Assignment - 8

Aim : Implementation page replacement algorithm (FIFO,LRU,LFU)

Theory :

Page Fault:
A page fault happens when a running program accesses a memory page that is mapped into the virtual address space but not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of a page fault, the Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace.

FIFO (First In, First Out):

- FIFO is one of the simplest page replacement algorithms.
- It works by evicting the oldest page in memory. When a new page needs to be loaded into memory and there's no space, the page that has been in memory the longest (i.e., the one that was loaded first) is replaced.
- This algorithm is easy to implement but may not always yield the best performance. It can suffer from the "Belady's anomaly," where increasing the number of frames can sometimes increase the number of page faults.

LRU (Least Recently Used):

- LRU replaces the least recently used page when a new page needs to be brought into memory and there's no space.
- It relies on the principle that pages that have not been used recently are less likely to be used in the near future.
- LRU requires keeping track of the order in which pages are accessed, which can be done efficiently using data structures like a linked list or a priority queue.
- While LRU provides better performance compared to FIFO, its implementation might be more complex and resource-intensive.

LFU (Least Frequently Used):

- LFU replaces the least frequently used page when a new page needs to be loaded and there's no space in memory.
- It keeps track of how often each page is accessed and evicts the page with the lowest access frequency.
- LFU assumes that pages with lower access frequencies are less likely to be needed in the future.
- Implementing LFU requires maintaining a counter for each page to track its access frequency, which can consume additional memory and processing resources.

Program & Output :

1.First In, First Out :

```
def fifo(page_references_string, frames_count):
   frames = []
   page_fault = 0
   hit = 0
   index = 0
   frameMatrix = [['-' for i in range(len(page_references_string))] for
j in range(frames_count)]
   statusArray = []

   j = 0
   for page in page_references_string:
       if page not in frames:
           page_fault += 1
           if len(frames) < frames_count:
               frames.append(page)
           else:
               frames[index] = page
               index = (index + 1)%frames_count
           statusArray.append("Miss")
       else:
           hit += 1
           statusArray.append("Hit")

       for i in range(len(frames)):
           frameMatrix[i][j] = frames[i]
```

```
        j+=1


    print("Frames :")
    for row in frameMatrix:
        for item in row:
            print("|    " + str(item) + "    ", end="")
        print()
    print(statusArray)
    print(f"\nHit : {hit}")
    print(f"Page Faults : {page_fault}")



if __name__ == "__main__":
    page_references_string = [1, 3, 0, 3, 5, 6, 3]
    frames_count = 3
    fifo(page_references_string, frames_count)
```

```
➜  OS git:(master) python3 fifo.py
Frames :
|    1    |    1    |    1    |    1    |    5    |    5    |    5
|    -    |    3    |    3    |    3    |    3    |    6    |    6
|    -    |    -    |    0    |    0    |    0    |    0    |    3
['Miss', 'Miss', 'Miss', 'Hit', 'Miss', 'Miss', 'Miss']

Hit : 1
Page Faults : 6
➜  OS git:(master)
```

2. Least Recently Used :

```
def lru(page_references_string, frames_count):
    frames = []
    page_fault = 0
    hit = 0
    frameMatrix = [['-' for _ in range(len(page_references_string))] for
_ in range(frames_count)]
    statusArray = []

    for j, page in enumerate(page_references_string):
        if page not in frames:
```

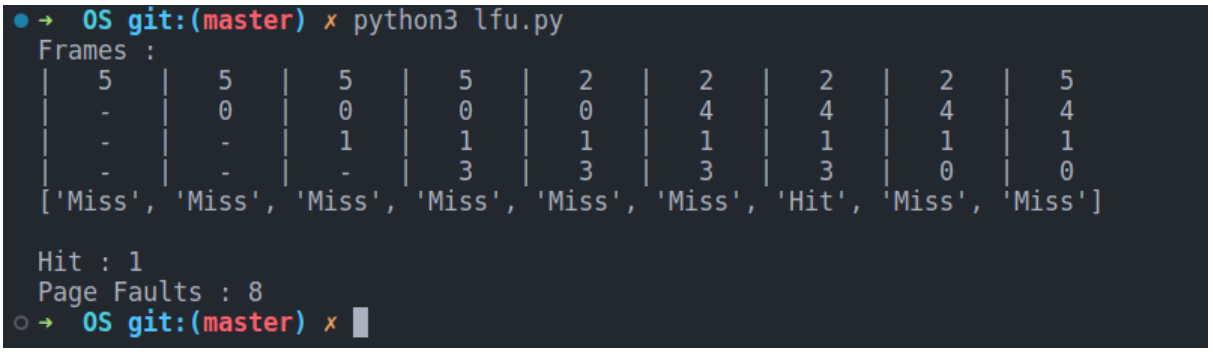
```
            page_fault += 1
            if len(frames) < frames_count:
                frames.append(page)
            else:
                least_recently_used = min(range(frames_count),
key=lambda x: page_references_string.index(frames[x]))
                frames[least_recently_used] = page
            statusArray.append("Miss")
        else:
            hit += 1
            statusArray.append("Hit")

        for i, frame in enumerate(frames):
            frameMatrix[i][j] = frame

    print("Frames :")
    for row in frameMatrix:
        for item in row:
            print("|    " + str(item) + "    ", end="")
        print()
    print(statusArray)
    print(f"\nHit : {hit}")
    print(f"Page Faults : {page_fault}")

if __name__ == "__main__":
    page_references_string = [1, 3, 0, 3, 5, 6, 3]
    frames_count = 3
    lru(page_references_string, frames_count)
```

```
● → OS git:(master) ✗ python3 lru.py
  Frames :
  |   1   |   1   |   1   |   1   |   1   |   6   |   6   |   6   |   6   |   6   |   6   |   6   |   6
  |   -   |   3   |   3   |   3   |   3   |   3   |   3   |   4   |   4   |   4   |   4   |   4   |   4
  |   -   |   -   |   0   |   0   |   0   |   0   |   0   |   0   |   1   |   3   |   3   |   3   |   3
  |   -   |   -   |   -   |   -   |   5   |   5   |   5   |   5   |   5   |   5   |   5   |   5   |   5
  ['Miss', 'Miss', 'Miss', 'Hit', 'Miss', 'Miss', 'Hit', 'Miss', 'Miss', 'Miss', 'Hit', 'Hit', 'Hit']

  Hit : 5
  Page Faults : 8
○ → OS git:(master) ✗
```

3) Least Frequently Used :

```
from collections import defaultdict
```

```
def lfu(page_references_string, frames_count):
    page_frequency = defaultdict(int)
    last_used_time = {}
    frames = []
    page_fault = 0
    hit = 0
    frameMatrix = [['-' for _ in range(len(page_references_string))] for
_ in range(frames_count)]
    statusArray = []

    for j, page in enumerate(page_references_string):
        page_frequency[page] += 1
        last_used_time[page] = j

        if page not in frames:
            page_fault += 1
            if len(frames) < frames_count:
                frames.append(page)
            else:
                least_frequent = min(frames, key=lambda x:
(page_frequency[x], last_used_time[x]))
                frames[frames.index(least_frequent)] = page
            statusArray.append("Miss")
        else:
            hit += 1
            statusArray.append("Hit")

        for i, frame in enumerate(frames):
            frameMatrix[i][j] = frame

    print("Frames :")
    for row in frameMatrix:
        for item in row:
            print("|   " + str(item) + "   ", end="")
        print()
    print(statusArray)
    print(f"\nHit : {hit}")
    print(f"Page Faults : {page_fault}")

if __name__ == "__main__":
    page_references_string = [5, 0, 1, 3, 2, 4, 1, 0, 5]
    frames_count = 4
    lfu(page_references_string, frames_count)
```



Conclusion : Here, we studied various page replacements algorithms like FIFO, LRU, LFU and implemented them in python.