Page Replacement Algorithms in Operating Systems

Operating system uses paging for memory management. The page replacement algorithm decides which memory page is to be replaced.

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.

Operating System might have to replace one of the existing pages with the newly needed page. So, we need proper mechanism to replace the pages. Different page replacement algorithms suggest different ways to decide which page to replace. Well popular page replacement algorithms are-

1. First In First Out (FIFO) page replacement

   In this algorithm, the page which is assigned the frame first will be replaced first.

   Input: Page reference, number of frame
   Output: Number of page fault

```
def FIFO(PageReference, NumberOfFrame)
```

is user defined functions

Step 1: Initialised frame value by infinite `(Frame = [np.inf for i in range(NumberOfFrame)])`.

Step 2: Set number of page fault is zero `(PageFault = 0)`.

Step 3: Picked up one page from page reference queue `(for i in range(len(PageReference)))`.

Sept 4: Check the page is present in the Frame or not `(if PageReference[i] not in Frame:)`.

Step 5: If page is not present in Frame, then replace the page by the page that came first, increment the number of page fault by 1, and update the frame pointer `(PageFault += 1, Frame[j] = PageReference[i], j = (j + 1) % NumberOfFrame)`.

Step 6: Go to step 3 until page reference queue is empty.

2. Optimal Page replacement (OPR).

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Input: Page reference, number of frame
Output: Number of page fault

`def OPR(PageReference, NumberOfFrame)` is user defined functions

Step 1: Initialised frame value by infinite `(Frame = [np.inf for i in range(NumberOfFrame)])`.

Step 2: Set number of page fault is zero `(PageFault = 0)`.

Step 3: Picked up one page from page reference queue `(for i in range(len(PageReference)))`.

Sept 4: Check the page is present in the Frame or not `(if PageReference[i] not in Frame:)`.

Step 5: If page is not present in Frame, then execute step 6 to step 8

Step 6: If frame has empty position then set Index = position `(index = Frame.index(np.inf)`.

Step7: Otherwise, calculate which page has least change to enter again `(Position = -1, for k in range(NumberOfFrame), if Frame[k] in RestList, P = RestList.index(Frame[k]), if P>Position, Position = P, index = k)`.

Step 8: Replace the page by the page that placed into selected position, increment the number of page fault by 1, and update the frame pointer `(if index != -1: Frame[index] = PageReference[i], else: Frame[j] = PageReference[i] j = (j + 1) % NumberOfFrame, PageFault += 1)`.

Step 9: Go to step 3 until page reference queue is empty.

3. Least Recently Used (LRU) Page replacement.

In this algorithm, page will be replaced which is least recently used.

Input: Page reference, number of frame
Output: Number of page fault

`def LRU(PageReference, NumberOfFrame)` is user defined functions

Step 1: Initialised frame value by infinite and create an index queue that decided wh ich index should replace next. `(Frame = [np.inf for i in range( NumberOfFr ame)], IndexList = [i for i in range(NumberOfFrame)] ).`

Step 2: Set number of page fault is zero `(PageFault = 0).`

Step 3: Picked up one page from page reference queue `(for i in range(len(P ageReference))).`

Step 4: If frame has empty position, then replace the page by the empty position an d set the position is current used position and increment the page fault count by 1. `(index = Frame.index(np.inf), Frame[index] = PageReference[i], IndexList.pop(0), IndexList.append(index), PageFault += 1).`

Sept 4: Check the page is present in the Frame or not `(if PageReference[i] n ot in Frame).`

Step 5: If page is not present in Frame, then replace the page by the page present into the first position of index queue, after that set the position is current used position, and incre ment the number of page fault by 1, and update the frame pointer `(PageReference[i] no t in Frame: Pos = IndexList[0] Frame[Pos] = PageReference[i] IndexList. pop(0) IndexList.append(Pos) PageFault += 1).`

Step 6: If page is present in Frame, then set the position as current used position `(i ndex = Frame.index(PageReference[i]) index_p = IndexList.index(index) I ndexList.pop(index_p) IndexList.append(index)).`

Step 7: Go to step 3 until page reference queue is empty.