بسم الله الرحمن الرحیم

۱۸۰

«سیستم عامل»

جلسه ۱۹: مدیریت حافظه (۷)

Page replacement

- **Assume a normal page table (e.g., BLITZ)**
- **User-program is executing**
- **A PageInvalidFault occurs!**
  - The page needed is not in memory
- **Select some frame and remove the page in it**
  - If it has been modified, it must be written back to disk
    - **the "dirty" bit in its page table entry tells us if this is necessary**
- **Figure out which page was needed from the faulting addr**
- **Read the needed page into this frame**
- **Restart the interrupted process by retrying the same instruction**

Page replacement algorithms

- **Which frame to replace?**
- **Algorithms:**
  - The Optimal Algorithm
  - First In First Out (FIFO)
  - Not Recently Used (NRU)
  - Second Chance / Clock
  - Least Recently Used (LRU)
  - Not Frequently Used (NFU)
  - Working Set (WS)
  - WSClock

The optimal page replacement algorithm

- **Idea:**
  - Given all the data, how to find the optimal page replacement?

Time	0	1	2	3	4	5	6	7	8	9	10
Requests		c	a	d	b	e	b	a	b	c	d
Page Frames	0	a									
	1	b									
	2	c									
	3	d									
Page faults											

The optimal page replacement algorithm

- **Idea:**
  - Given all the data, how to find the optimal page replacement?

Time		0	1	2	3	4	5	6	7	8	9	10
Requests			c	a	d	b	e	b	a	b	c	d
Page Frames	0	a	a	a	a	a						
	1	b	b	b	b	b						
	2	c	c	c	c	c						
	3	d	d	d	d	d						
Page faults							X					

The optimal page replacement algorithm

- **Idea:**
  - Given all the data, how to find the optimal page replacement?
  - **Longest Forward Distance (LFD):** Select the page that will not be needed for the longest time

LFD

- Replace the page that will not be needed for the longest
- Example:

Time		0	1	2	3	4	5	6	7	8	9	10
Requests			c	a	d	b	e	b	a	b	c	d
Page Frames	0	a	a	a	a	a						
	1	b	b	b	b	b						
	2	c	c	c	c	c						
	3	d	d	d	d	d						
Page faults							X					

LFD

- **Select the page that will not be needed for the longest time**
- **Example:**

Time		0	1	2	3	4	5	6	7	8	9	10
Requests			c	a	d	b	e	b	a	b	c	d
Page Frames	0	a	a	a	a	a	a	a	a	a	a	
	1	b	b	b	b	b	b	b	b	b	b	
	2	c	c	c	c	c	c	c	c	c	c	
	3	d	d	d	d	d	e	e	e	e	e	
Page faults							X				X	

Is LFD optimal?

Is LFD optimal?

- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD

Is LFD optimal?

- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - First non-equal potision: i+1

Is LFD optimal?

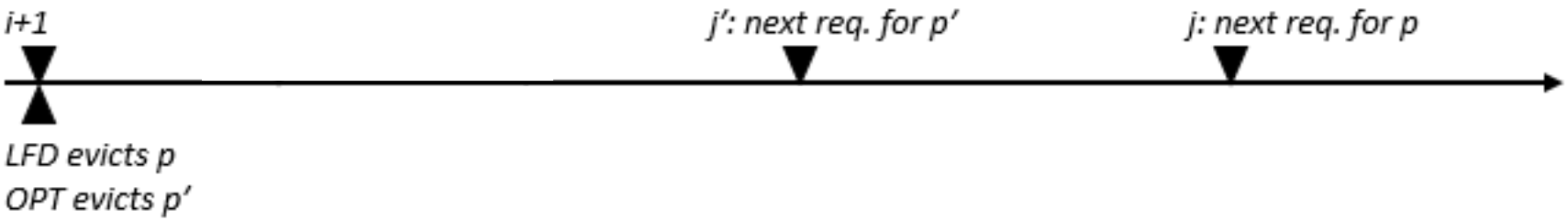
- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - First non-equal potision: i+1
  - Case 1) i+1 is not a page fault.
  - Case 2) i+1 is page fault

Is LFD optimal?

- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - First non-equal potision: i+1
  - Case 1) i+1 is not a page fault.
  - Case 2) i+1 is page fault
    - LFD evicts p
    - OPT evicts p'

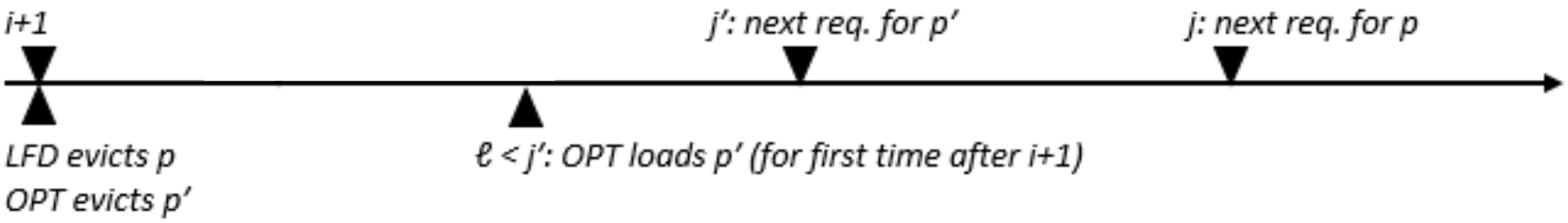
Is LFD optimal?

- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - First non-equal potision: i+1
  - Case 1) i+1 is not a page fault.
  - Case 2) i+1 is page fault
    - LFD evicts p
    - OPT evicts p'



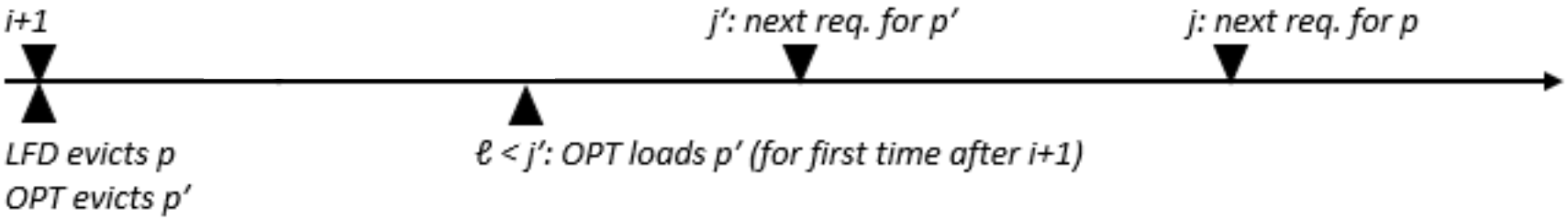
Is LFD optimal?

- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - Case 2) i+1 is page fault
    - Case 2-A) OPT keeps p until l
    - Case 2-B) OPT evicts p at l' < l



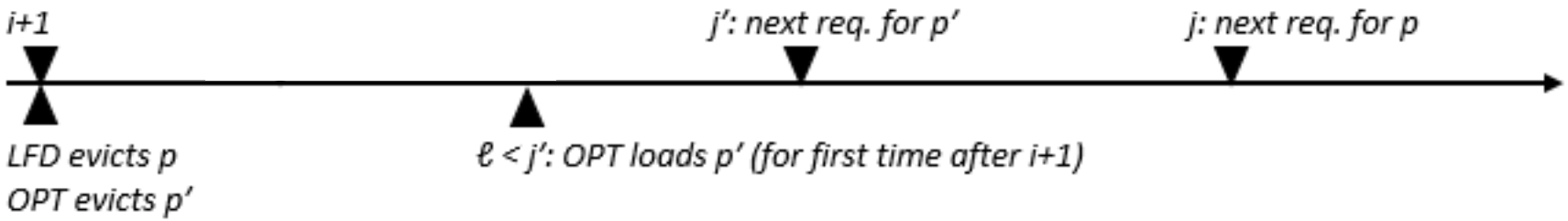
Is LFD optimal?

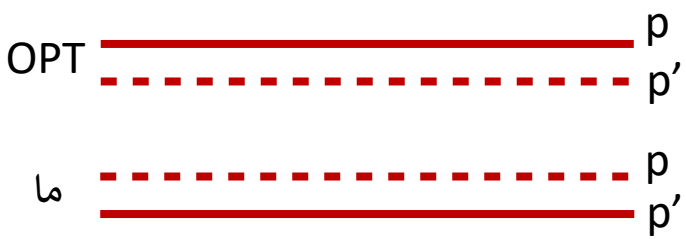
- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - Case 2) i+1 is page fault
    - Case 2-A) OPT keeps p until l
    - Case 2-B) OPT evicts p at l' < l



Is LFD optimal?

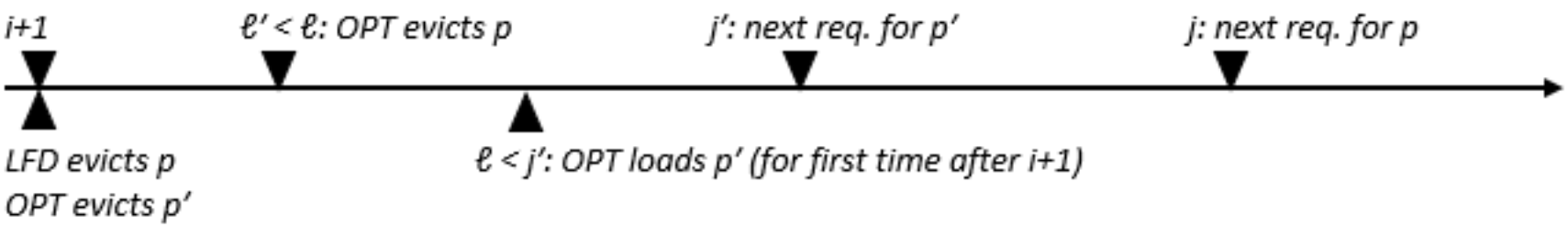
- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - Case 2) i+1 is page fault
    - Case 2-A) OPT keeps p until l
    - Case 2-B) OPT evicts p at l' < l





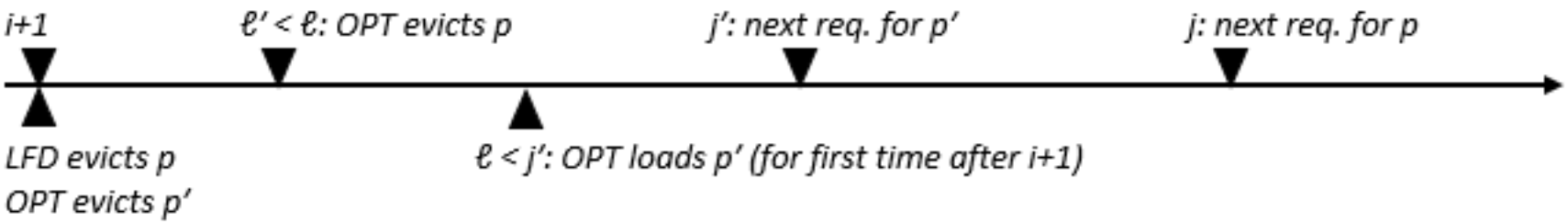
Is LFD optimal?

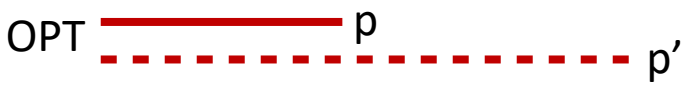
- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - Case 2) i+1 is page fault
    - Case 2-A) OPT keeps p until l
    - Case 2-B) OPT evicts p at l' < l



Is LFD optimal?

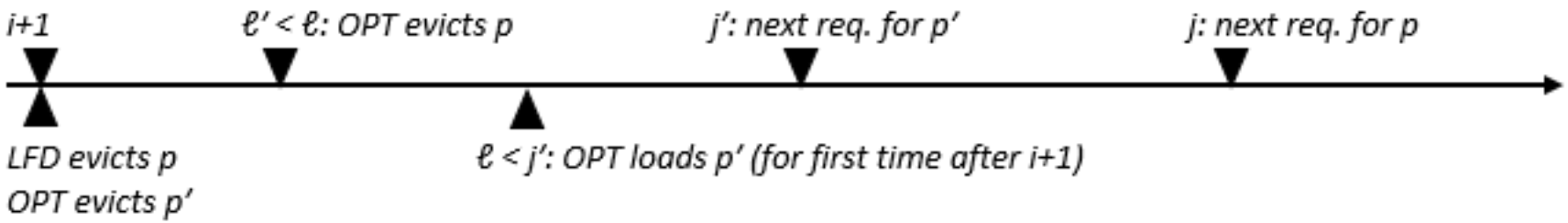
- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - Case 2) i+1 is page fault
    - Case 2-A) OPT keeps p until l
    - Case 2-B) OPT evicts p at l' < l

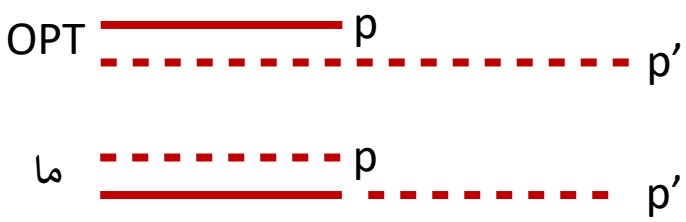




Is LFD optimal?

- Proof (by contradiction):
  - **OPT**: Optimum with longest prefix equal to LFD
  - Case 2) i+1 is page fault
    - Case 2-A) OPT keeps p until l
    - Case 2-B) OPT evicts p at l' < l





LFD = OPT