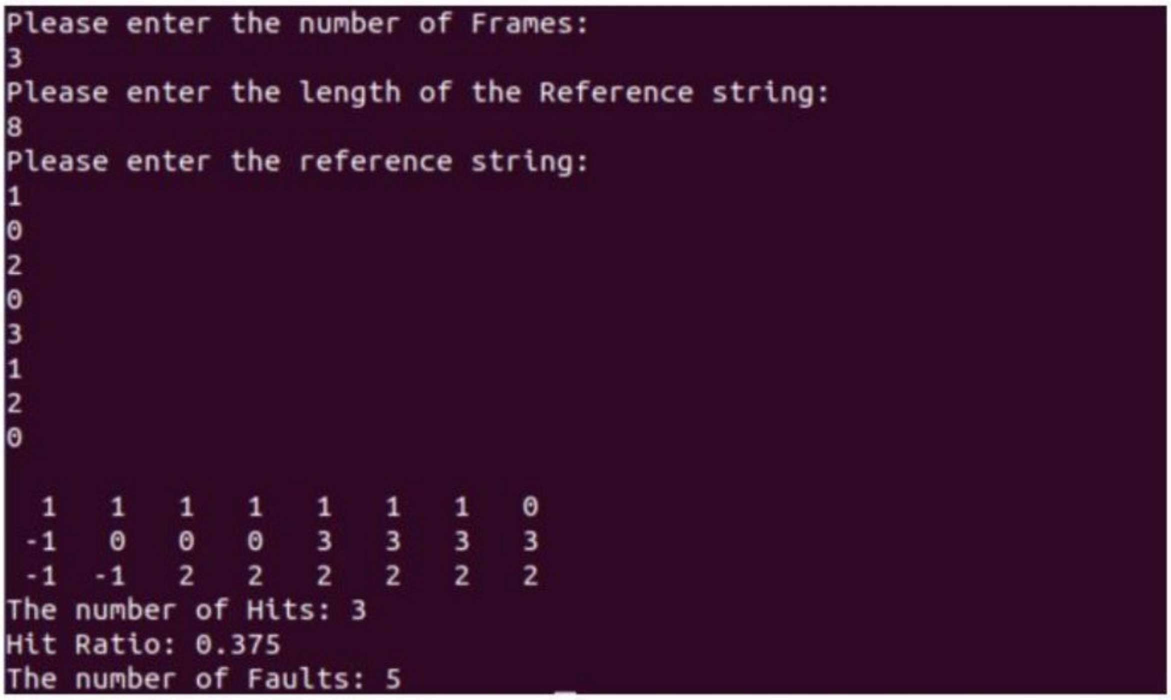
2. Optimal Replacement Program:



```
Enter total no of frames
3
Enter total no of pages
8
Enter page value
1
Page Miss
Enter page value
0
Page Hit
Enter page value
2
Page Miss
Enter page value
0
Page Hit
Enter page value
3
Page Miss
Enter page value
1
Page Hit
Enter page value
2
Page Hit
Enter page value
0
Page Miss
Replace 3
Total page hit4
Total Page miss 4
[1, 2, 0]
```

```
import java.io.BufferedReader;
import java.io.IOException; import
java.io.InputStreamReader; public
class OptimalReplacement {
 public static void main(String[] args) throws IOException
 {
 BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));  int frames,
pointer = 0, hit = 0, fault = 0,ref_len;  boolean
isFull = false;  int buffer[];  int reference[];  int
mem_layout[][];

 System.out.println("Please enter the number of Frames: ");
frames = Integer.parseInt(br.readLine());

 System.out.println("Please enter the length of the Reference string:");
ref_len = Integer.parseInt(br.readLine());

 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] = Integer.parseInt(br.readLine());
 }
 System.out.println();  for(int
i = 0; i < ref_len; 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 index[] = new int[frames];  boolean
index_flag[] = new boolean[frames];  for(int j
= i + 1; j < ref_len; j++)
 {
 for(int k = 0; k < frames; k++)
 {
 if((reference[j] == buffer[k]) && (index_flag[k] == false))
 {  index[k] = j;
index_flag[k] = true;
break;  }
 }  }  int max =
index[0];  pointer =
0;  if(max == 0)
max = 200;
 for(int j = 0; j < frames; j++)
 {  if(index[j] ==
0)  index[j] = 200;
if(index[j] > max)
 {  max =
index[j];
pointer = j;
 }
 }
}
 buffer[pointer] = reference[i];
fault++;  if(!isFull)  {
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);  }

}
```

OUTPUT:



```
Please enter the number of Frames:
3
Please enter the length of the Reference string:
8
Please enter the reference string:
1
0
2
0
3
1
2
0

  1   1   1   1   1   1   1   0
 -1   0   0   0   3   3   3   3
 -1  -1   2   2   2   2   2   2
The number of Hits: 3
Hit Ratio: 0.375
The number of Faults: 5
```