**First In First Out (FIFO) Algorithm**

**FIFO** algorithm is the simplest of all the page replacement algorithms. In this, we maintain a queue of all the pages that are in the memory currently. The oldest page in the memory is at the front-end of the queue and the most recent page is at the back or rear-end of the queue.

Whenever a page fault occurs, the operating system looks at the front-end of the queue to know the page to be replaced by the newly requested page. It also adds this newly requested page at the rear-end and removes the oldest page from the front-end of the queue.

**Example:** Consider the page reference string as **3, 1, 2, 1, 6, 5, 1, 3** with 3-page frames. Let’s try to find the number of page faults:

Initially, all of the slots are empty so page faults occur at **3,1,2**.

**Page faults = 3**

• When page 1 comes, it is in the memory so no page fault occurs.

**Page faults = 3**

• When page 6 comes, it is not present and a page fault occurs. Since there are no empty slots, we remove the front of the queue, i.e 3.

**Page faults = 4**

• When page 5 comes, it is also not present and hence a page fault occurs. The front of the queue i.e 1 is removed.

**Page faults = 5**

• When page 1 comes, it is not found in memory and again a page fault occurs. The front of the queue i.e. 2 is removed.

**Page faults = 6**

• When page 3 comes, it is again not found in memory, a page fault occurs, and page 6 is removed being on top of the queue

Total page faults = 7

**Belady's anomaly:** Generally, if we increase the number of frames in the memory, the number of page faults should decrease due to obvious reasons. Belady’s anomaly refers to the phenomena where increasing the number of frames in memory, increases the page faults as well.

**Advantages :**

• Simple to understand and implement

• Does not cause more overhead Disadvantages

• Poor performance

• Doesn’t use the frequency of the last used time and just simply replaces the oldest page.

• Suffers from Belady’s anomaly.