**Question:** When does a page fault occur? Explain various page replacement strategies/algorithms.

**Answer:** In demand paging memory management technique, if a page demanded for execution is not present in main memory, then a page fault occurs. To load the page in demand into main memory, a free page frame is searched in main memory and allocated. If no page frame is free, memory manager has to free a frame by swapping its contents to secondary storage and thus make room for the required page. To swap pages, many schemes or strategies are used.

**Various page replacement Strategies / Algorithms**

1. **The Optimal Page Replacement Algorithm** - This algorithm replaces the page that will not be used for the longest period of time. The moment the page fault occurs, some set of pages are in memory. One of these page will be referenced on the very next instruction. Other pages may not be referenced until 10,100 or perhaps 1000 instructions. This information can be stored with each page in the PMT(Page Map Table).

<table>
<thead>
<tr>
<th>P#</th>
<th>base</th>
<th>Offset</th>
<th>MISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>NIL</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

   The optimal page algorithm simply removes the page with the highest number of such instructions implying that it will be needed in the most distant future. this algorithm was introduced long back and is difficult to implement because it requires future knowledge of the program behavior. However it is possible to implement optimal page replacement on the second run by using the page reference information collected on the first run.

2. **NRU(Not Recently Used) Page Replacement Algorithm** - This algorithm requires that each page have two additional status bits ‘R’ and ‘M’ called reference bit and change bit respectively. The reference bit(R) is automatically set to 1 whenever the page is referenced. The change bit (M) is set to 1 whenever the page is modified. These bits are stored in the PMT and are updated on every memory reference. When a page fault occurs, the memory manager inspects all the pages and divides them into 4 classes based on R and M bits.

   - **Class 1**: (0,0) - neither recently used nor modified - the best page to replace.
   - **Class 2**: (0,1) - not recently used but modified - the page will need to be written out before replacement.
   - **Class 3**: (1,0) - recently used but clean - probably will be used again soon.
   - **Class 4**: (1,1) - recently used and modified - probably will be used again, and write out will be needed before replacing it.

   This algorithm removes a page at random from the lowest numbered non-empty class.

   **Advantages:**

   - It is easy to understand.
   - It is efficient to implement.

3. **FIFO (First in First out) Page Replacement Algorithm** - It is one of the simplest page replacement algorithm.
The oldest page, which has spent the longest time in memory is chosen and replaced. This algorithm is implemented with the help of FIFO queue to hold the pages in memory. A page is inserted at the rear end of the queue and is replaced at the front of the queue.

The reference string is 5, 4, 3, 2, 5, 4, 6, 5, 4, 3, 2, 6 and there are 3 frames empty. The first 3 reference (5, 4, 3) cause page faults and are brought into empty frames. The next reference (2) replaces page 5 because page 5 was loaded first and so on. After 7 page faults, the next reference is 5 and 5 is already in memory so no page fault for this reference. Similarly for next reference 4. The + marks shows incoming of a page while circle shows the page chosen for removal.

**Advantages**

- FIFO is easy to understand.
- It is very easy to implement.

**Disadvantage**

- Not always good at performance. It may replace an active page to bring a new one and thus may cause a page fault of that page immediately.
- Another unexpected side effect is the FIFO anomaly or Belady’s anomaly. This anomaly says that the page fault rate may increase as the number of allocated page frames increases.

i.e. The following figure presents the same page trace but with a larger memory. Here number of page frame is 4.

**4. LRU(Least Recently Used) Algorithm** - The Least Recently used (LRU) algorithm replaces the page that has not been used for the longest period of time. It is based on the observation that pages that have not been used for long time will probably remain unused for the longest time and are to be replaced.
Initially, 3 page frames are empty. The first 3 references (7, 0, 1) cause page faults and are brought into these empty frames. The next reference (2) replaces page 7. Since next page reference (0) is already in memory, there is no page fault. Now, for the next reference (3), LRU replacement sees that, of the three frames in memory, page 1 was used least recently, and thus is replaced. And thus the process continues.

**Advantages**

- LRU page replacement algorithm is quiet efficient.
- It does not suffer from Belady's Anomaly.

**Disadvantages**

- Its implementation is not very easy.
- Its implementation may require substantial hardware assistance.