MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Dr. Vishwanath Karad

MIT WORLD PEACE UNIVERSITY | PUNE

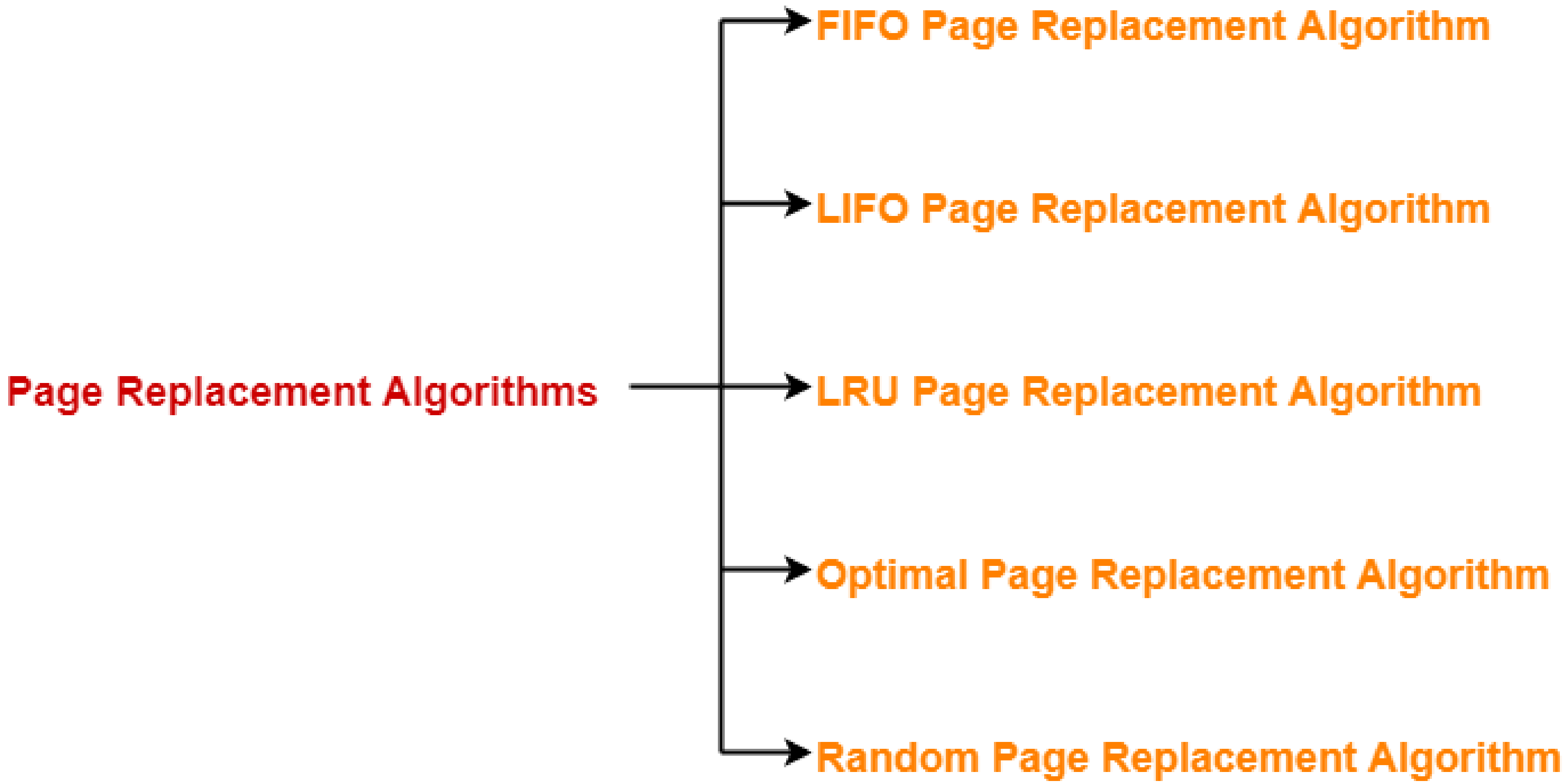
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Page Replacement Algorithms for Memory Management

Assignment No 8- Page Replacement Algorithm

- Write a program to simulate LRU page replacement algorithm.

Page Replacement Algorithms


Page Replacement Algorithms
FIFO Page Replacement Algorithm
LIFO Page Replacement Algorithm
LRU Page Replacement Algorithm
Optimal Page Replacement Algorithm
Random Page Replacement Algorithm

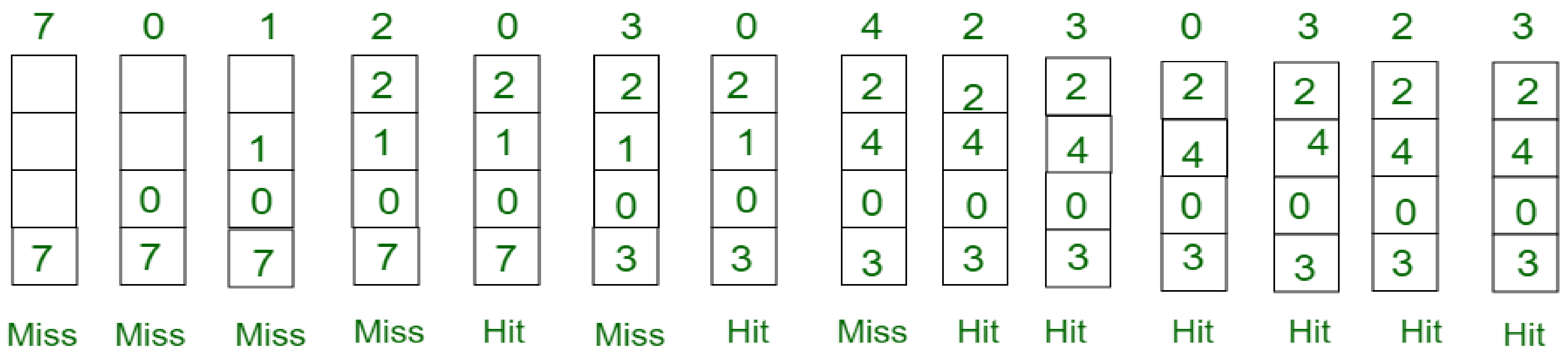
LRU Page Replacement Algorithm

- In **L**east **R**ecently **U**sed (LRU) algorithm is a Greedy algorithm where the page to be replaced is least recently used.
- The opposite of **FIFO** is LIFO, in which the last data entered is the first to be removed
- It works on the idea that the more recently used blocks are more likely to be referenced again.
- When the page requested by the program is not present in the RAM, page fault occurs and then if the page frame is full then we have to remove the page that has not been in use for the longest period of time.
- It works on the idea that the more recently used blocks are more likely to be referenced again. LRU is the most frequently used algorithm as it gives less number of page faults when compared to the other algorithms.

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.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3

No. of Page frame - 4



Total Page Fault = 6

Here LRU has same number of page fault as optimal but it may differ according to question.

LRU Page Replacement Algorithm

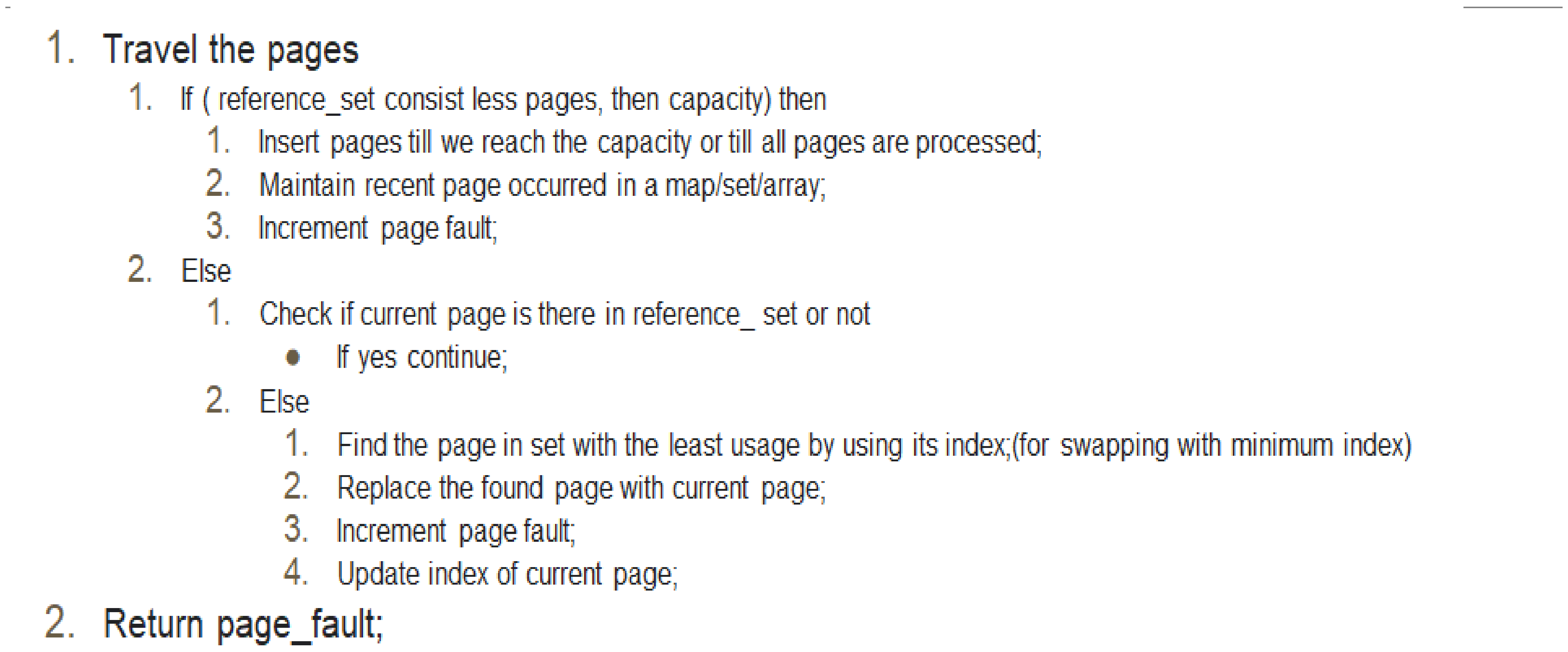
Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults
0 is already their so —> 0 Page fault.

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.

Pseudo Code - LRU



```
1. Travel the pages
    1. If ( reference_set consist less pages, then capacity) then
        1. Insert pages till we reach the capacity or till all pages are processed;
        2. Maintain recent page occurred in a map/set/array;
        3. Increment page fault;
    2. Else
        1. Check if current page is there in reference_ set or not
            • If yes continue;
        2. Else
            1. Find the page in set with the least usage by using its index;(for swapping with minimum index)
            2. Replace the found page with current page;
            3. Increment page fault;
            4. Update index of current page;
2. Return page_fault;
```

FIFO vs LRU

FIFO	**LRU**
Number of page fault is more than LRU and optimal.	Number of page fault is more than optimal and less than FIFO.
Suffer from belady's anomaly.	Does not suffer from belady's anomaly.
It is little overhead as there is no need to keep line of the pages.	It is more overhead because you have to keep line of time when the pages were used.
Think of **FIFO** as cars going through a tunnel. The first car to go **in the** tunnel will be the first one to go out the other side	Think of the **LRU** cache as cleaning out the garage. You will throw away items that you have not used for a long time, and keep the ones that you use frequently.