

Fokane Sakshi Anil

TE-A-42

Assignment No:-12

Page No.	
Date	

①

## Page Replacement Algorithm.

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

problem statement :- To write a java program (using OOP features) to implement LRU & optimal algorithm for page replacement.

Pre-requisites :-

- 1) Explain the concept of virtual memory.
- 2) Define page replacement algorithm LRU & optimal.
- 3) Explain addr translation in paging system.
- 4) Explain Belady's Anomaly.

Theory :-

Whenever there is page reference for which the page needed in memory that event is called page fault or page fetch or page failure situation.

- There are several algorithm to achieve :- 1) last recently used (LRU) 2) Optimal.

1) LRU page replacement :-



The main difference between FIFO & optimal page replacement is that the FIFO algorithm uses the time when a page is to be used.

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.

\* Algorithm for LRU :-

i) Start traversing the pages.

2) Set holds less pages than capacity

a) insert page into the set one by one

While the size of set reaches capacity or all page request are processed.

b) Simultaneously maintain the recent occurred index of each page in a Map called 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.



- Page No. \_\_\_\_\_  
Date \_\_\_\_\_
- ③
- b) Replace the found page with current page.
  - c) Increment page fault
  - d) update index of current page
  - e) Return page fault

\* Algorithm for optimal :

- 1) Start the process
- 2) Declare the size
- 3) Get the number of page to be inserted.
- 4) Get the value
- 5) Compare counter label & stack
- 6) select the optimal page by counter value
- 7) Stack them according the selection
- 8) print page with page fault
- 9) stop the process.

\* Java program for LRU :-

```
import java.util.HashMap;  
import java.util.HashSet;  
import java.util.Iterator;  
class Test {  
    static int pageFault(int page[], int n,  
        int capacity)  
    {  
        HashSet<Integer> s = new HashSet<>  
            (capacity)  
        HashMap<Integer, Integer> indexs =  
            new HashMap<>
```



```

int page_fault = 0;
for (int i = 0; i < n; i++) {
    if (s.size() < capacity) {
        if (!s.contains(page[i])) {
            s.add(page[i]);
            page_fault++;
        }
        indexes.put(page[i], i);
    }
    else {
        if (!s.contains(page[i])) {
            int lru = Integer.MAX_VALUE;
            val = Integer.MIN_VALUE;
            Integer<Integer> itr = s.iterator();
            while (itr.hasNext()) {
                int temp = itr.next();
                if (indexes.get(temp) < lru) {
                    lru = indexes.get(temp);
                    val = temp;
                }
            }
            s.remove(val);
            s.add(page[i]);
            page_fault++;
        }
        indexes.put(page[i], i);
    }
}
return page_fault;
}

```



public static void main (Strings  
arg[])

{

int pages[] = { 7, 10, 2, 0, 3, 0, 4, 2, 3, 0,  
3, 2 };

int capacity = '4';

System.out.println (pageFaults (pages,  
pages.length, capacity));

}



```
//Name:Fokane Sakshi Anil
// TE-A 42
// ASSINGNMENT:GROUP_D_1
```

```
/*
```

**Problem Statement :**

**Write a Java Program (using OOP features) to implement paging simulation using**

**1. Least Recently Used (LRU)**

**2. Optimal algorithm**

**\*\*\*\*LRU\*\*\*\***

```
*/
```

```
import java.io.*;
class lru
{
    public static void main(String args[])throws IOException
    {
        BufferedReader obj=new BufferedReader(new InputStreamReader(System.in));
        int f,page=0,ch,pgf=0,n,chn=0;
        boolean flag;
        int pages[];          //pgf-page fault

        System.out.println("1.LRU");
        int pt=0;
        System.out.println("enter no. of frames: ");
        f=Integer.parseInt(obj.readLine());
        int frame[]=new int[f];

        for(int i=0;i<f;i++)
        {
            frame[i]=-1;
        }

        System.out.println("enter the no of pages ");
        n=Integer.parseInt(obj.readLine());

        pages=new int[n];
        System.out.println("enter the page no ");

        for(int j=0;j<n;j++)
            pages[j]=Integer.parseInt(obj.readLine());
```

```

int pg=0;
for(pg=0;pg<n;pg++)
{
    page=pages[pg];
    flag=true;
    for(int j=0;j<f;j++)
    {
        if(page==frame[j])
        {
            flag=false;
            break;
        }
    }
    int temp,h=3,i;
    if(flag)
    {
        if( frame[1]!=-1 && frame[2]!=-1 && frame[0]!=-1)
        {
            temp=pages[pg-3];
            if(temp==pages[pg-2] || temp==pages[pg-1])
                temp=pages[pg-4];

            for(i=0;i<f;i++)
                if(temp==frame[i])
                    break;
            frame[i]=pages[pg];
        }
        else
        {
            if(frame[0]==-1)
                frame[0]=pages[pg];
            else if(frame[1]==-1)
                frame[1]=pages[pg];
            else if(frame[2]==-1)
                frame[2]=pages[pg];
        }

        System.out.print("frame :");
        for(int j=0;j<f;j++)
            System.out.print(frame[j]+" ");
    }
}

```





```

frame :1 0 -1 -1
frame :1 0 2 -1
frame :1 3 2 -1
frame :7 3 2 -1
frame :7 3 8 -1
frame :7 1 8 -1
frame :5 1 8 -1
frame :5 1 2 -1
Page fault:9
*/

```

```

*****Optimal*****

```

```

*/

```

```

import java.util.*;
import java.io.*;

```

```

class Optimal

```

```

{

```

```

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

```

```

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int numberOfFrames, numberOfPages, flag1, flag2, flag3, i, j, k, pos = 0, max;
        int faults = 0;
        int temp[] = new int[10];

```

```

        System.out.println("Enter number of Frames: ");
        numberOfFrames = Integer.parseInt(br.readLine());
        int frame[] = new int[numberOfFrames];

```

```

        System.out.println("Enter number of Pages: ");
        numberOfPages = Integer.parseInt(br.readLine());

```

```

        int pages[] = new int[numberOfPages];
        System.out.println("Enter the pages: ");
        for(i=0; i<numberOfPages; i++)
            pages[i] = Integer.parseInt(br.readLine());

```

```

        for(i = 0; i < numberOfFrames; i++)
            frame[i] = -1;

```



```

for(i = 0; i < numberOfPages; ++i){
    flag1 = flag2 = 0;

    for(j = 0; j < numberOfFrames; ++j){
        if(frame[j] == pages[i]){
            flag1 = flag2 = 1;
            break;
        }
    }

    if(flag1 == 0){
        for(j = 0; j < numberOfFrames; ++j){
            if(frame[j] == -1){
                faults++;
                frame[j] = pages[i];
                flag2 = 1;
                break;
            }
        }
    }

    if(flag2 == 0){
        flag3 = 0;

        for(j = 0; j < numberOfFrames; ++j){
            temp[j] = -1;

            for(k = i + 1; k < numberOfPages; ++k){
                if(frame[j] == pages[k]){
                    temp[j] = k;
                    break;
                }
            }
        }

        for(j = 0; j < numberOfFrames; ++j){
            if(temp[j] == -1){
                pos = j;
                flag3 = 1;
                break;
            }
        }
    }
}

```



```

    }

    if(flag3 ==0){
        max = temp[0];
        pos = 0;

        for(j = 1; j < numberOfFrames; ++j){
            if(temp[j] > max){
                max = temp[j];
                pos = j;
            }
        }

        frame[pos] = pages[i];
        faults++;
    }

//    System.out.print();

    for(j = 0; j < numberOfFrames; ++j){
        System.out.print("\t"+ frame[j]);
    }
}

System.out.println("\n\nTotal Page Faults: "+ faults);

}

}

//7 0 1 2 0 3 0 4 2 3 0 3 2

```