grop D

Assignment: 12

Aim: Implementing page replacement algo LRU Optimal

Problem statement: To w Java prog to implement LRU and optimal algo for Page replacement

Theory

① - whenever there is a page refrence for which the page in memory that event is called page fault or page fetch or page failure situation
- In such case we have to make space in memory for this new page by replacing the existing page.
- But we cannot replace any page.
- We have to replace the page that is not used currently
- There are some algorithm for this purpose.
Last recently used (LRU)
Optimal

② LRU page replacement
- The main difference between FIFO and optimal page replacement in that the FIFO algorithm
- uses the time when the page was brought in to memory and the.
- Optimal algorithm uses the time when a page is to be used.

- If we use the recent past as an approxim of the future then we will replace the page that has not been used for the longest period of time
- This approach is called as least recently used (LRU) algorithm.
- LRU replacement associate with each page must be replaced.
- LRU chooses that has not been used for the longest period of time.
- This approach is called as least recently used (LRU) algorithm

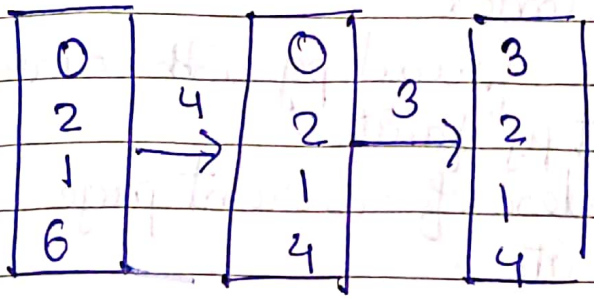
- LRU replacement associates with each page must be replace LRU choose that page that has not been used for the longest period of time.
- Now, consider refrence string 7, 0, 1, 2, 0, 3, 4 2, 3, 0, 3 with three memory frames per block
- The first three refrence case page fault that fill the empty frame

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	0	3	2	2	2	2
	+	+	+	+		+		+			+

③ Optimal Page replacement algo.

- This algo 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 ie future knowledge of refrence string is required
- often called Balady's Min Basic idea: Replace the page that will not be refrenced for the longest time
- Impossible to implement

Consider 0 2 1 6 4 0 1 0 3 1 2 1



Fault Rate = 6/12 = 0.50
With the above ref str this is the best we can hope to do.

④ Algorithm for LRU:

Let capacity be the no of pages that memory can hold. Let set be the current set of pages in memory

① Start traversing the pages
i) If set holds less pages than capacity
a) Insert page into set 1 by 1 until the size of set reaches capacity or all page req are processed
b) Simultaneously maintain the recent occured index of each page in a map called indices
ii) Else
if current page is present in set do nothing
Else
a) Find the pg in set that was least recently used we find it using index array.
We basically need to replace the pg with minimum index.
b) Replace the found pg with current pg.
c) Increment pg faults.
d) Update index of current page
② Return page faults

⑤ Algorithm for optimal

1 Start the process
2 Declare the size
3 Get the no of pages 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

Conclusion :

Thus we have studied LRU and optimal page replacement algorithm

```
//Name Ankita Bonde
// TE-A 19
//  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]+"   ");
```

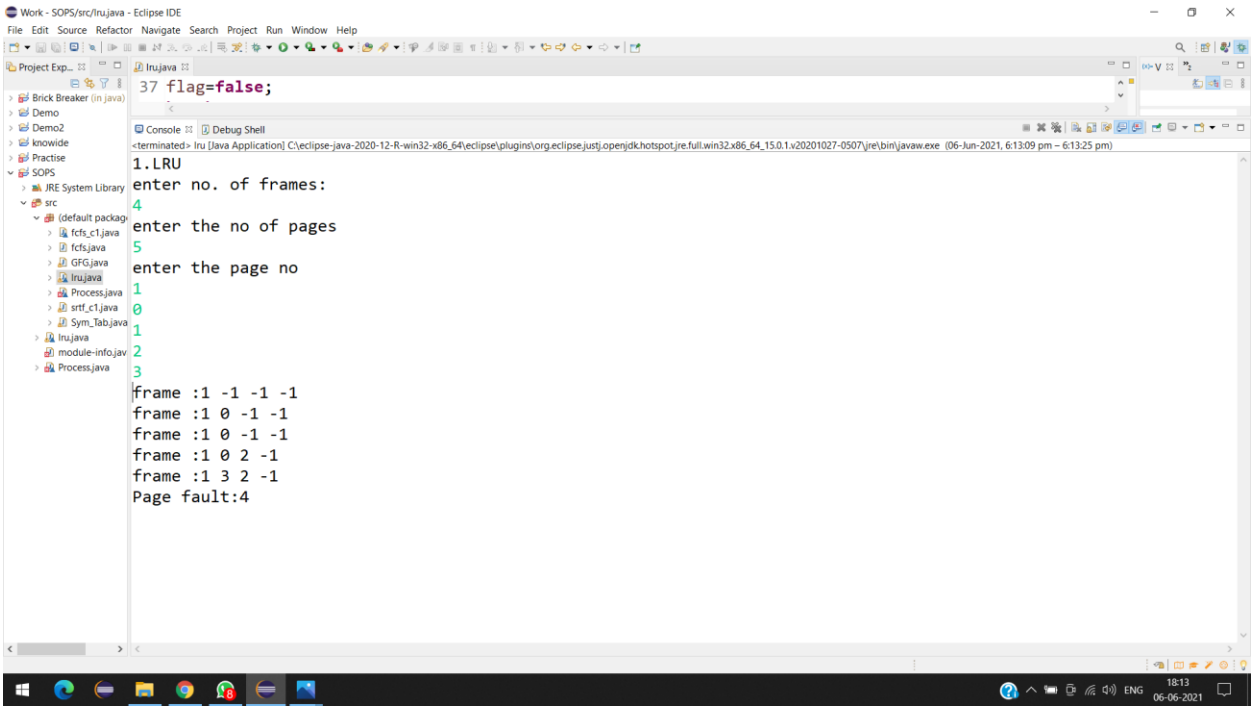
```
                    System.out.println();
                    pgf++;
                }
                else
                {
                    System.out.print("frame :");
                    for(int j=0;j<f;j++)
                    System.out.print(frame[j]+"  ");
                    System.out.println();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

            }//for

        System.out.println("Page fault:"+pgf);

}//main
}//class
```

OUTPUT:-



**********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);



        }


}
```

OUTPUT