

## CERTIFICATE

Name of the Lab : OPERATING SYSTEMS

Name of the Student : Nagababu Thota

Student Regd. No. : 18BQ1A05K3

CLASS : III B.TECH. I SEM CSE – D

GIT HUB LINK:

<https://github.com/nagababuthota984/5K3-OS-LAB>

## INDEX

S. No.	Name of the Experiment	LAST DATE	PAGE NO
6 (a)	Simulate the FIFO page replacement algorithm.	OCT 31	
6 (b)	Simulate the Optimal page replacement algorithm.		
6 (c)	Simulate the LRU page replacement algorithm.		
6 (d)	Simulate the LFU page replacement algorithm.		

### **EXPERIMENT NO: 6 (a)**

**AIM :** Simulate the FIFO Page Replacement Algorithm

#### **DESCRIPTION :**

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

**PROGRAMMING LANGUAGE USED:** \* PYTHON3 has been used to implement this page replacement algorithm.

**LIBRARIES USED:** Since this problem does not require typical operations, None modules rather than default are used.

**SYNTAX:** No user-defined methods used in this program.

Main method is the one and only method.

#### **PROGRAM:**

```
no_of_frames = int(input("\nEnter the no of frames allotted: "))

process_list = list(map(int,input("Enter the string of page numbers with space
between each number: ").split()))

frame_queue = [-1]*no_of_frames

page_fault = 0

end = 0

print("\tInput\t\tFrame queue")

for i in process_list:

    if i not in frame_queue:

        page_fault+=1

        if end<=no_of_frames-1:
```

```

        frame_queue[end]=i

        end+=1

    else:

        del frame_queue[0]

        frame_queue.append(i)

    print("\t|",i,"|\t-->\t|", " ".join(map(str,frame_queue)),"|")
print("Page faults = ",page_fault)

```

### OUTPUT:

Enter the no of frames allotted: 4

Enter the string of page numbers with space between each number: 1 3 0 5 6 3 5  
2 1

Input	Frame queue
1   -->	1 -1 -1 -1
3   -->	1 3 -1 -1
0   -->	1 3 0 -1
5   -->	1 3 0 5
6   -->	3 0 5 6
2   -->	0 5 6 2
1   -->	5 6 2 1

Page faults = 7

### OUTPUT SCREEN SHOTS:

```

Enter the no of frames allotted: 4
Enter the string of page numbers with space between each number:
1 3 0 5 6 3 5 2 1
      Input      Frame queue
      | 1 |      --> | 1 -1 -1 -1 |
      | 3 |      --> | 1 3 -1 -1 |
      | 0 |      --> | 1 3 0 -1 |
      | 5 |      --> | 1 3 0 5 |
      | 6 |      --> | 3 0 5 6 |
      | 2 |      --> | 0 5 6 2 |
      | 1 |      --> | 5 6 2 1 |
Page faults = 7

```

### OUTPUT SCREENSHOT 2:

```

Enter the no of frames allotted: 3
Enter the string of page numbers with space between each number:
1 3 0 5 6 3 5 2 1
      Input      Frame queue
      | 1 |      --> | 1 -1 -1 |
      | 3 |      --> | 1 3 -1 |
      | 0 |      --> | 1 3 0 |
      | 5 |      --> | 3 0 5 |
      | 6 |      --> | 0 5 6 |
      | 3 |      --> | 5 6 3 |
      | 2 |      --> | 6 3 2 |
      | 1 |      --> | 3 2 1 |
Page faults = 8

```

### **EXPERIMENT NO: 6 (c)**

**AIM :** Simulate the Optimal Page Replacement Algorithm

### **DESCRIPTION :**

An optimal page-replacement algorithm has the lowest page-fault rate among all algorithms. It is also called **MIN** algorithm.

- *Optimal* or *MIN* algorithm replaces the page that will not be used for the longest period of time. In other words, it will replace the page whose next reference is far away in the *page reference string*.
- Optimal page replacement is infeasible because the virtual memory handler does not have knowledge of the future reference. It is just used to evaluate the performance with other algorithms for comparison.

**LIBRARIES USED:** Since this algorithm does not require any typical operations, None of the modules except default modules are used.

**SYNTAX:** There are no user defined methods defined in this program. Main method is the only method that has been used.

### **PROGRAM:**

```
no_of_frames=int(input("Enter no of frames:"))
result=[]
print("Enter input string seperated with spaces")
pages=[int(x) for x in input().split()]
count=0

print("\tInput\t\tFrame Queue")

x=[]
indexes=[]
duplicate=[]
```

```
for i in range(len(pages)):

    ##if page is not found increment pagefault count

    if pages[i] not in result:

        count+=1

        ##free frame is present allocate in that frame

        if len(result)<no_of_frames:

            result.append(pages[i])

        ##frames are full we have to replace

        else:

            #print("new elemnet")

            duplicate=result.copy()

            #find one element to be replaced

            for j in range(i+1,len(pages)):

                if pages[j]in duplicate and len(duplicate)>1:

                    duplicate.remove(pages[j])

            z=result.index(duplicate[0])

            result[z]=pages[i]

            #print(result)

    print("\t|",pages[i],"|\t-->\t|", " ".join(map(str,result)),"|")

print("No.of page faults are {}".format(count))
```

## OUTPUT:

### Enter no of frames:3

Enter input string seperated with spaces

1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3

Input	Frame Queue
1   -->	1
2   -->	1 2
3   -->	1 2 3
2   -->	1 2 3
1   -->	1 2 3
5   -->	1 2 5
2   -->	1 2 5
1   -->	1 2 5
6   -->	6 2 5
2   -->	6 2 5
5   -->	6 2 5
6   -->	6 2 5
3   -->	6 2 3
1   -->	6 1 3
3   -->	6 1 3
6   -->	6 1 3
1   -->	6 1 3
2   -->	2 1 3
4   -->	4 1 3
3   -->	4 1 3

No.of page faults are 9



### OUTPUT SCREENSHOT:

```
nag-1211@hp:~/Documents/3-1/OS Lab/Exp 6$ python3 optimalPR.py
Enter no of frames:3
Enter input string seperated with spaces
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Input      Frame Queue
7          7 |
0          7 0 |
1          7 0 1 |
2          2 0 1 |
0          2 0 1 |
3          2 0 3 |
0          2 0 3 |
4          2 4 3 |
2          2 4 3 |
3          2 4 3 |
0          2 0 3 |
3          2 0 3 |
2          2 0 3 |
1          2 0 1 |
2          2 0 1 |
0          2 0 1 |
1          2 0 1 |
7          7 0 1 |
0          7 0 1 |
1          7 0 1 |

No.of page faults are 9
```

### OUTPUT SCREEN SHOTS:

```
nag-1211@hp:~/Documents/3-1/OS Lab/Exp 6$ python3 optimalPR.py
Enter no of frames:3
Enter input string seperated with spaces
1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3
Input      Frame Queue
1          1 |
2          1 2 |
3          1 2 3 |
2          1 2 3 |
1          1 2 3 |
5          1 2 5 |
2          1 2 5 |
1          1 2 5 |
6          6 2 5 |
2          6 2 5 |
5          6 2 5 |
6          6 2 5 |
3          6 2 3 |
1          6 1 3 |
3          6 1 3 |
6          6 1 3 |
1          6 1 3 |
2          2 1 3 |
4          4 1 3 |
3          4 1 3 |

No.of page faults are 9
```

### **EXPERIMENT NO: 6 (c)**

**AIM :** Simulate the Least Recently Used Page Replacement Algorithm

#### **DESCRIPTION :**

Least Recently Used (LRU) page replacement policy **replaces the page that has not been used for the longest period of time**. It is one of the algorithms that were made to approximate if not better the efficiency of the optimal page replacement algorithm. The optimal algorithm assumes the entire reference string to be present at the time of allocation and replaces the page that will not be used for the longest period of time. LRU page replacement policy is based on the observation that pages that have been heavily used in the last few instructions will probably be heavily used again in the next few. Conversely, pages that have not been used for ages will probably remain unused for a long time.

**LIBRARIES USED:** Since this algorithm does not require any typical operations, None of the modules except default modules are used.

**SYNTAX:** There are no user defined methods defined in this program. Main method is the only method that has been used.

#### **PROGRAM:**

```
no_of_frames = int(input("\nEnter the no of frames allotted: "))

process_list = list(map(int,input("Enter the string of page numbers with space between each number: ").split()))

frame_queue = [-1]*no_of_frames

page_fault = 0

end = 0
```

```
print("\tInput\t\tFrame Queue")

for i in process_list:
    if i not in frame_queue:
        page_fault+=1
        if end<no_of_frames:
            frame_queue[end]=i
            end+=1
        else:
            del frame_queue[0]
            frame_queue.append(i)

    else:
        for j in range(len(frame_queue)):
            if i==frame_queue[j]:
                for k in range(j+1,end):
                    frame_queue[k-1]=frame_queue[k]
                    frame_queue[end-1]=i

        print("\t|",i,"|\t-->\t|", " ".join(map(str,frame_queue)),"|")

print("Total page faults: ",page_fault)
```

**OUTPUT:** Enter the no of frames allotted: 3

Enter the string of page numbers with space between each number: 0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4

Input		Frame Queue
0	-->	0 -1 -1
4	-->	0 4 -1
1	-->	0 4 1
4	-->	0 1 4
2	-->	1 4 2
4	-->	1 2 4
3	-->	2 4 3
4	-->	2 3 4

2	-->	3 4 2
4	-->	3 2 4
0	-->	2 4 0
4	-->	2 0 4
1	-->	0 4 1
4	-->	0 1 4
2	-->	1 4 2
4	-->	1 2 4
3	-->	2 4 3
4	-->	2 3 4

Total page faults: 9

## OUTPUT SCREEN SHOTS:

```
nag-1211@hp:~/Documents/3-1/OS Lab/Exp 6$ python3 LRUPageReplacement.py
Enter the no of frames allotted: 3
Enter the string of page numbers with space between each number: 0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4
Input      Frame Queue
| 0 | --> | 0 -1 -1 |
| 4 | --> | 0 4 -1 |
| 1 | --> | 0 4 1 |
| 4 | --> | 0 1 4 |
| 2 | --> | 1 4 2 |
| 4 | --> | 1 2 4 |
| 3 | --> | 2 4 3 |
| 4 | --> | 2 3 4 |
| 2 | --> | 3 4 2 |
| 4 | --> | 3 2 4 |
| 0 | --> | 2 4 0 |
| 4 | --> | 2 0 4 |
| 1 | --> | 0 4 1 |
| 4 | --> | 0 1 4 |
| 2 | --> | 1 4 2 |
| 4 | --> | 1 2 4 |
| 3 | --> | 2 4 3 |
| 4 | --> | 2 3 4 |
Total page faults: 9
```

## OUTPUT SCREENSHOT 2:

```

Enter the no of frames allotted: 3
Enter the string of page numbers with space between each number:
0 1 2 3 2 1 0 2 2 3 1 0
      Input      Frame Queue
      | 0 |      --> | 0 -1 -1 |
      | 1 |      --> | 0 1 -1 |
      | 2 |      --> | 0 1 2 |
      | 3 |      --> | 1 2 3 |
      | 2 |      --> | 1 3 2 |
      | 1 |      --> | 3 2 1 |
      | 0 |      --> | 2 1 0 |
      | 2 |      --> | 1 0 2 |
      | 2 |      --> | 1 0 2 |
      | 3 |      --> | 0 2 3 |
      | 1 |      --> | 2 3 1 |
      | 0 |      --> | 3 1 0 |
Total page faults: 8

```

### **EXPERIMENT NO: 6 (d)**

**AIM :** Simulate the Least Frequently Used Page Replacement Algorithm

#### **DESCRIPTION :**

In LFU Page Replacement method, the page with the minimum count is selected for replacement with the page that needs to enter into the system.

**LIBRARIES USED:** Since this algorithm does not require any typical operations, None of the modules except default modules are used.

However, the LFU technique is hardly implemented these days but this algorithm is normally combined with other algorithms which make it a hybrid algorithm, and then it is implemented.

LFU algorithm is sometimes also combined with LRU replacement algorithm, and then implemented.

**SYNTAX:** There are no user defined methods defined in this program. Main method is the only method that has been used.

#### **PROGRAM:**

```
no_of_frames = int(input("\nEnter the no of frames allotted: "))

process_list = list(map(int,input("Enter the string of page numbers with space between each number: ").split()))

frame_queue = [-1]*no_of_frames

page_fault = 0

pageUsageCount = dict()

end = 0

def getLeastUsedPage(q,dictionary):

    mini = min(dictionary.values())

    for k in dictionary.keys():
```



```

        if k in q and dictionary[k]==mini:

            return k

print("\tInput\t\t\tFrameQueue")

for i in process_list:

    #increment the no of times this page has been used.

    if i in pageUsageCount.keys():

        pageUsageCount[i]+=1

    else:

        pageUsageCount[i]=1

    #check whether the upcoming page is already in the frame queue.. if not follow below

    if i not in frame_queue:

        page_fault+=1

    #now try inserting

    #if frame queue is not full then no need to replace anything

    if end<=no_of_frames-1:

        frame_queue[end] = i

        end+=1

    #if full, get the leastTimes used page and replace it

    else:

        minValue = getLeastUsedPage(frame_queue,pageUsageCount)

        for j in range(len(frame_queue)):

            if frame_queue[j]==minValue:

                frame_queue[j] = i

print("\t|",i,"|\t-->\t|", " ".join(map(str,frame_queue)),"|")

print("Total Page Faults :",page_fault)

```

**OUTPUT:** Enter the no of frames allotted: 4

Enter the string of page numbers with space between each number: 0 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4

Input	FrameQueue
0   -->	0 -1 -1 -1
1   -->	0 1 -1 -1
4   -->	0 1 4 -1
2   -->	0 1 4 2
4   -->	0 1 4 2
3   -->	3 1 4 2
4   -->	3 1 4 2
2   -->	3 1 4 2
4   -->	3 1 4 2
0   -->	3 0 4 2
4   -->	3 0 4 2
1   -->	1 0 4 2
4   -->	1 0 4 2
2   -->	1 0 4 2
4   -->	1 0 4 2
3   -->	1 3 4 2
4   -->	1 3 4 2

Total Page Faults : 8

## OUTPUT SCREEN SHOTS:

```
nag-1211@hp:~/Documents/3-1/OS Lab/Exp 6$ python3 LFUPageReplacement.py
Enter the no of frames allotted: 4
Enter the string of page numbers with space between each number: 0 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4
Input      FrameQueue
| 0 | -->   | 0 -1 -1 -1 |
| 1 | -->   | 0 1 -1 -1 |
| 4 | -->   | 0 1 4 -1 |
| 2 | -->   | 0 1 4 2 |
| 4 | -->   | 0 1 4 2 |
| 3 | -->   | 3 1 4 2 |
| 4 | -->   | 3 1 4 2 |
| 2 | -->   | 3 1 4 2 |
| 4 | -->   | 3 1 4 2 |
| 0 | -->   | 3 0 4 2 |
| 4 | -->   | 3 0 4 2 |
| 1 | -->   | 1 0 4 2 |
| 4 | -->   | 1 0 4 2 |
| 2 | -->   | 1 0 4 2 |
| 4 | -->   | 1 0 4 2 |
| 3 | -->   | 1 3 4 2 |
| 4 | -->   | 1 3 4 2 |
Total Page Faults : 8
```



## OUTPUT SCREENSHOT 2:

```
nag-1211@hp:~/Documents/3-1/OS Lab/Exp 6$ python3 LFUPageReplacement.py
Enter the no of frames allotted: 3
Enter the string of page numbers with space between each number: 0 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4
Input      -->      FrameQueue
| 0 |      -->      | 0 -1 -1 |
| 1 |      -->      | 0 1 -1 |
| 4 |      -->      | 0 1 4 |
| 2 |      -->      | 2 1 4 |
| 4 |      -->      | 2 1 4 |
| 3 |      -->      | 2 3 4 |
| 4 |      -->      | 2 3 4 |
| 2 |      -->      | 2 3 4 |
| 4 |      -->      | 2 3 4 |
| 0 |      -->      | 2 0 4 |
| 4 |      -->      | 2 0 4 |
| 1 |      -->      | 2 0 4 |
| 4 |      -->      | 2 0 4 |
| 2 |      -->      | 2 0 4 |
| 4 |      -->      | 2 0 4 |
| 3 |      -->      | 2 3 4 |
| 4 |      -->      | 2 3 4 |
Total Page Faults : 8
```



## **VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified

Nambur, Pedakakani (M), Guntur (Dt) - 522508

### **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

B.Tech Program is Accredited by NBA