# Name: Ebrahim Hirani UID: 2018130015

**Class: SE Comps Batch: B**

**Roll Number: 18**

**COA Lab**

**Experiment No. 4**

**Aim:** To implement Page Replacement Algorithms

## PAGE REPLACEMENT ALGORITHMS

* In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in.
* 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.
* Page Hit – When we want to load the page on the memory, and the page is already available on memory, then it is called page hit.
* Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, 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. The target for all algorithms is to reduce the number of page faults.

## First In First Out (FIFO)

* + 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.
  + Example:
    - Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames.Find number of page faults.
    - Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> 3 Page Faults. when 3 comes, it is already in memory so —> 0 Page Faults.
    - Then 5 comes, it is not available in memory so it replaces the oldest page slot

i.e 1. —>1 Page Fault. 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>1 Page Fault.

* + - Finally when 3 come it is not avilable so it replaces 0 1 page fault

## Optimal Page replacement

* + In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.
  + Example:
    - Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame. Find number of page fault.
    - Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots

—> 4 Page faults 0 is already there so —> 0 Page fault.

* + - when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>1 Page fault.
    - 0 is already there so —> 0 Page fault.. 4 will takes place of 1 —> 1 Page Fault.
    - Now for the further page reference string —> 0 Page fault because they are already available in the memory.
    - Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

## Least Recently Used

* + In this algorithm page will be replaced which is least recently used.
  + Example:
    - Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames. Find number of page faults.
    - Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots

—> 4 Page faults 0 is already their so —> 0 Page fault.

* + - when 3 came it will take the place of 7 because it is least recently used —>1 Page fault 0 is already in memory so —> 0 Page fault.
    - 4 will takes place of 1 —> 1 Page Fault
    - Now for the further page reference string —> 0 Page fault because they are already available in the memory.

## IMPLEMENTATION OF ALGORITHMS

**Code:**

*def* FIFO(*pages*, *frameSize*): currPages = []

allPages = [] hits = [] pageHits = 0

for page in pages:

if page not in currPages:

if len(currPages) == frameSize: del currPages[0]

currPages.append(page) hits.append(' ')

else:

pageHits += 1 hits.append('\* ')

allPages.append(currPages[:]) for i in range(frameSize):

for pageset in allPages:

if i < len(pageset): print(*str*(pageset[i]) + '|', *end*='')

else:

print(' |', *end*='')

print()

for hit in hits: print(hit, *end*='')

print("\nPage Hits = " + *str*(pageHits))

print("Page Faults = " + *str*(len(pages) - pageHits)) print("Fault Rate = " + *str*(len(pages)//pageHits))

*def* LRU(*pages*, *frameSize*): currPages = [] allPages = []

hits = [] pageHits = 0

for page in pages:

if page not in currPages:

if len(currPages) == frameSize: del currPages[0]

hits.append(' ') else:

pageHits += 1 hits.append('\* ')

del currPages[currPages.index(page)] currPages.append(page) allPages.append(currPages[:])

for i in range(frameSize): for pageset in allPages:

if i < len(pageset): print(*str*(pageset[i]) + '|', *end*='')

else:

print(' |', *end*='')

print()

for hit in hits: print(hit, *end*='')

print("\nPage Hits = " + *str*(pageHits))

print("Page Faults = " + *str*(len(pages) - pageHits)) print("Fault Rate = " + *str*(len(pages)//pageHits))

*def* Optimal(*pages*, *frameSize*): pageHits = 0

allPages = [] hits = [] currPages = []

for i,page in enumerate(pages): if page not in currPages:

if len(currPages) == frameSize:

max = -1 deleted = False try:

for cpage in currPages:

curri = pages.index(cpage,i+1)

max = curri if curri > max else max except(*ValueError*):

del currPages[currPages.index(cpage)] else:

del currPages[currPages.index(pages[max])] currPages.append(page)

hits.append(' ') else:

pageHits += 1 hits.append('\* ')

allPages.append(currPages[:]) for i in range(frameSize):

for pageset in allPages: if i < len(pageset):

print(*str*(pageset[i]) + '|', *end*='') else:

print(' |', *end*='')

print()

for hit in hits: print(hit, *end*='')

print("\nPage Hits = " + *str*(pageHits))

print("Page Faults = " + *str*(len(pages) - pageHits)) print("Fault Rate = " + *str*(len(pages)/pageHits))

if name == ' main ':

choice = *int*(input("Enter 1 for custom input and 0 for default: ")) pages = [1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6]

if choice == 1:

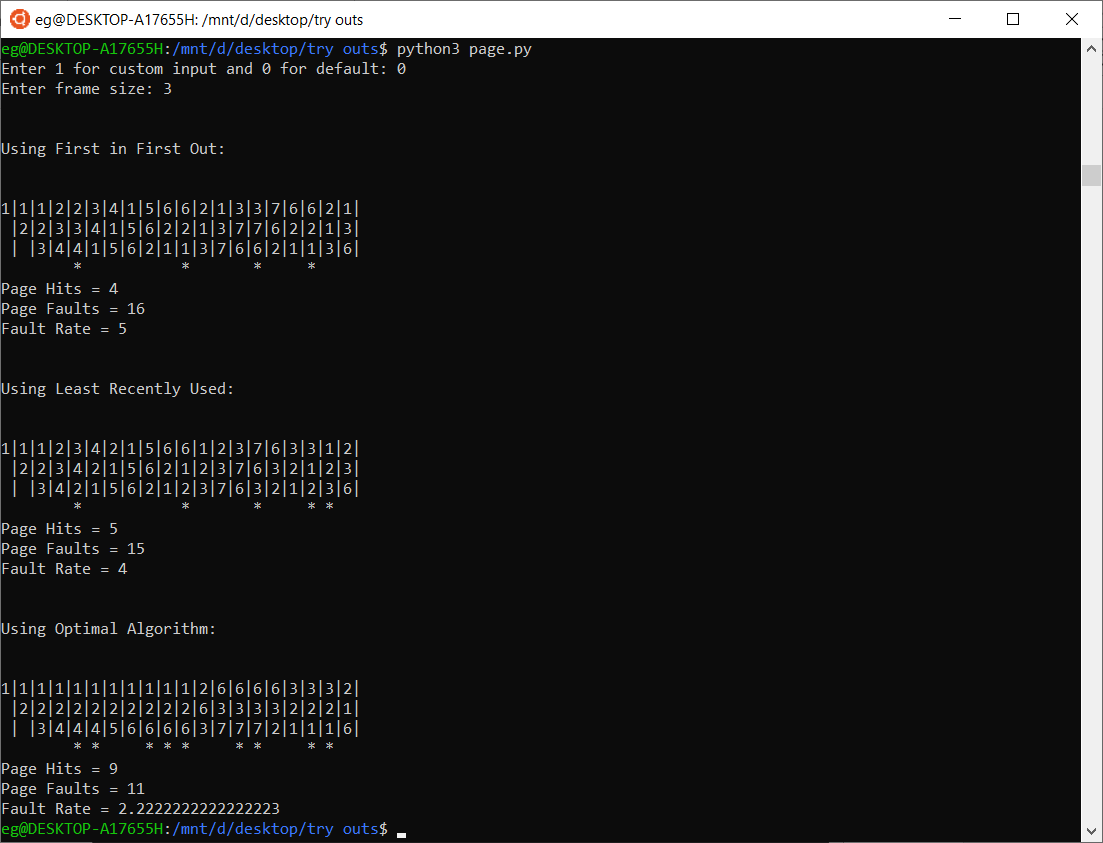
pages = *list*(map(*int*,input("Enter space seperated integers for pages:\ n").strip().split()))

n = *int*(input("Enter frame size: ")) print("\n\nUsing First in First Out:\n\n") FIFO(pages,n)

print("\n\nUsing Least Recently Used:\n\n") LRU(pages,n)

print("\n\nUsing Optimal Algorithm:\n\n") Optimal(pages,n)

## Output:



# CONCLUSION

* Learnt the Page Replacement Algorithms.
* Implemented the algorithms using Python and displayed the table, page hits, page faults and fault rate.