12. Page Replacement Algorithm

Aim:- Implementing Page replacement algorithm,
1) LRU
2) Optimal

Problem statement :- To write java program to implement LRU & Optimal algorithm for page Replacement.

Theory:-
There are several algorithms to achieve.
1) Last Recently Used (LRU)
2) Optimal

1] LRU Page Replacement:-
If we use the recent past as an approximation of the future then we will replace the page that has not been used for the longest period of time. This approch is called as least recently Use (LRU) algorithm.
LRU replacement associates with each page must be replaced LRU choose that page that has not been used for the longest period of time.

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
- Often called Balady's Min Basic idea: Replace the page that will not be referenced for the longest time.
- Impossible to implement

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 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. we find it using index array. we basically need to replace the page with minimum index.
b) Replace found page with current page.
c) Increment Page faults.
d) Update index of current page

Algorithm for Optimal:

1. Start the process
2. Declare the size
3. Get the no. of pages 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 Pages with fault Pages
9. Stop the process

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
```