Page Replacement Policy

P. K. Roy
Asst. Professor
Siliguri Institute of Technology

Major Concern (1)

- Memory Capacity: Limited

- Page Fault: The page fault rate must not increase with an increase in the memory allocation in a program.

- Demand Paging: What actually happens when a page fault occurs?

- Locality of Reference: The replacement policy must not remove a page which may be referenced in the immediate future.

Major Concern (2)

Page Reference String: Sequence of page numbers, arranged in chronological order in which the pages are referenced during execution.

Logical Timestamp: Advanced only when the program is in running state.

Page Reference String

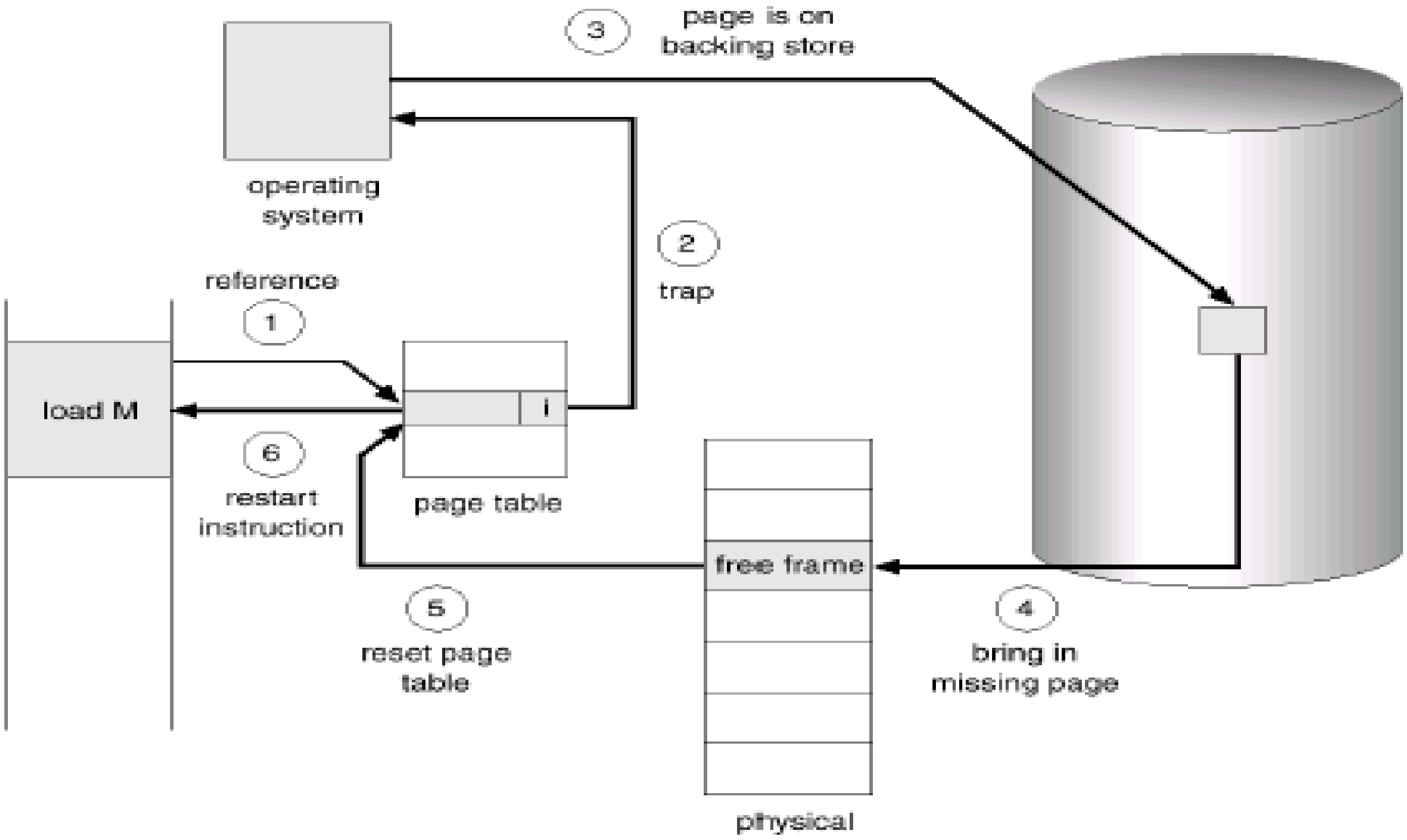
1	1	2	1	1	1	3	1	3	1
t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9

Reference Time String

Functions Involved

- Which page is to be removed from the memory.
- Page-out(Swap-out) operation
- Page-in(Swap-in) operation

Handling Page Fault (1)



Handling Page Fault (2)

1- Check whether the reference page was valid or in invalid memory access.

2- If the reference was invalid, the process terminated.

3- The operating system will find a free frame (by taking one from the free frame list, for example).

4- The scheduling a disc operation to read the desired page into the newly allocated frame.

5- The internal table will modify and the page table to indicate that page is now in memory.

6- Restart the instruction that was interrupted by the illegal address trap. The process can now access the page as though it had always been in memory.

Page Replacement Algorithms

1- Find the location of the desired page on the backing store.

2- Find a free frame:

a. If there is a free frame uses it,

b. Otherwise use a page replacement algorithm to select a victim frame

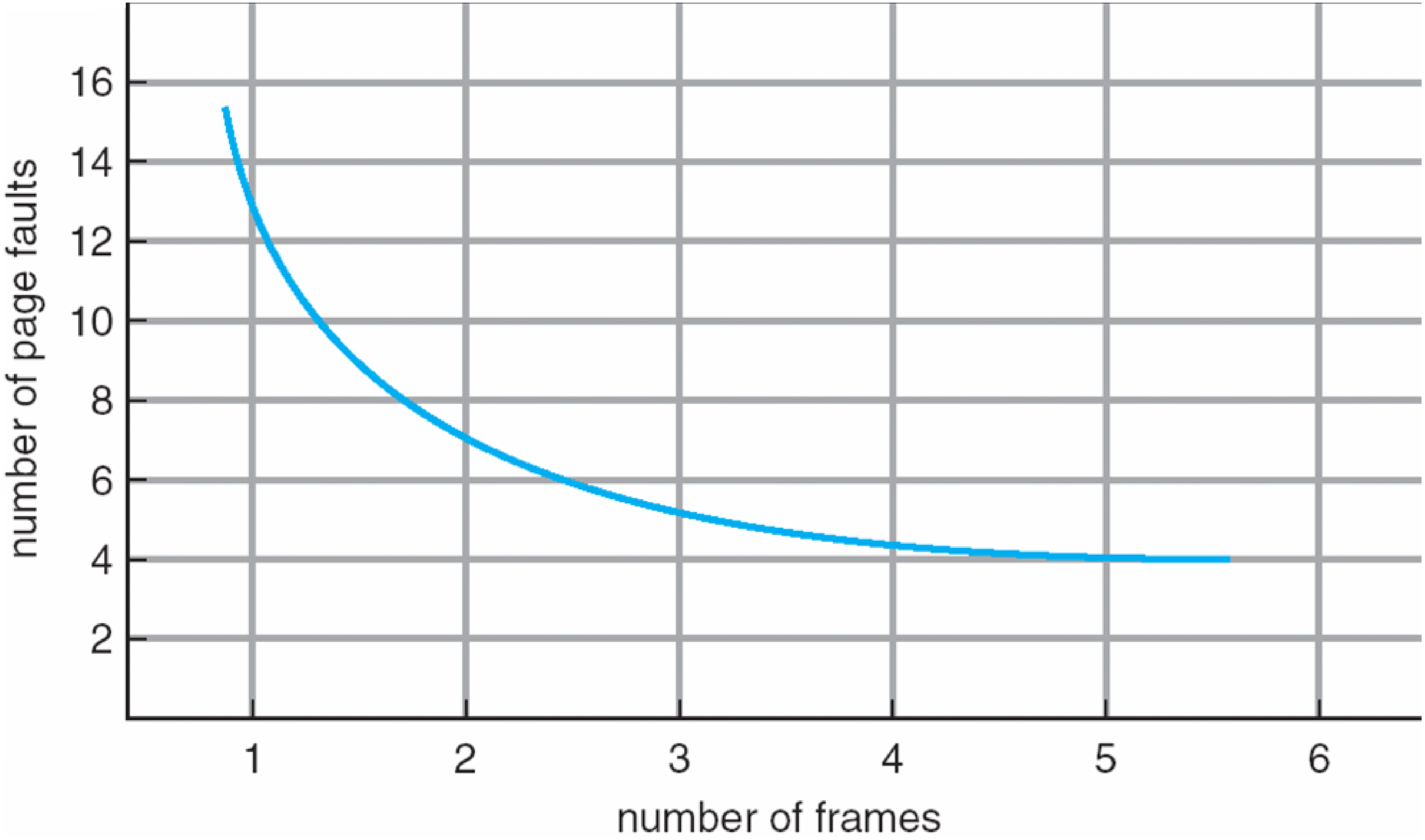
c. Write the victim page to the backing store and change the page and frame tables accordingly.

3- Read the desired page into the free frame changing the page and frame tables

4- Restart the user process

There are many different page replacement algorithms probably every operating system has its own unique replacement scheme. How do we select particular replacement algorithms? In general, we want the one with the lowest page fault rate.

Graph of Page Faults vs. the Number of Frames



Main Page Replacement Algorithms

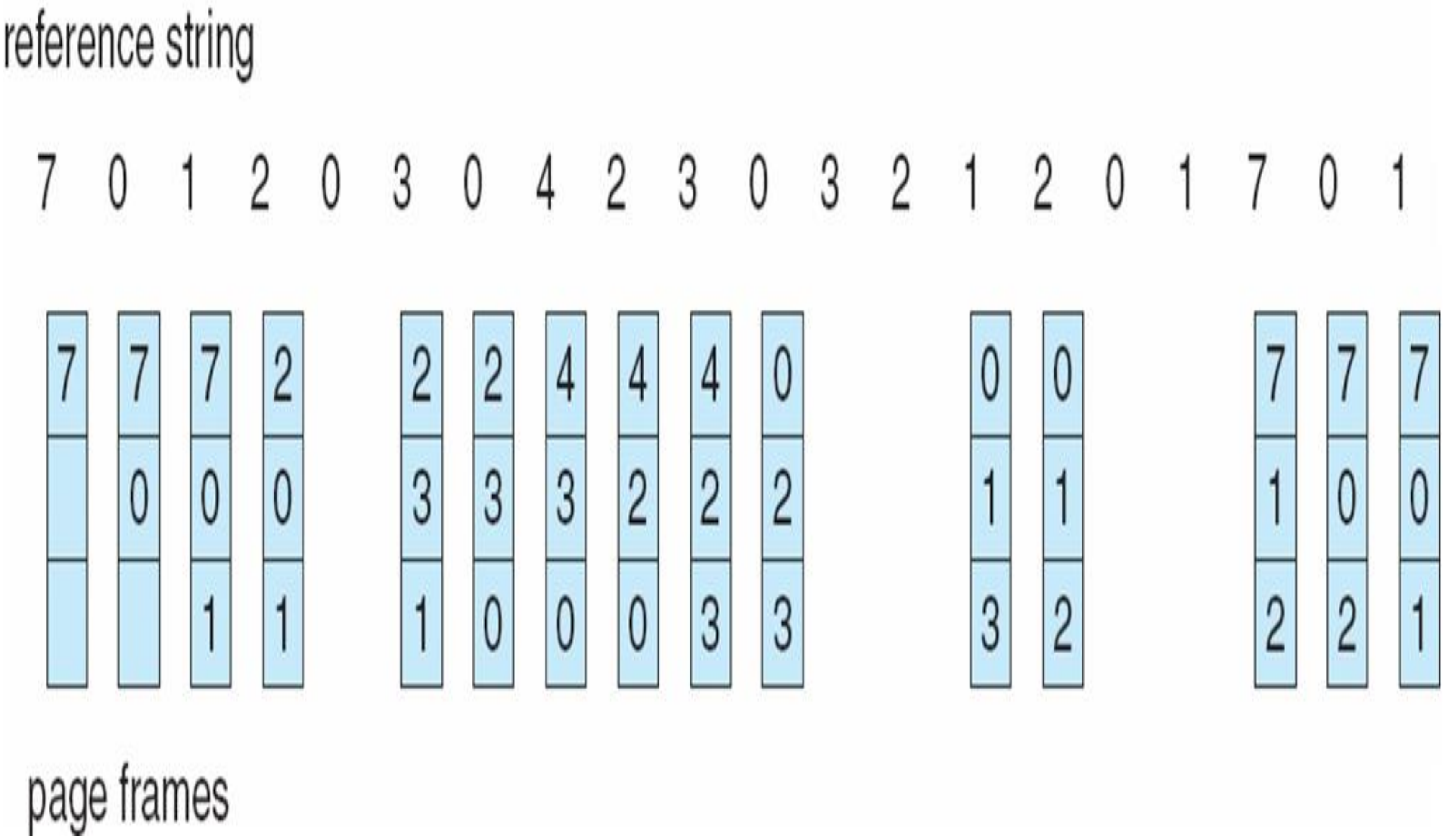
- First In First Out (FIFO)
- Optimal (OPT)
- Least Recently Used (LRU)

First In First Out (FIFO)

Treats page frames allocated to a process as a circular buffer:

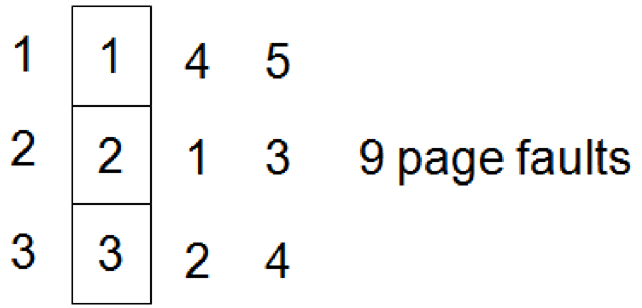
- When the buffer is full, the oldest page is replaced. Hence first-in, first-out:
  - A frequently used page is often the oldest, so it will be repeatedly paged out by FIFO.
- Simple to implement:
  - requires only a pointer that circles through the page frames of the process.

First In First Out (FIFO)



First In First Out (FIFO)

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process):

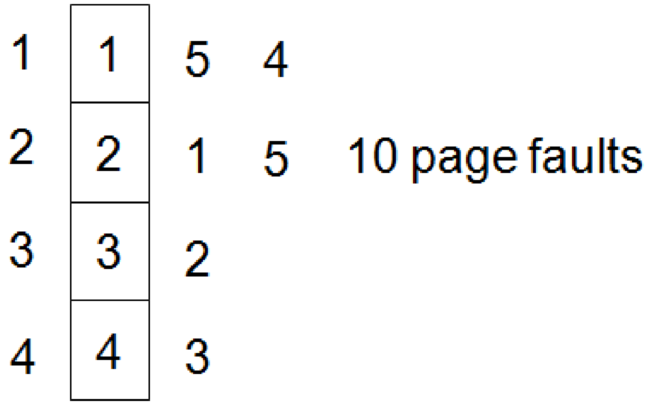


1 1 4 5

2 2 1 3 9 page faults

3 3 2 4

- 4 frames:



1 1 5 4

2 2 1 5 10 page faults

3 3 2

4 4 3

- FIFO Replacement manifests Belady's Anomaly:
  - more frames ⇒ more page faults

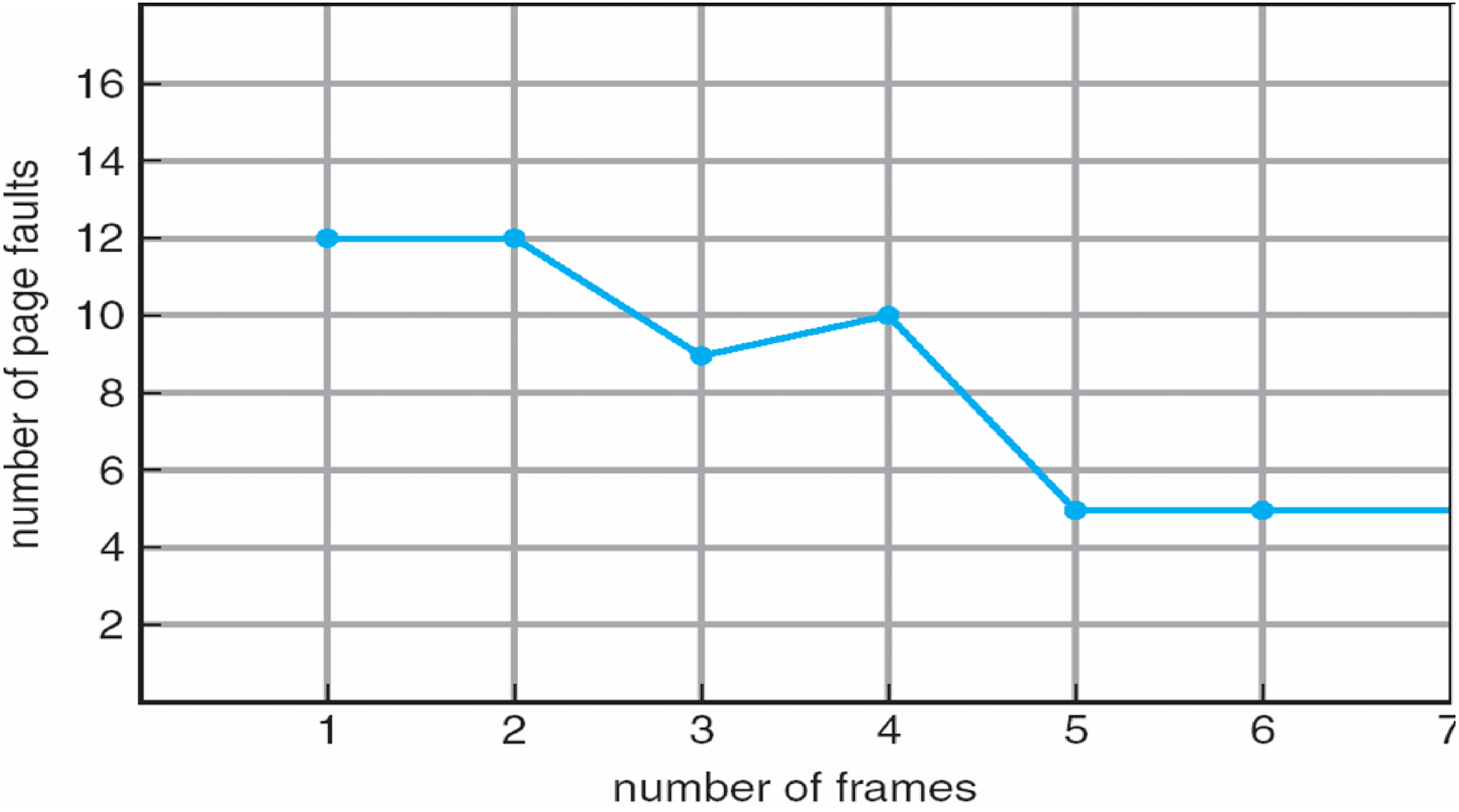
Belady’s Anomaly

- In computer storage, Belady's **anomaly** is the name given to the phenomenon in which increasing the number of page frames results in an increase in the number of page faults for certain memory access patterns.

- This phenomenon is commonly experienced when using the First in First Out (FIFO) page replacement algorithm.

FIFO Illustrating Belady’s Anomaly

This algorithm suffers from Belady's anomaly. The fault rate might actually increase when the algorithm is given more memory.



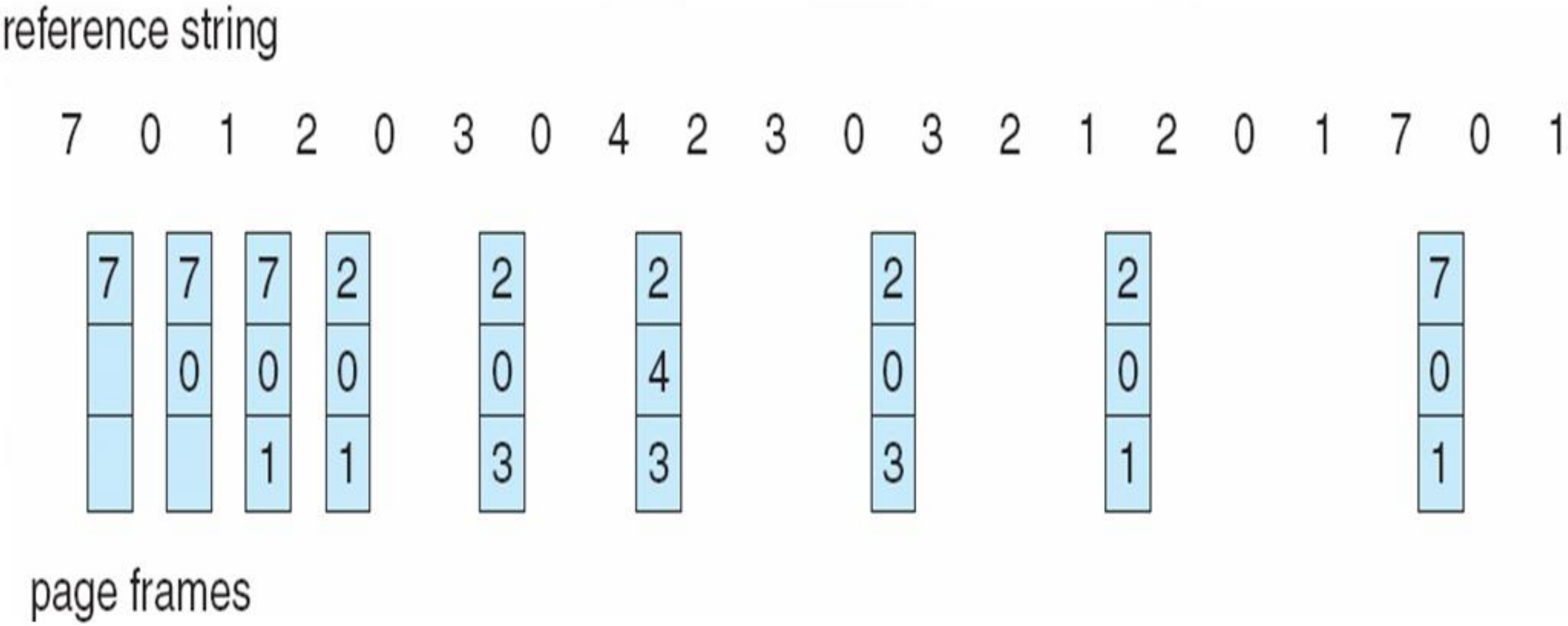
Optimal Page Replacement

- The Optimal policy selects for replacement the page that will not be used for longest period of time.

- Has the lowest page fault rate for any page reference stream.

- Impossible to implement (need to know the future) but serves as a standard to compare with the other algorithms we shall study.

Optimal Page Replacement



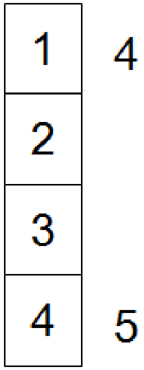
reference string

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		2			2			2				7		
	0	0	0		0		4			0			0				0		
		1	1		3		3			3			1				1		

page frames

Optimal Page Replacement

- Reference string : 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 4 frames example



1 4

2

3

4 5

6 page faults

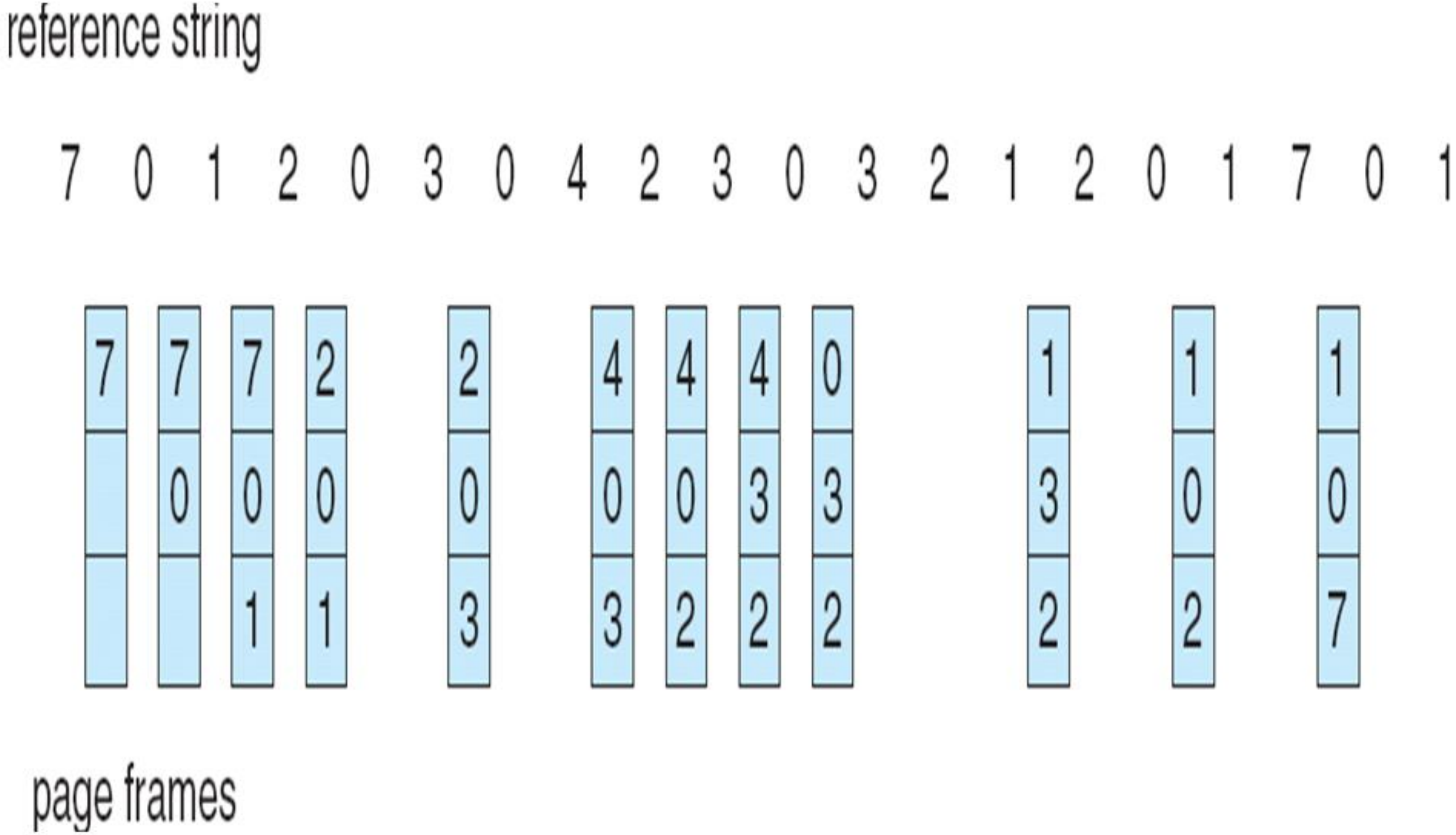
- How do you know future use? You don’t!
- Used for measuring how well your algorithm performs.

The LRU Policy

Replaces the page that has not been referenced for the longest time:

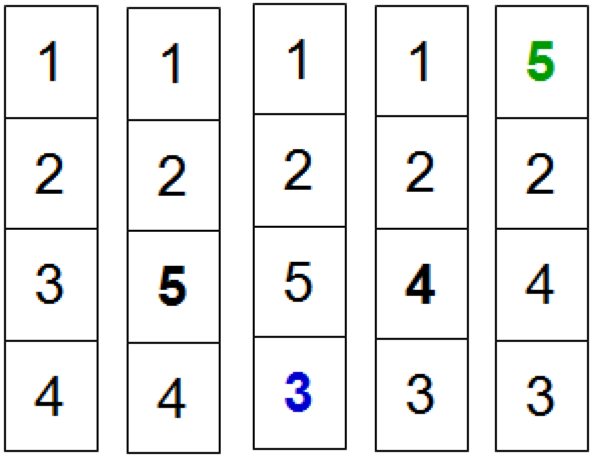
- By the principle of locality, this should be the page least likely to be referenced in the near future.
- LRU make guess based upon past experience.
- chooses that page which has not been used for longest period of time.
- each page is associated with the time of its last use.
- performs nearly as well as the optimal policy.

The LRU Policy



The LRU Policy

Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

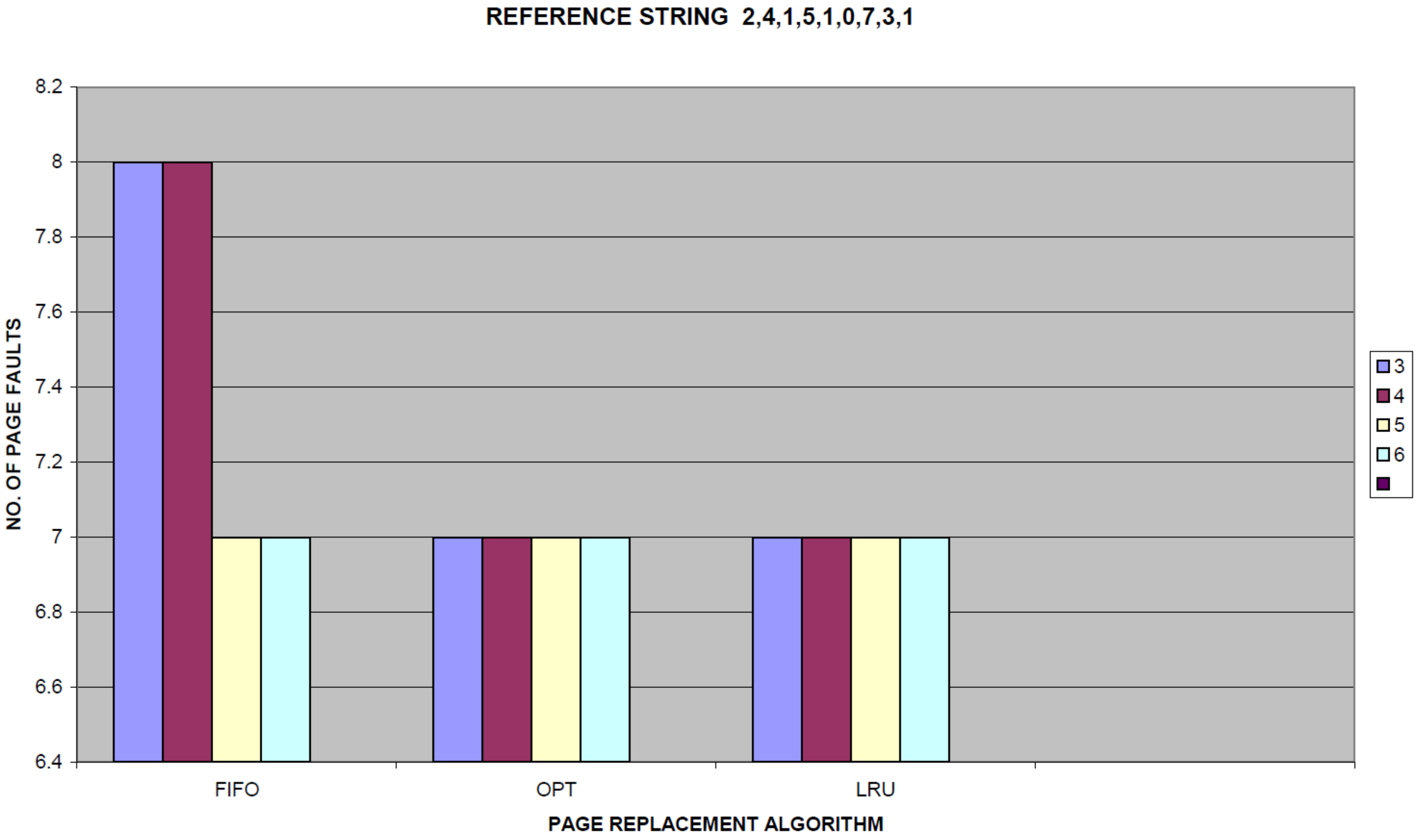


1	1	1	1	**5**
2	2	2	2	2
3	**5**	5	**4**	4
4	4	**3**	3	3

8 page faults

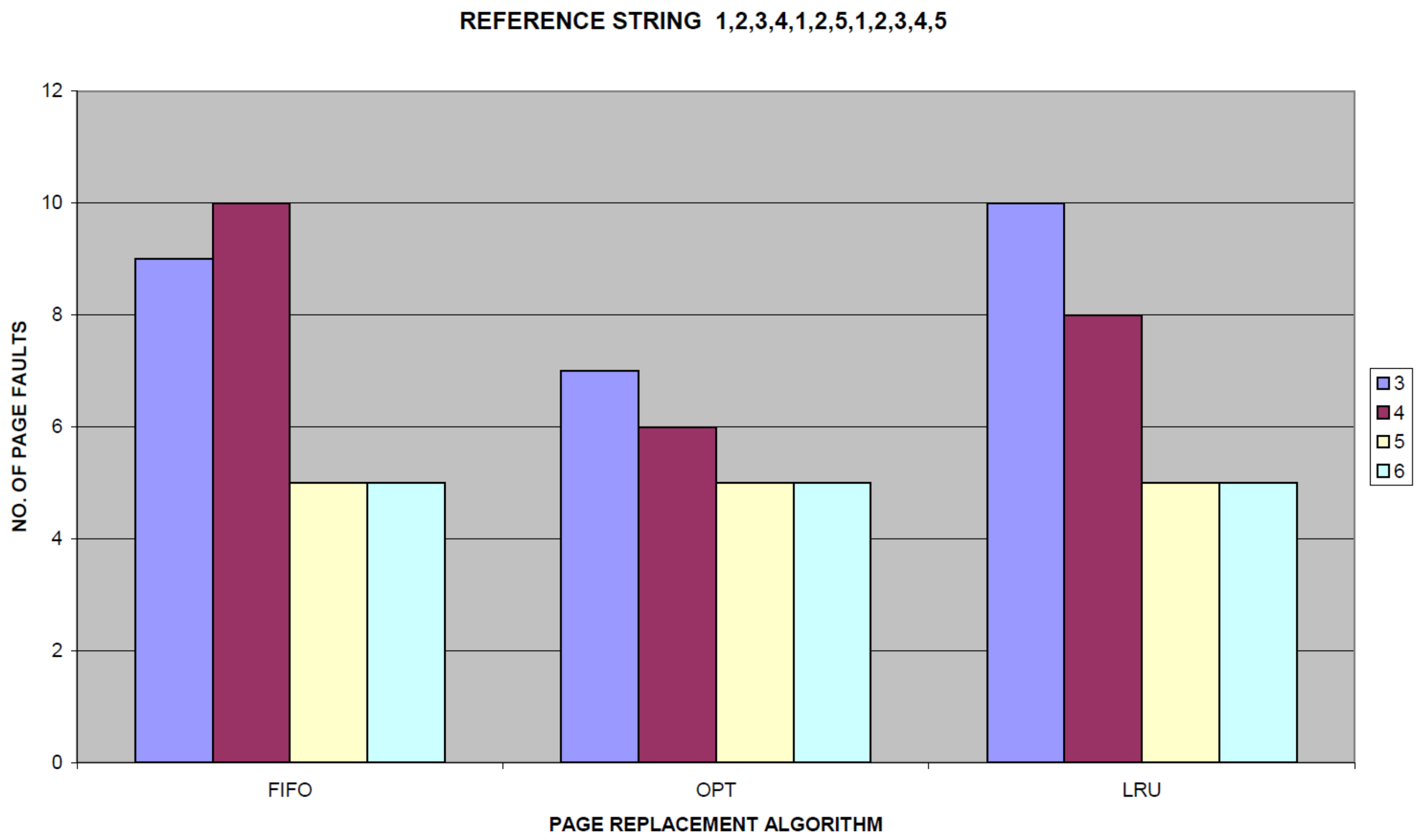
Case Study(1)

length of random reference string is (9)



Case Study(2-a)

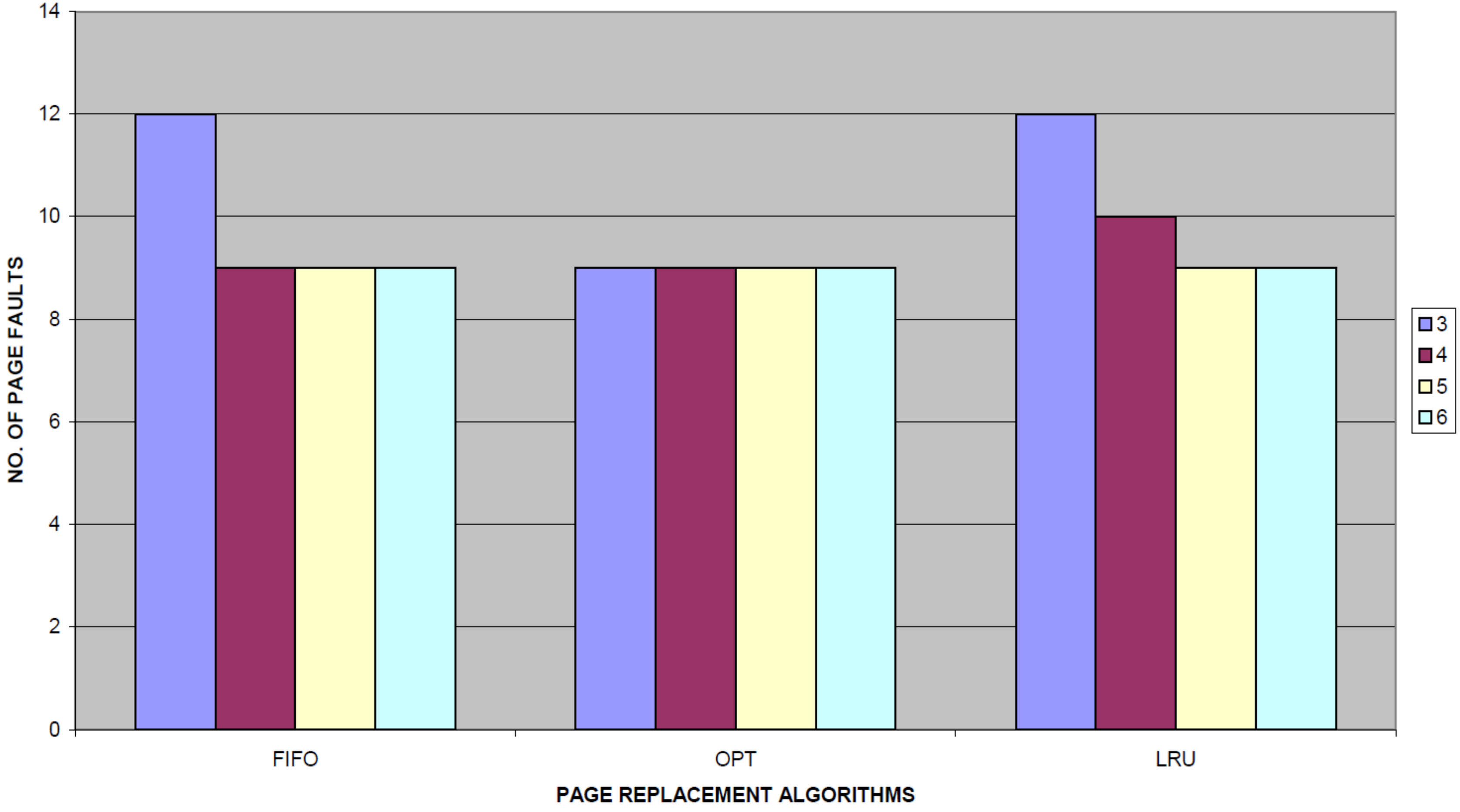
length of random reference string is (12)



Case Study(2-b)

length of random reference string is (12)

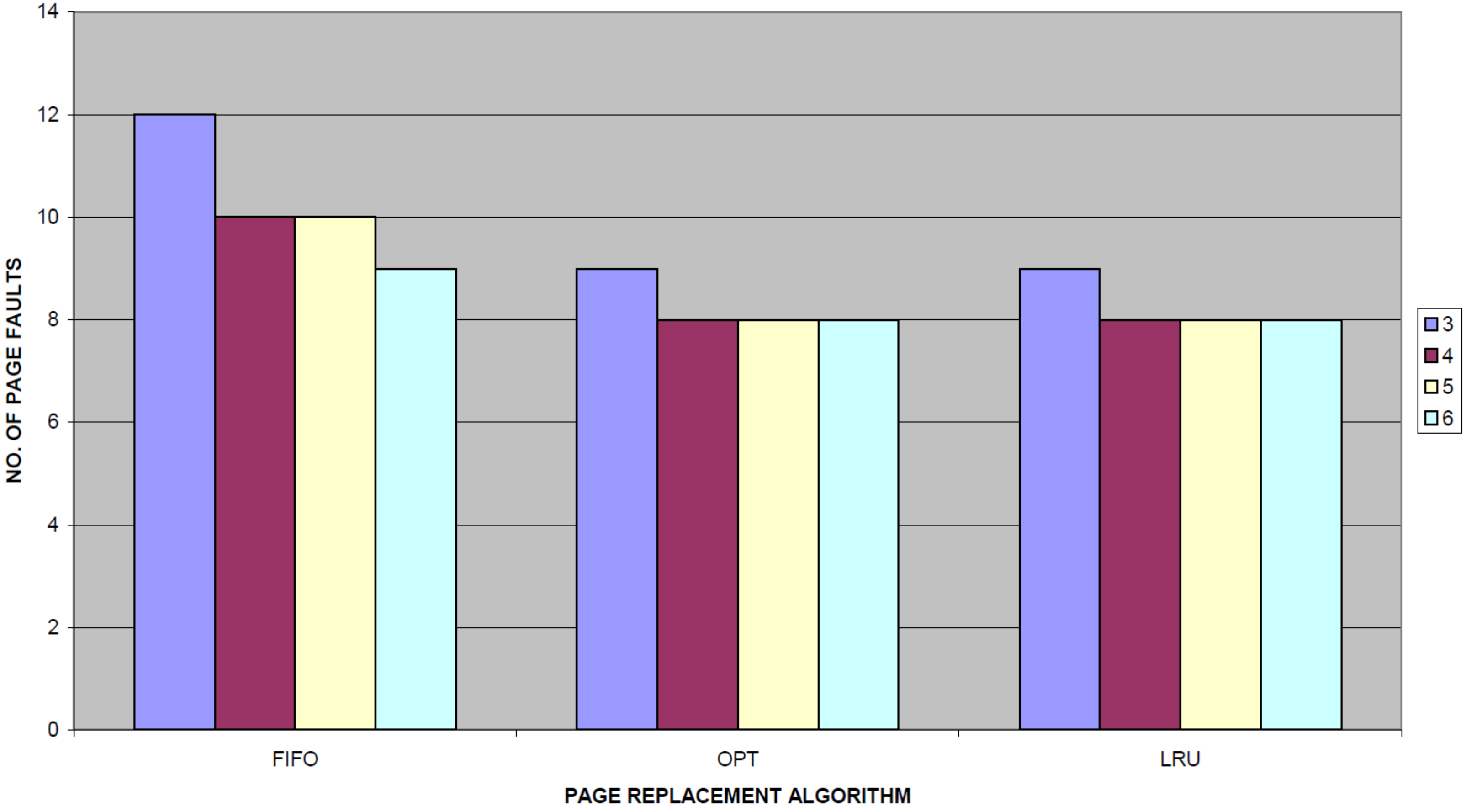
REFERENCE STRING 1,3,4,7,1,8,9,7,6,2,8,0



Case Study(3)

length of random reference string is (15)

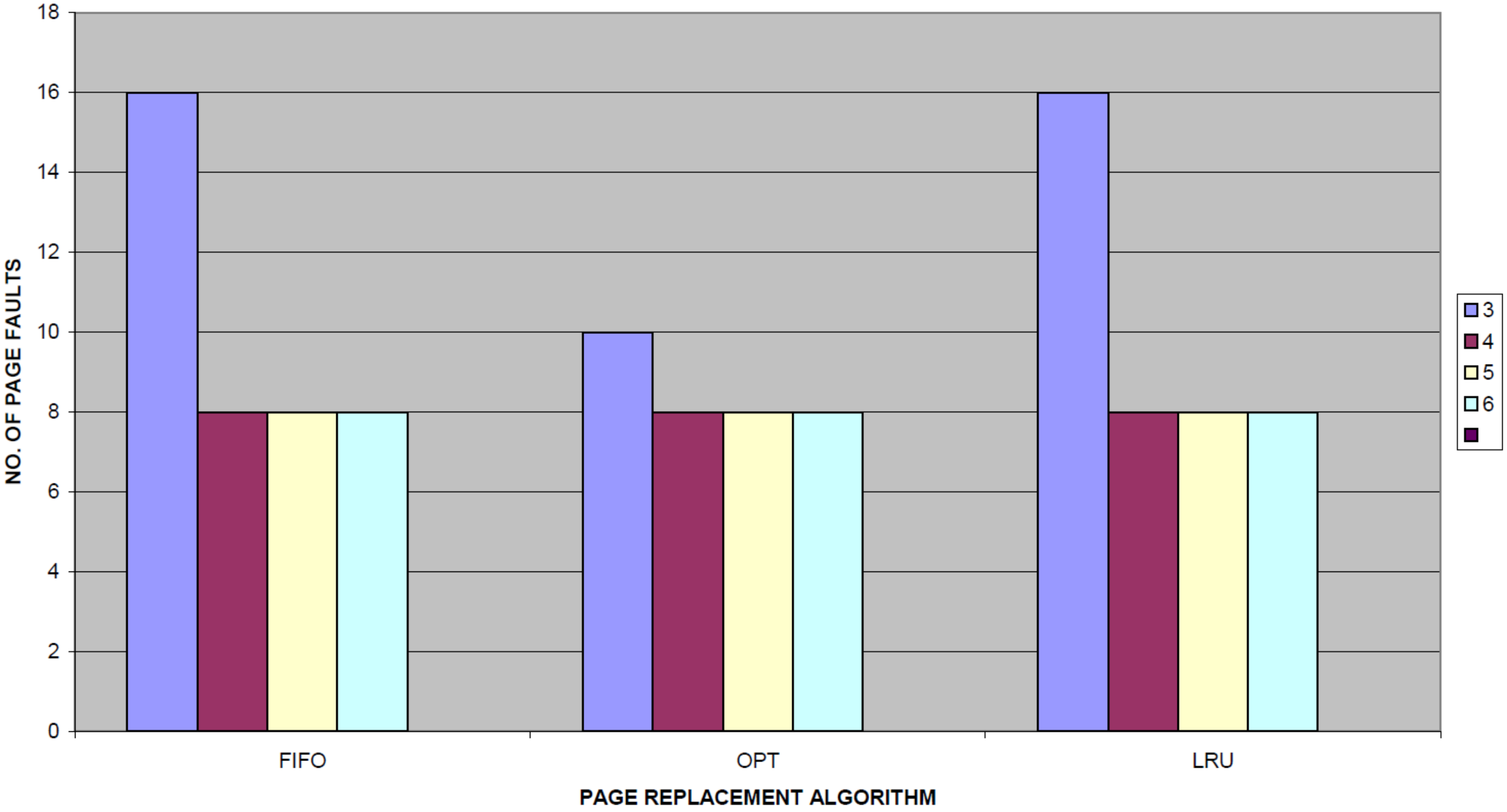
REFERENCE STRING 3,4,2,5,2,6,3,8,7,0,2,4,0,7,4



Case Study(4)

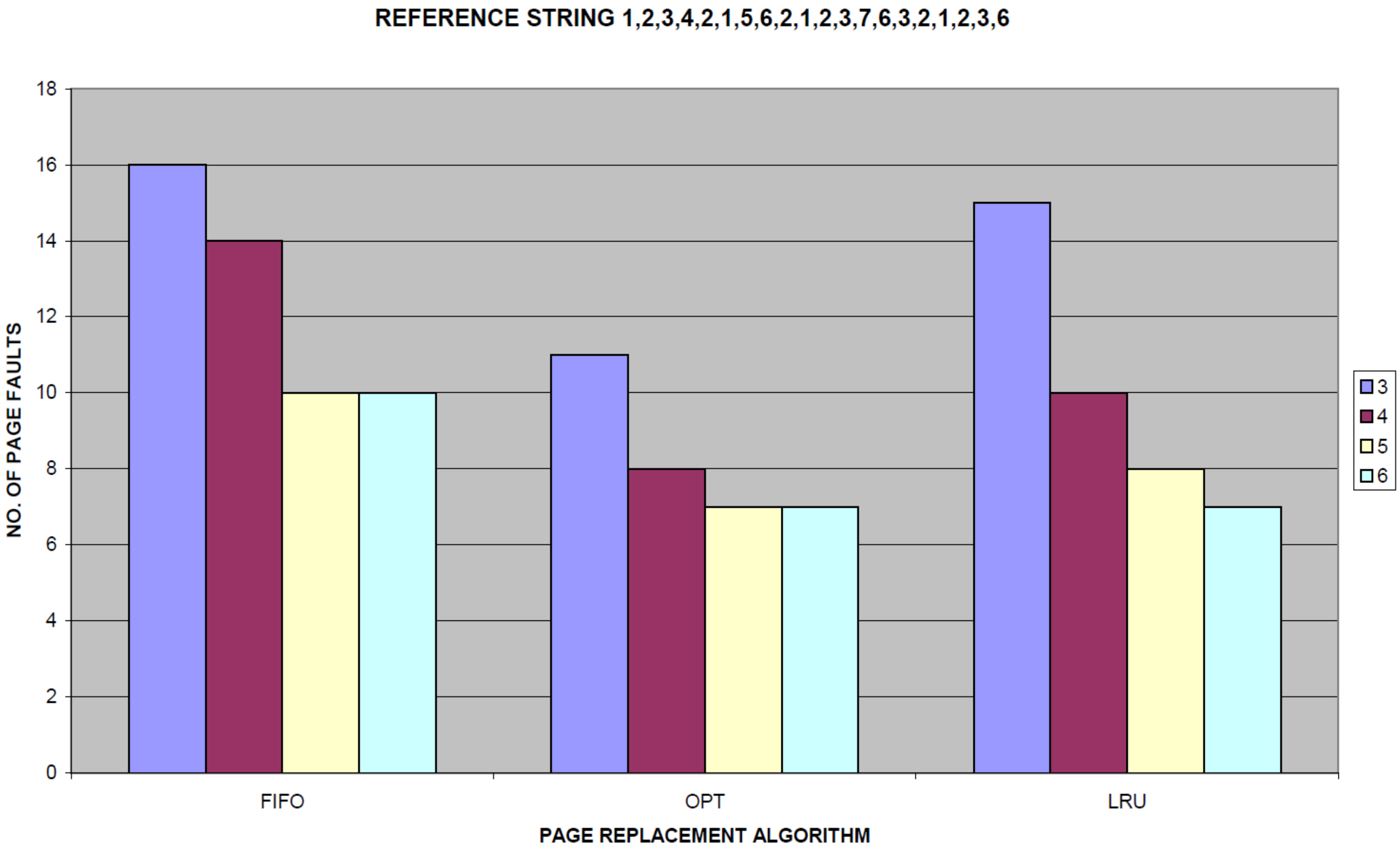
length of random reference string is (16)

REFERENCE STRING 0,1,2,3,0,1,2,3,0,1,2,3,4,5,6,7



Case Study(5)

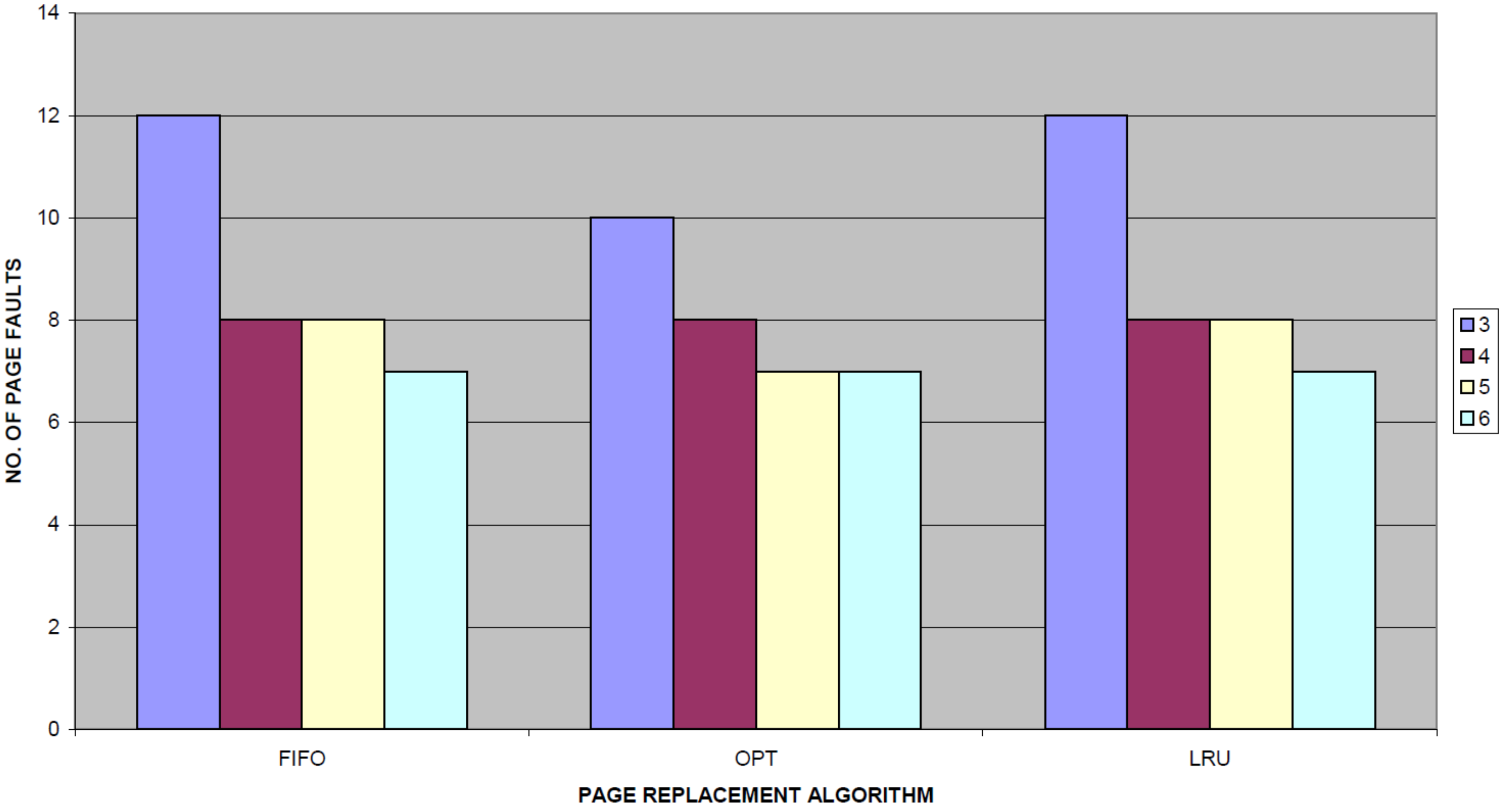
length of random reference string is (20)



Case Study(6)

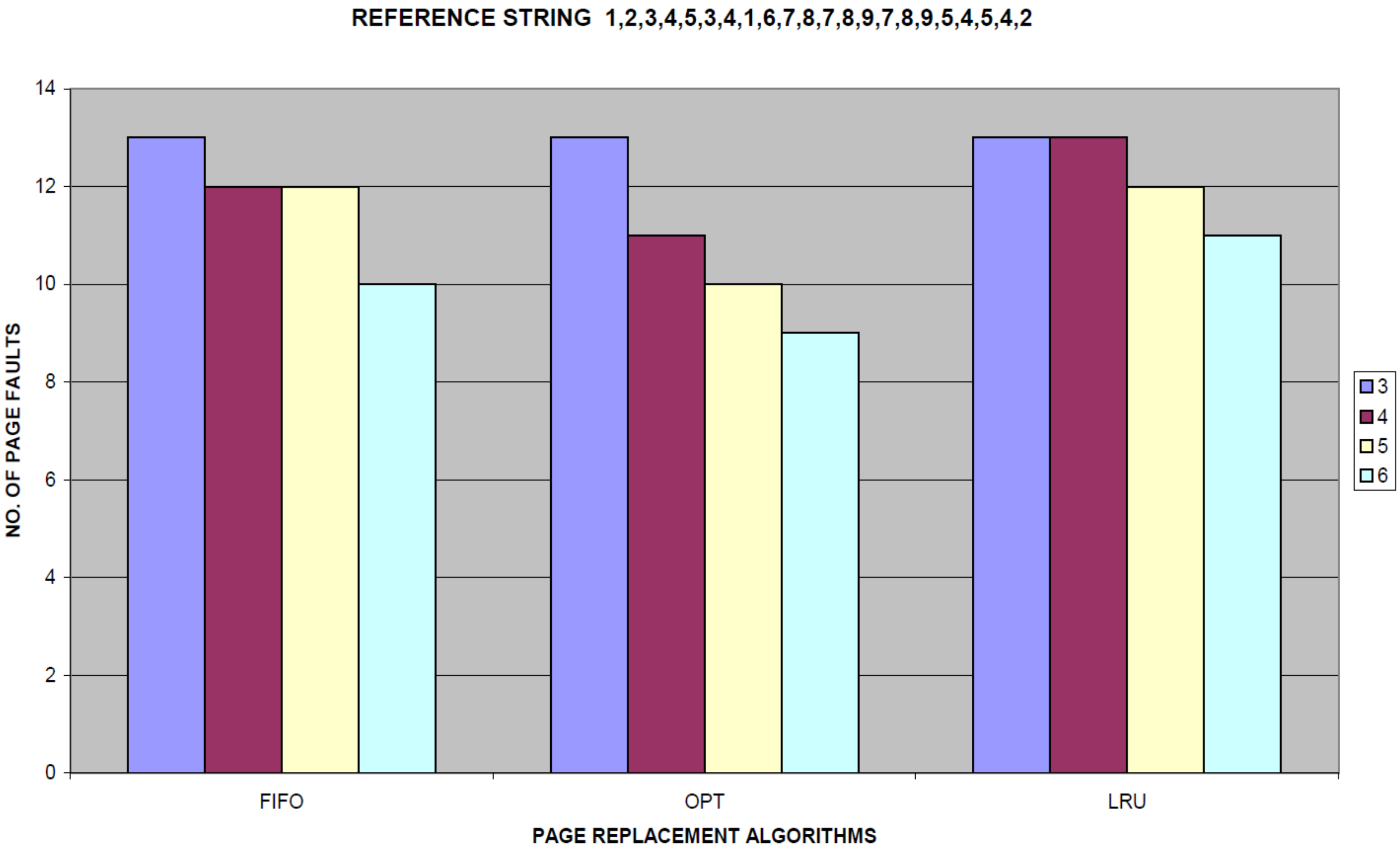
length of random reference string is (21)

REFERENCE STRING 2,3,4,3,2,4,3,2,4,5,6,7,5,6,7,4,5,6,7,2,1



Case Study(7)

length of random reference string is (22)



REFERENCE STRING 1,2,3,4,5,3,4,1,6,7,8,7,8,9,7,8,9,5,4,5,4,2

Conclusion ????

References

- System Programming and Operating System: D M Dhamdhere
- Modern Operating System: Andrew S. Tanenbaum
- Computer Organization & Architecture: T. K. Ghosh
- https://www.**wikipedia**.org