Page Replacement

- When physical memory is full, we need to choose a "victim" to evict
- Local vs Global replacement
  - Local: victim chosen from frames of process experiencing page fault
    - fixed allocation of frames per process
  - Global: victim chosen from frames allocated to any process
    - variable allocation of frames per process
- Goal: minimizing number of page faults

Page Replacement Algorithms

- Random: Pick any page to eject at random
  - Used mainly for comparison
- FIFO: The page brought in earliest is evicted
  - Ignores usage
- LRU: Evict page not been used the longest
  - Assumes past is good predictor of the future
- MRU: Evict most recently used page
  - Good for data accessed only once, e.g., a movie
- LFU: Evict least frequently used page
- OPT: Belady's algorithm

How do we pick a victim?

- We want:
  - low page fault-rate
  - page faults as inexpensive as possible
- We need:
  - a way to compare the relative performance of different page replacement algorithms
  - some absolute notion of what a "good" page replacement algorithm should accomplish

Comparing Page Replacement Algorithms

- Record a trace of the pages accessed by a process
  - E.g. 3,1,4,2,5,2,1,2,3,4 (or c,a,d,b,e,b,a,b,c,b)
- Simulate behavior of page replacement algorithm on trace
- Record number of page faults generated

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a												
Page Frames	0													
	1													
	2													
Faults		X												

Process can use 3 frames
(3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b											
Page Frames	0		a											
	1													
	2													
Faults		X	X											

Process can use 3 frames
(3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c										
Page Frames	0		a	a										
	1			b										
	2													
Faults		X	X	X										

Process can use 3 frames
(3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a									
	1			b	b									
	2				c									
Faults		X	X	X	X									

Process can use 3 frames
(3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a								
Page Frames	0		a	a	a	a								
	1			b	b	b								
	2				c	d								
Faults		X	X	X	X	✓								

Process can use 3 frames (3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b							
Page Frames	0		a	a	a	a	a							
	1			b	b	b	b							
	2				c	d	d							
Faults		X	X	X	X	✓	✓							

Process can use 3 frames (3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	a	a	a						
	1			b	b	b	b	b						
	2				c	d	d	d						
Faults		X	X	X	X	✓	✓	X						

Process can use 3 frames (3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a					
Page Frames	0		a	a	a	a	a	a	a					
	1			b	b	b	b	b	b					
	2				c	d	d	d	e					
Faults		X	X	X	X	✓	✓	X	✓					

Process can use 3 frames
(3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b				
Page Frames	0		a	a	a	a	a	a	a	a				
	1			b	b	b	b	b	b	b				
	2				c	d	d	d	e	e				
Faults		X	X	X	X	✓	✓	X	✓	✓				

Process can use 3 frames
(3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	a	a	a	a	a	a			
	1			b	b	b	b	b	b	b	b			
	2				c	d	d	d	e	e	e			
Faults		X	X	X	X	✓	✓	X	✓	✓	X			

Process can use 3 frames
(3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	a	a	a	a	a	a	c		
	1			b	b	b	b	b	b	b	b	b		
	2				c	d	d	d	e	e	e	e		
Faults		X	X	X	X	✓	✓	X	✓	✓	X	X		

Process can use 3 frames
(3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	a	a	a	a	a	a	c	c	
	1			b	b	b	b	b	b	b	b	b	d	
	2				c	d	d	d	e	e	e	e	e	
Faults		X	X	X	X	✓	✓	X	✓	✓	X	X	✓	

Process can use 3 frames
(3 pages in memory)

Page loaded

Optimal Page Replacement

- Replace page needed furthest in future

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	a	a	a	a	a	a	c	c	c
	1			b	b	b	b	b	b	b	b	b	d	d
	2				c	d	d	d	e	e	e	e	e	e
Faults		X	X	X	X	✓	✓	X	✓	✓	X	X	✓	

7 page faults

Process can use 3 frames (3 pages in memory)

Page loaded

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	a	a	a	a	a	a	a	d	d
	1			b	b	b	b	b	b	b	b	b	a	e
	2				c	c	c	c	c	c	c	c	b	b
	3					d	d	d	e	e	e	e	e	c
Faults		X	X	X	X	✓	✓	X	✓	✓	✓	X	✓	

6 page faults

Process can use 4 frames (4 pages in memory)

FIFO Replacement

- Replace pages in the order they come into memory

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	d	d	d	e	e	e	e	e	e
	1			b	b	b	a	a	a	a	a	c	c	c
	2				c	c	c	b	b	b	b	b	d	d
Faults		X	X	X	X	X	X	X	✓	✓	X	X	✓	

Process can use 3 frames (3 pages in memory)

Page loaded

9 page faults

FIFO Replacement

- Replace pages in the order they come into memory

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	a	a	a	e	e	e	e	d	d
	1			b	b	b	b	b	b	a	a	a	a	e
	2				c	c	c	c	c	c	b	b	b	b
	3					d	d	d	d	d	d	c	c	c
Faults		X	X	X	X	✓	✓	X	X	X	X	X	X	

Process can use 4 frames (4 pages in memory)

Page loaded

10 page faults

More frames —> more page faults?

Belady's Anomaly

Locality of Reference

- If a process access a memory location, then it is likely that
  - the same memory location is going to be accessed again in the near future (temporal locality)
  - nearby memory locations are going to be accessed in the future (spatial locality)
- 90% of the execution of a program is sequential
- Most iterative constructs consist of a relatively small number of instructions

LRU: Least Recently Used

- Replace page not referenced for the longest time

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b							
Page Frames	0		a	a	a	a	a	a						
	1			b	b	b	b	b						
	2				c	c	c	c						
	3					d	d	d						
Faults		X	X	X	X	✓	✓							

Process can use 4 frames
(4 pages in memory)

LRU: Least Recently Used

- Replace page not referenced for the longest time

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e						
Page Frames	0		a	a	a	a	a	a						
	1			b	b	b	b	b						
	2				c	c	c	c						
	3					d	d	d						
Faults		X	X	X	X	✓	✓	X						

Process can use 4 frames
(4 pages in memory)

LRU: Least Recently Used

- Replace page not referenced for the longest time

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e						
Page Frames	0		a	a	a	a	a	a	a					
	1			b	b	b	b	b	b					
	2				c	c	c	c	e					
	3					d	d	d	d					
Faults		X	X	X	X	✓	✓	X						

Process can use 4 frames
(4 pages in memory)

LRU: Least Recently Used

- Replace page not referenced for the longest time

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	a	a	a	a	a	a	a	a	e
	1			b	b	b	b	b	b	b	b	b	b	b
	2				c	c	c	c	e	e	e	e	d	d
	3					d	d	d	d	d	d	c	c	c
Faults		X	X	X	X	✓	✓	X	✓	✓	X	X	X	

8 page faults

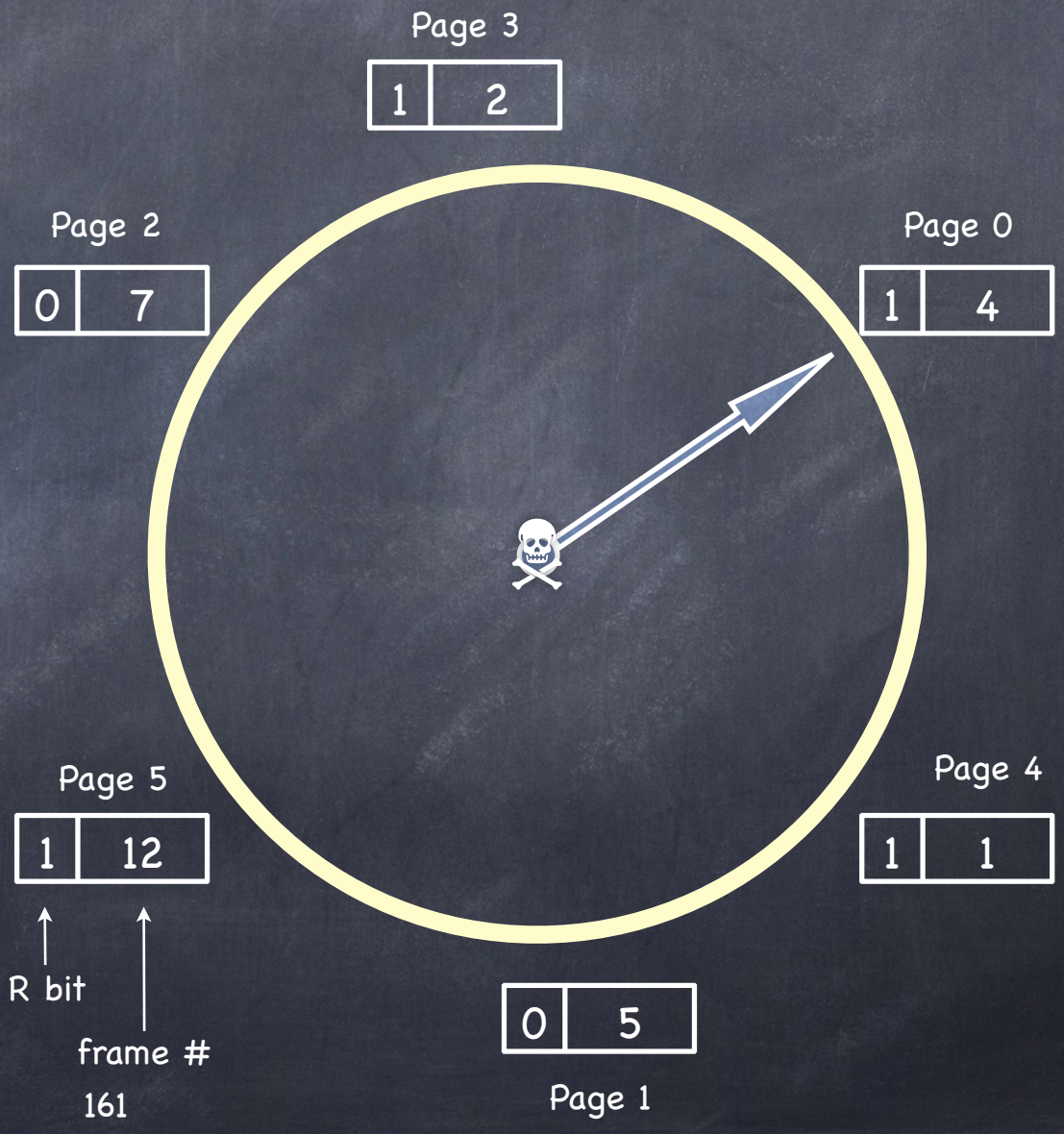
Process can use 4 frames
(4 pages in memory)

Implementing LRU

- On reference: timestamp each page
- On eviction: scan for oldest page
- Problems:
  - Large page lists
  - Timestamps are costly
- Solution: approximate LRU
  - after all, LRU is already an approximation! (of OPT)
  - exploit Reference (R) bit

The Clock Algorithm

- Organize pages in memory as a circular list
- When page is referenced, set its reference bit R to 1
- On page fault
  - if R = 1: clear R
  - else if R = 0:
    - evict page pointed to
    - load new page
    - set R to 1
  - advance hand



Clock Page Replacement

Time		0	1	2	3	4	5	6	7	8	9	10
Trace		c	a	d	b							
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						
Faults												

Page table entries for resident pages

1	a
1	b
1	c
1	d

Hand clock:

Clock Page Replacement

Time		0	1	2	3	4	5	6	7	8	9	10
Trace		c	a	d	b	e						
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						
Faults						X						

Page table entries for resident pages

1	a
1	b
1	c
1	d

Hand clock:

Clock Page Replacement

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

Page table entries for resident pages

1	a
1	b
1	c
1	d

1	e
0	b
0	c
0	d

Hand clock:

Clock Page Replacement

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

Page table entries for resident pages

1	a
1	b
1	c
1	d

1	e
0	b
0	c
0	d

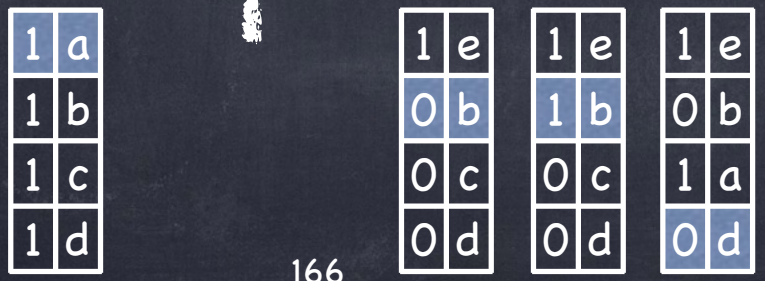
1	e
1	b
0	c
0	d

Hand clock:

Clock Page Replacement

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

Page table entries for resident pages



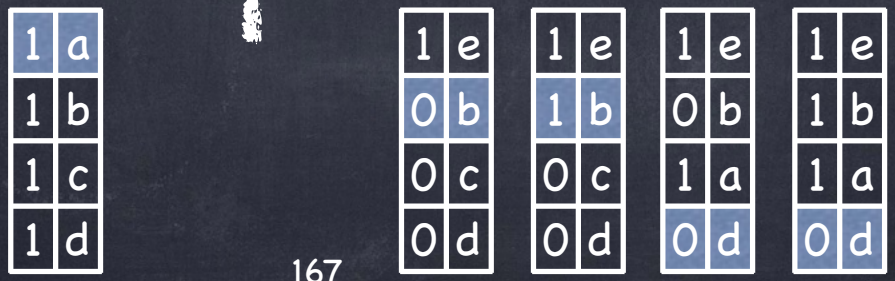
Hand clock:

Clock Page Replacement

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

Page table entries for resident pages

Hand clock:



Clock Page Replacement

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

Page table entries for resident pages

1	a
1	b
1	c
1	d

1	e
0	b
0	c
0	d

1	e
1	b
0	c
0	d

1	e
0	b
1	a
0	d

1	e
1	b
1	a
0	d

1	e
1	b
1	a
1	c

Hand clock: ■

Clock Page Replacement

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

Page table entries for resident pages

1	a
1	b
1	c
1	d

1	e
0	b
0	c
0	d

1	e
1	b
0	c
0	d

1	e
0	b
1	a
0	d

1	e
1	b
1	a
0	d

1	e
1	b
1	a
1	c

1	d
0	b
0	a
0	c

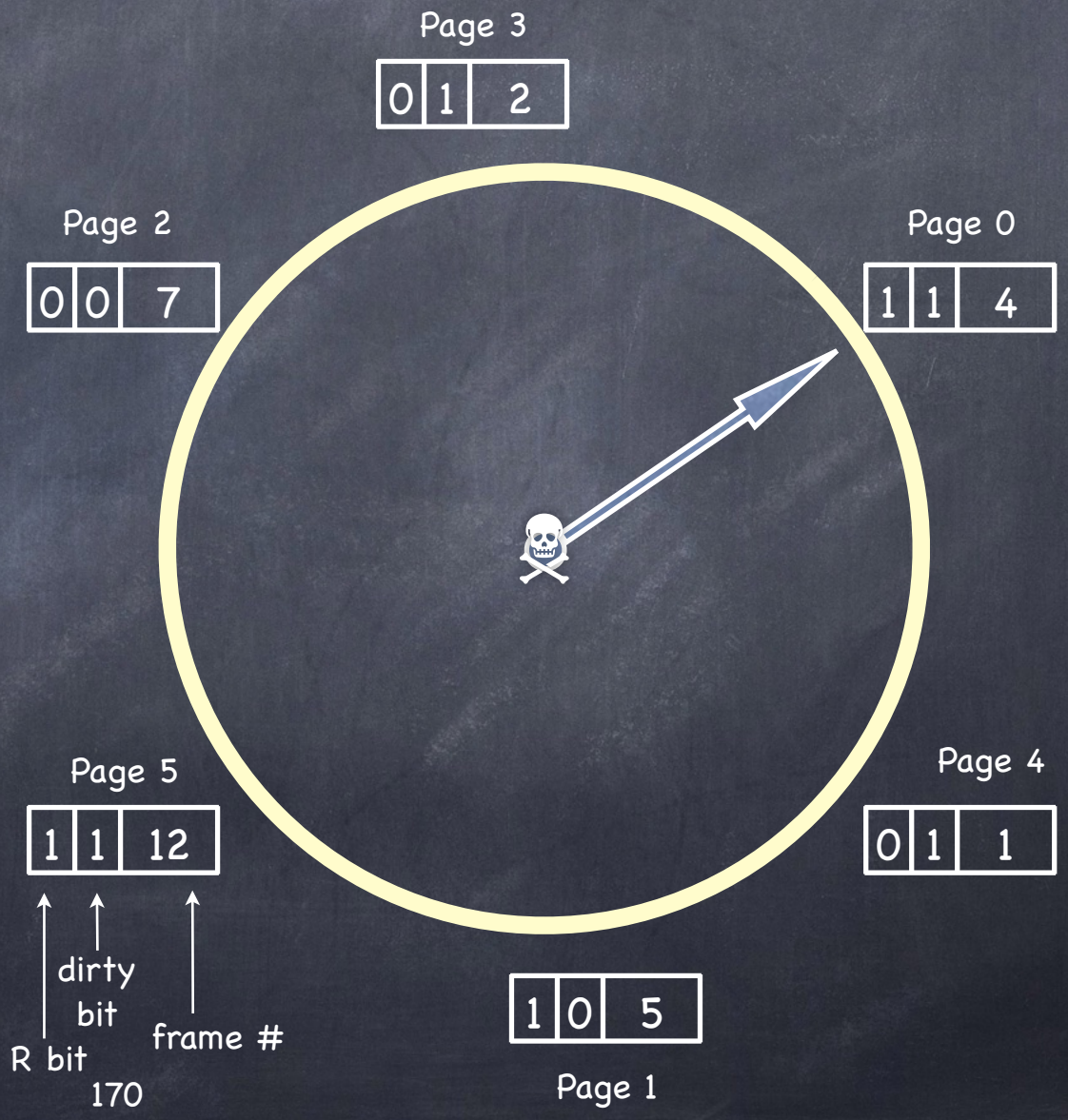
Hand clock:

The Second Chance Algorithm

- Dirty pages get "second chance" before eviction
  - replacing dirty pages is expensive!

Before Clock sweep		After Clock sweep	
R	dirty	R	dirty
0	0	replace page	
1	0	0	0
0	1	0	0*
1	1	0	1

* = remember when evicted must be saved to disk!



Second Chance Page Replacement

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

Page table entries for resident pages

10	a
10	b
10	c
10	d

Hand clock:

Second Chance Page Replacement

Time		0	1	2	3	4	5	6	7	8	9	10
Requests		c	a^w	d	b^w	e						
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						
Faults						X						

Page table entries for resident pages

10	a
10	b
10	c
10	d

11	a
11	b
10	c
10	d

Hand clock: 

Second Chance Page Replacement

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

Page table entries for resident pages

10	a
10	b
10	c
10	d

11	a
11	b
10	c
10	d

00	a
00	b
10	e
00	d

Hand clock: 

Second Chance Page Replacement

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

Page table entries for resident pages

10	a
10	b
10	c
10	d

11	a
11	b
10	c
10	d

00	a
00	b
10	e
00	d

00	a
10	b
10	e
00	d

Hand clock:

Second Chance Page Replacement

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

Page table entries for resident pages

10	a
10	b
10	c
10	d

11	a
11	b
10	c
10	d

00	a
00	b
10	e
00	d

00	a
10	b
10	e
00	d

Hand clock:

Second Chance Page Replacement

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

Page table entries for resident pages

10	a	11	a	00	a	00	a	11	a
10	b	11	b	00	b	10	b	10	b
10	c	10	c	10	e	10	e	10	e
10	d	10	d	00	d	00	d	00	d

Hand clock: ■

Second Chance Page Replacement

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

Page table entries for resident pages

10	a	11	a	00	a	00	a	11	a
10	b	11	b	00	b	10	b	10	b
10	c	10	c	10	e	10	e	10	e
10	d	10	d	00	d	00	d	00	d

Hand clock:

Second Chance Page Replacement

Time		0	1	2	3	4	5	6	7	8	9	10
Requests		c	a^w	d	b^w	e	b	a^w	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	e	e	e	e	e	
	3	d	d	d	d	d	d	d	d	d	c	
Faults						X				X	X	

Page table entries for resident pages

10	a	11	a	00	a	00	a	11	a		11	a
10	b	11	b	00	b	10	b	10	b		10	b
10	c	10	c	10	e	10	e	10	e		10	e
10	d	10	d	00	d	00	d	00	d		10	c

Hand clock:

Second Chance Page Replacement

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

Page table entries for resident pages

10	a	11	a	00	a	00	a	11	a	11	a	00	a
10	b	11	b	00	b	10	b	10	b	10	b	10	d
10	c	10	c	10	e	10	e	10	e	10	e	00	e
10	d	10	d	00	d	00	d	00	d	10	c	00	c

Hand clock:

Back to Belady's Anomaly

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	d	d	d	e	e	e	e	e	e
	1			b	b	b	a	a	a	a	a	c	c	c
	2				c	c	c	b	b	b	b	b	d	d
Faults		X	X	X	X	X	X	X			X	X	X	

3 frames:
9 page faults

Time		0	1	2	3	4	5	6	7	8	9	10	11	12
Trace		a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames	0		a	a	a	a	a	a	e	e	e	e	d	d
	1			b	b	b	b	b	a	a	a	a	a	e
	2				c	c	c	c	c	c	b	b	b	b
	3					d	d	d	d	d	d	c	c	c
Faults			X	X	X	X			X	X	X	X	X	X

4 frames:
10 page faults!

Taming Belady:
Stack Page Replacement Policies

- Given m frames for trace r, let M(m,r) be the set of virtual pages in physical memory
- A stack page replacement policy has the property that, for all number of frames m and for all traces r

M(m,r) is a subset of M(m+1, r)

Stack page replacement policies do
not suffer from Belady's anomaly:
more frames —> not more misses

FIFO: m=3 vs m=4

Time	0	1	2	3	4	5	6	7	8	9	10	11	12
Trace	a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames 0		a	a	a	d	d	d	e	e	e	e	e	e
Page Frames 1			b	b	b	a	a	a	a	a	c	c	c
Page Frames 2				c	c	c	b	b	b	b	b	d	d
Faults	X	X	X	X	X	X	X			X	X	X	

Time	0	1	2	3	4	5	6	7	8	9	10	11	12
Trace	a	b	c	d	a	b	e	a	b	c	d	e	
Page Frames 0		a	a	a	a	a	a	e	e	e	e	d	d
Page Frames 1			b	b	b	b	b	a	a	a	a	a	e
Page Frames 2				c	c	c	c	c	c	b	b	b	b
Page Frames 3					d	d	d	d	d	d	c	c	c
Faults		X	X	X	X			X	X	X	X	X	X

Stack "subset property" violated

Theorem: Stack Property holds for LRU and MRU

- By definition:
  - For LRU: M(m+1,r) contains m most frequently used frames, so M(m,r) is a subset of M(m+1,r)
  - A similar argument holds for MRU, LFU

Theorem: Stack Property holds for OPT

- Proof is non-trivial!
  - You can find more in the paper that introduced the notion of stock replacement policies

R.L. Mattson, J. Gecsei, D.R. Slutz, and I. L. Traiger, "Evaluation Techniques for Storage Hierarchies" in IBM Systems Journl, vol. 9, no. 2, pp. 78-117, 1970

Local vs. Global Page Replacement

- Local: processes select victim among frames allocated to them
  - Can lead to under utilization
- Global: Select any frame, even if allocated to another process

Brother, can you spare a frame?

FIFO

Time	0	1	2	3	4	5	6	7	8	9	10	11	12
Trace	a	b	c	d	a	b	c	d	a	b	c	d	
0		a	a	a	d	d	d	c	c	c	b	b	b
1			b	b	b	a	a	a	d	d	d	c	c
2				c	c	c	b	b	b	a	a	a	d
Faults	X	X	X	X	X	X	X	X	X	X	X	X	X

Brother, can you spare a frame?

FIFO

Time	0	1	2	3	4	5	6	7	8	9	10	11	12
Trace	a	b	c	d	a	b	c	d	a	b	c	d	
0		a	a	a	a	a	a	a	a	a	a	a	a
1			b	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	d	d	d	d
Faults	X	X	X	X	X								

What's not to like?

- Global page replacement can use more efficiently a limited pool of frames...
- ... but processes lose control over their own page fault rate
- How can we manage global replacement to minimize page faults?

Back to basics: physical memory as a cache

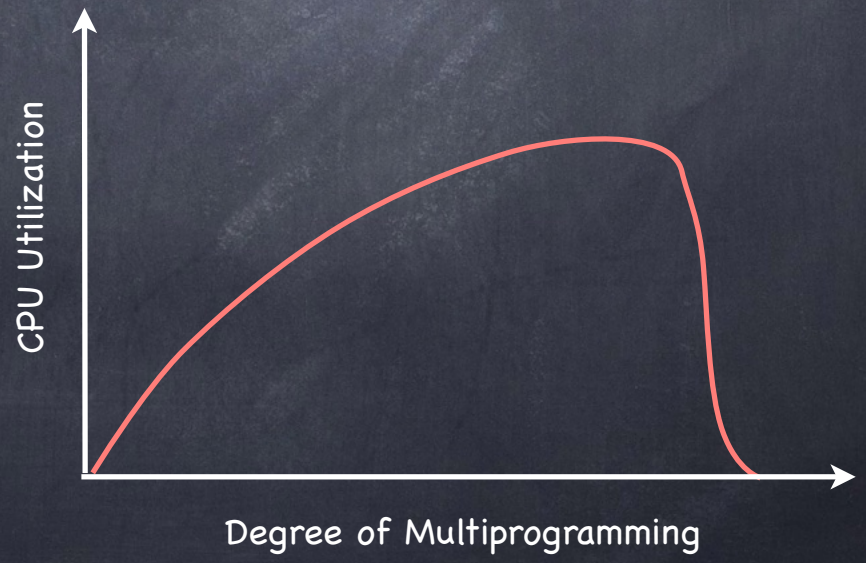
- Demand paging enables frames to cache part of a process VA space
- If the cache is large enough, hit ratio is high
  - few page faults
- What if there aren't enough frames to go around?
  - should decrease degree of multiprogramming
    - a swapped out process can then release its frames

Instead, if we are not careful...

- When not enough frames...
  - high page fault rate leads to low CPU utilization
  - OS may increase degree of multiprogramming!

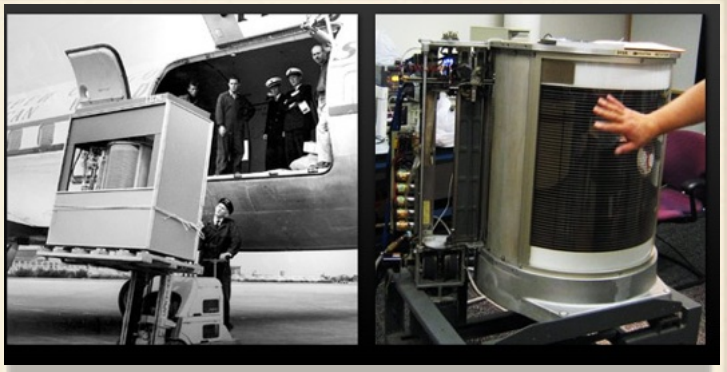
Instead, if we are not careful...

- When not enough frames...
  - high page fault rate leads to low CPU utilization
  - OS may increase degree of multiprogramming!
- Thrashing
  - process spends all its time swapping pages in and out



Why "thrashing"?

"Thrash" dates from the 1960's, when disk drives were as large as washing machines. If a program's working set did not fit in memory, the system would need to shuffle memory pages back and forth to disk. This burst of activity would violently shake the disk drive.



The first hard disk drive:
IBM Model 350 Disk File
(came w/IBM 305 RAMAC)
1956

Total storage =
5 million characters
(just under 5 MB).

http://royal.pingdom.com/2008/04/08/the-history-of-computer-data-storage-in-pictures/

Locality of Reference

- If a process access a memory location, then it is likely that
  - the same memory location is going to be accessed again in the near future (temporal locality)
  - nearby memory locations are going to be accessed in the future (spatial locality)
- 90% of the execution of a program is sequential
- Most iterative constructs consist of a relatively small number of instructions

Tracking Locality

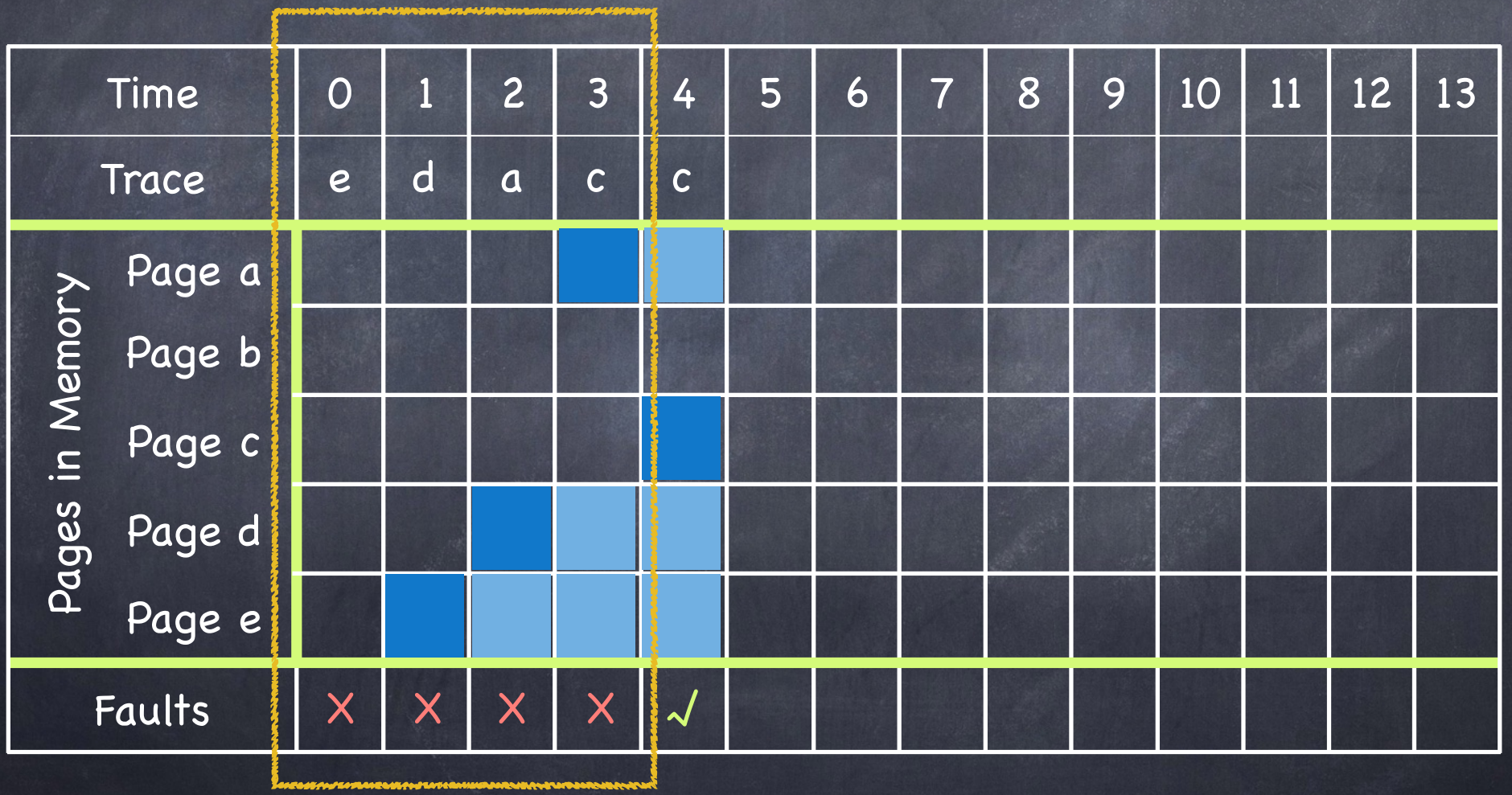
- When a process executes it moves from locality (set of pages used together) to locality
  - the size of the process' locality (a.k.a. its working set) can change over time
- Goal: track the size of the process' working set, dynamically acquiring and releasing frames as necessary

The Working Set Model

- Choose Δ page references as WS sliding window
  - track WS for the last Δ page references
- WSS_i = # of distinct pages referenced by p_i in latest Δ references
  - Δ too small does not cover locality
  - Δ too large covers many localities
- Thrashing if $\sum_i WSS_i$ > # frames
  - if so, swap out one of the processes; free its frames
- If enough free frames, increase degree of multiprogramming

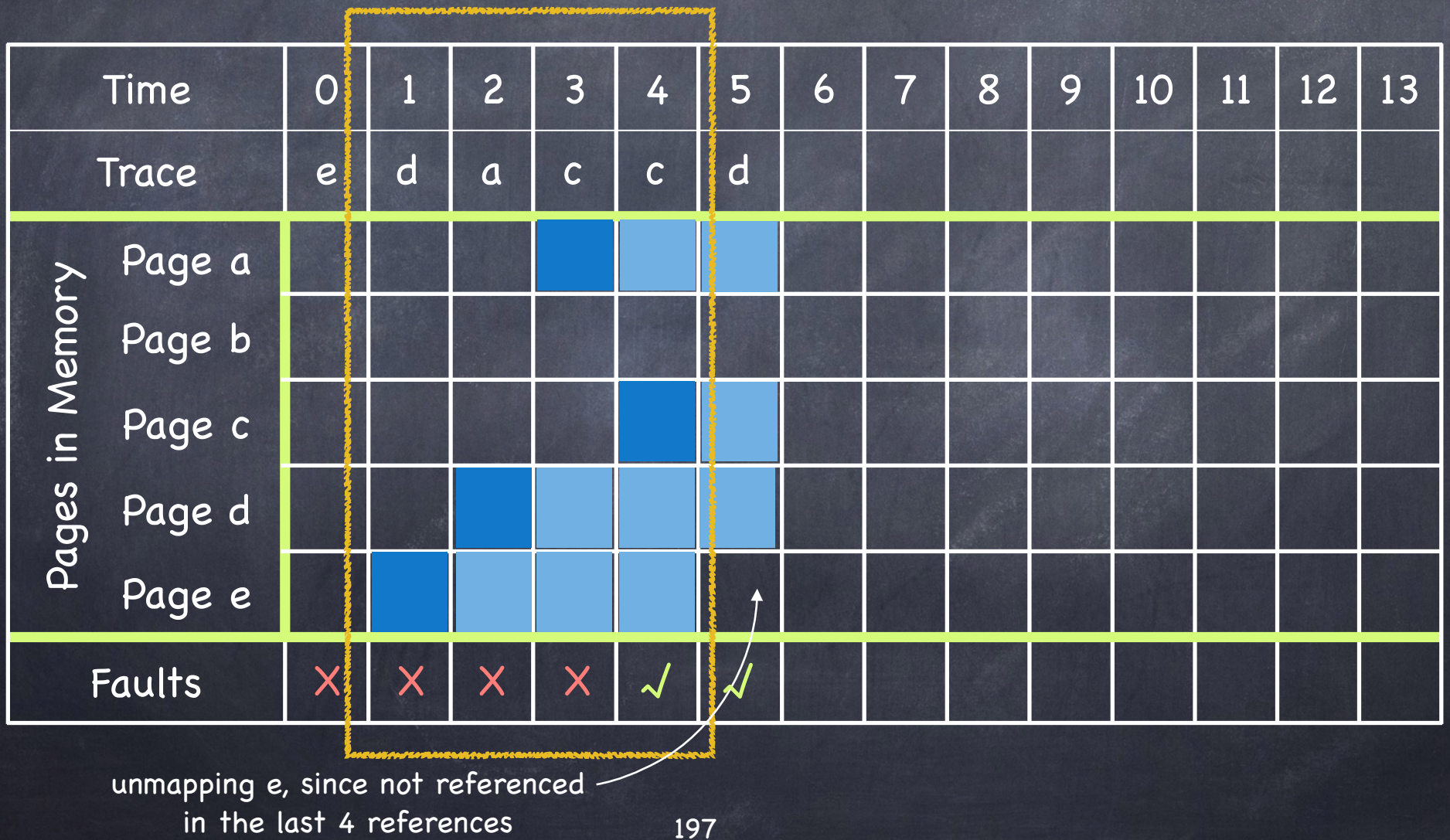
WS Page Replacement

$\Delta = 4$



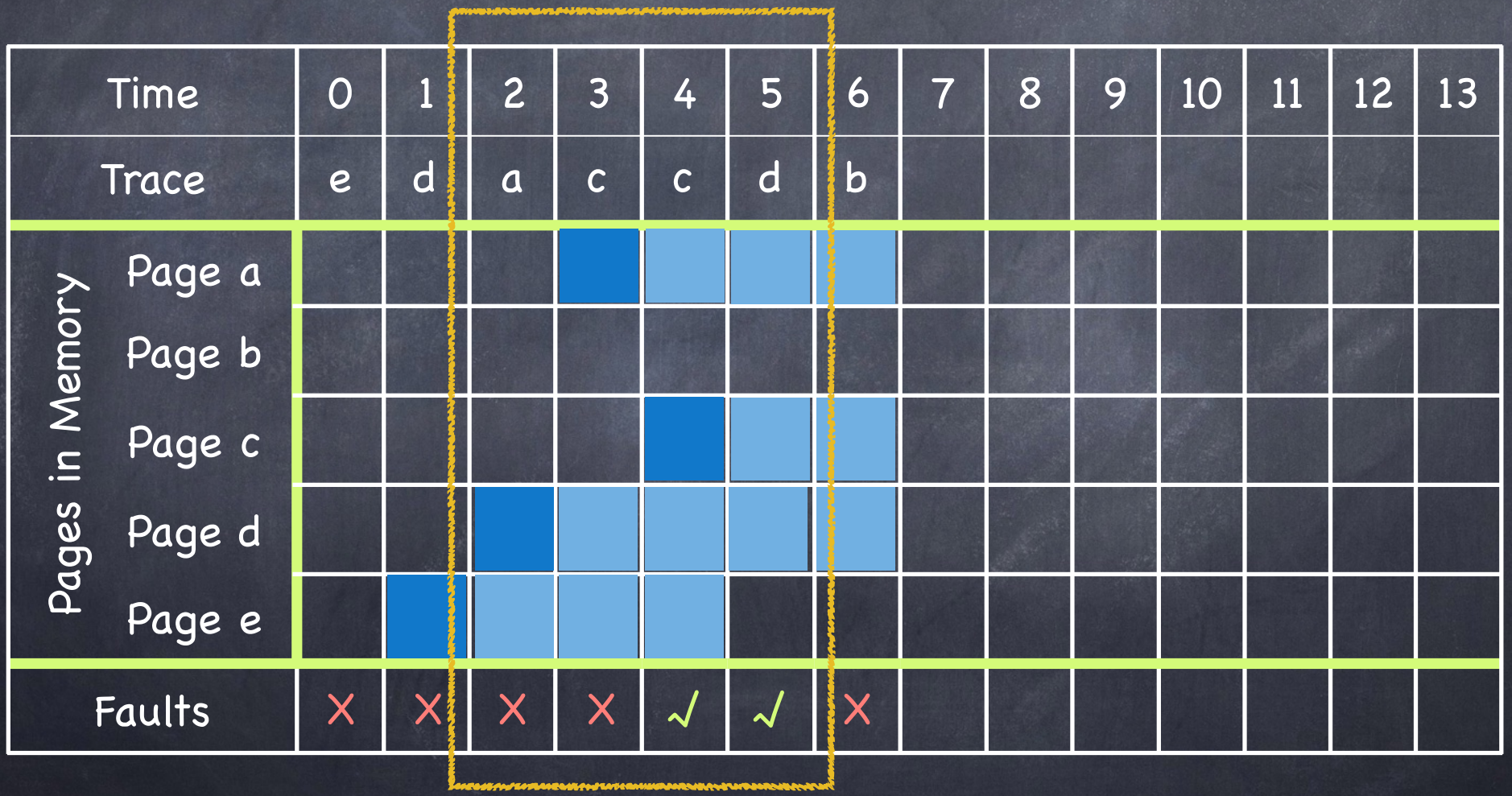
WS Page Replacement

$\Delta = 4$



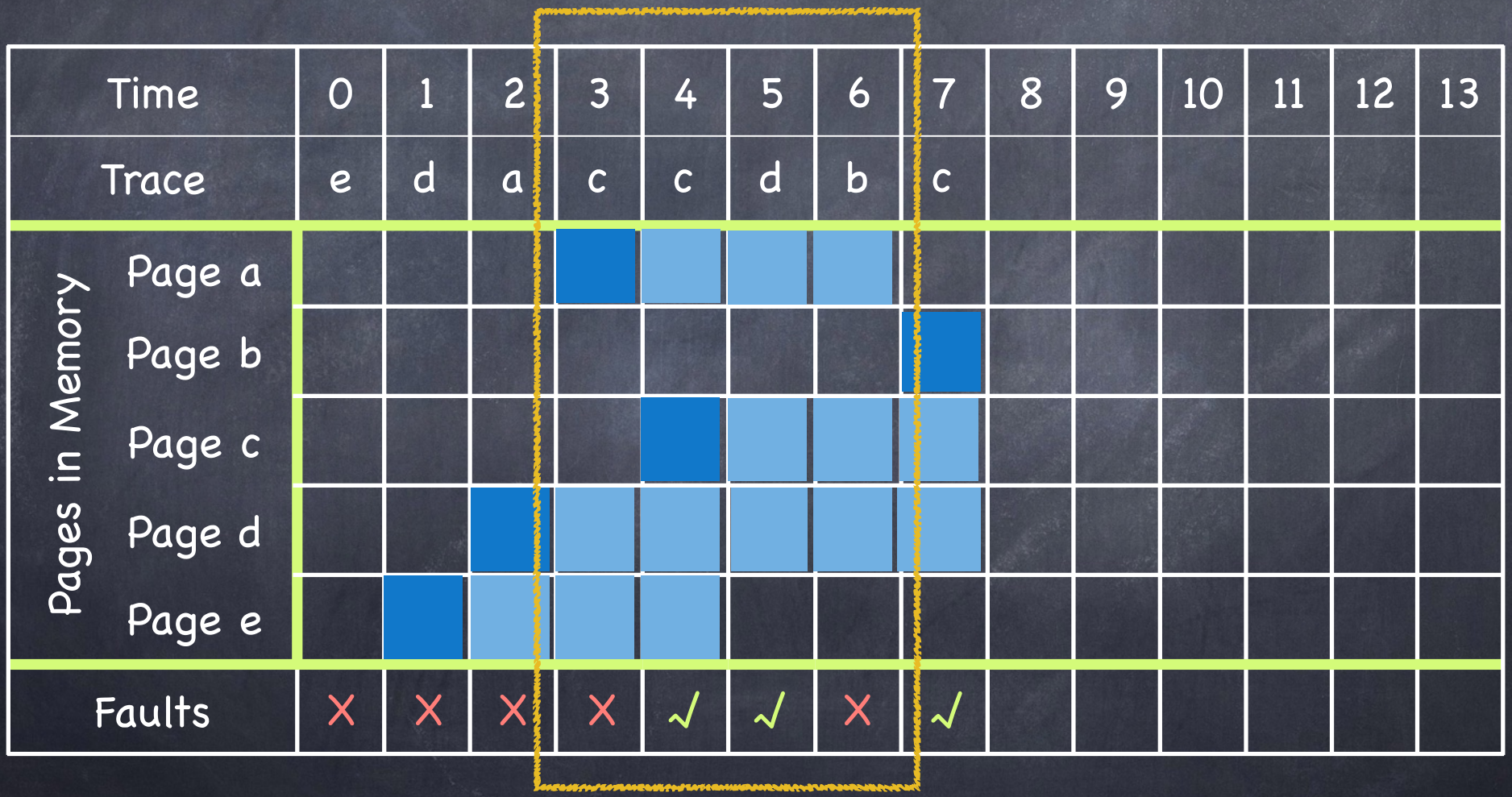
WS Page Replacement

$\Delta = 4$



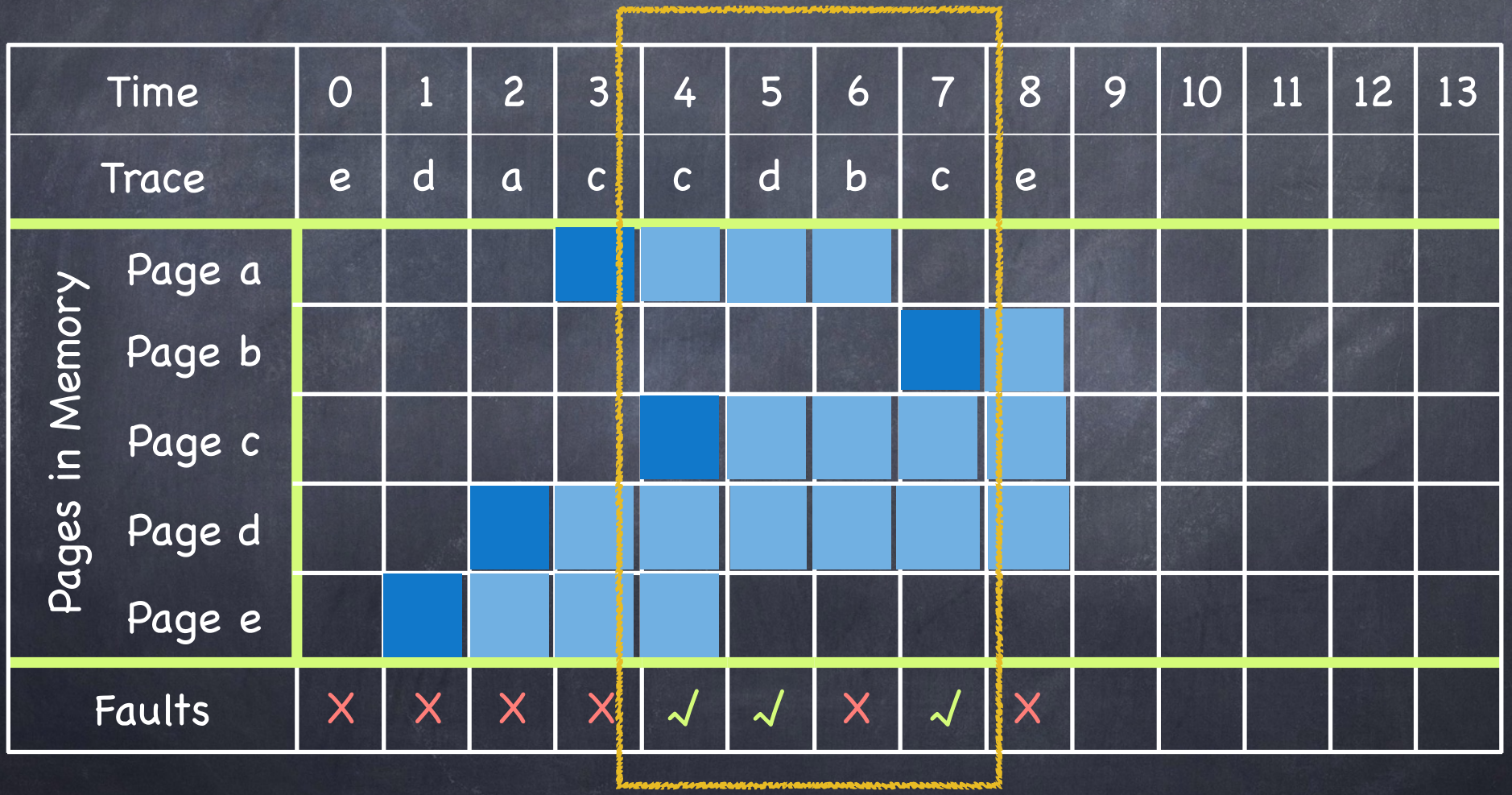
WS Page Replacement

$\Delta = 4$



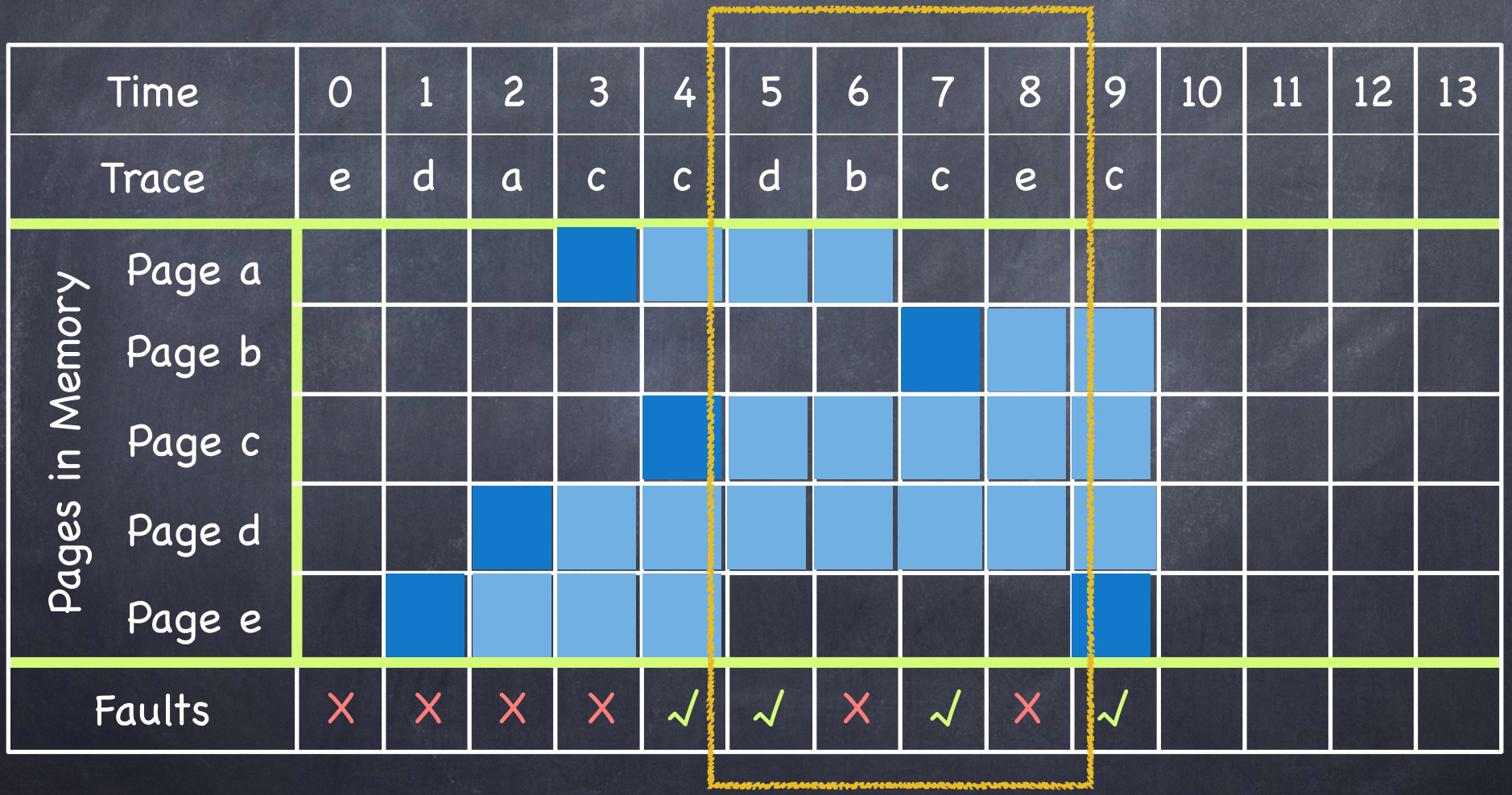
WS Page Replacement

$\Delta = 4$



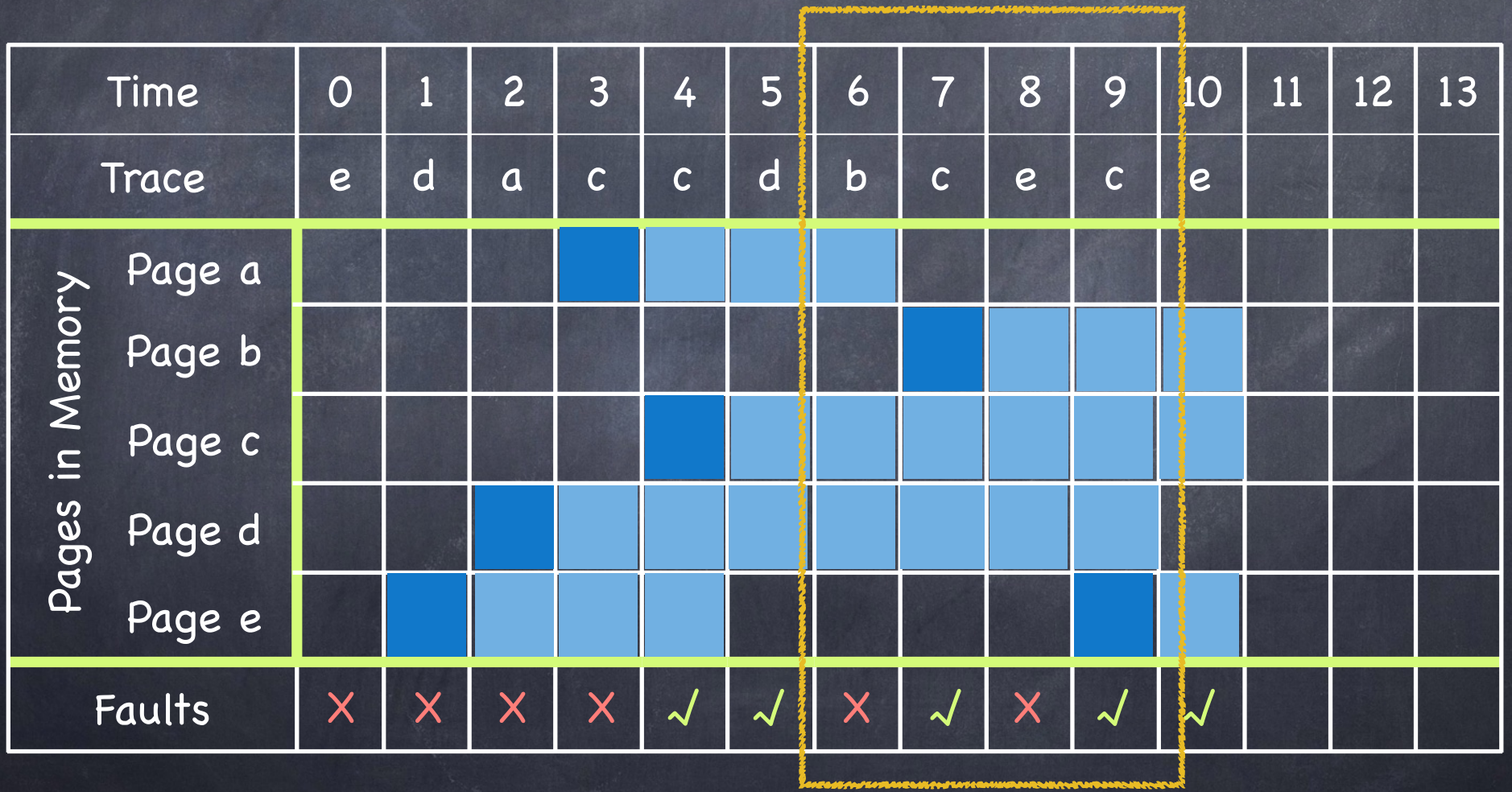
WS Page Replacement

$\Delta = 4$



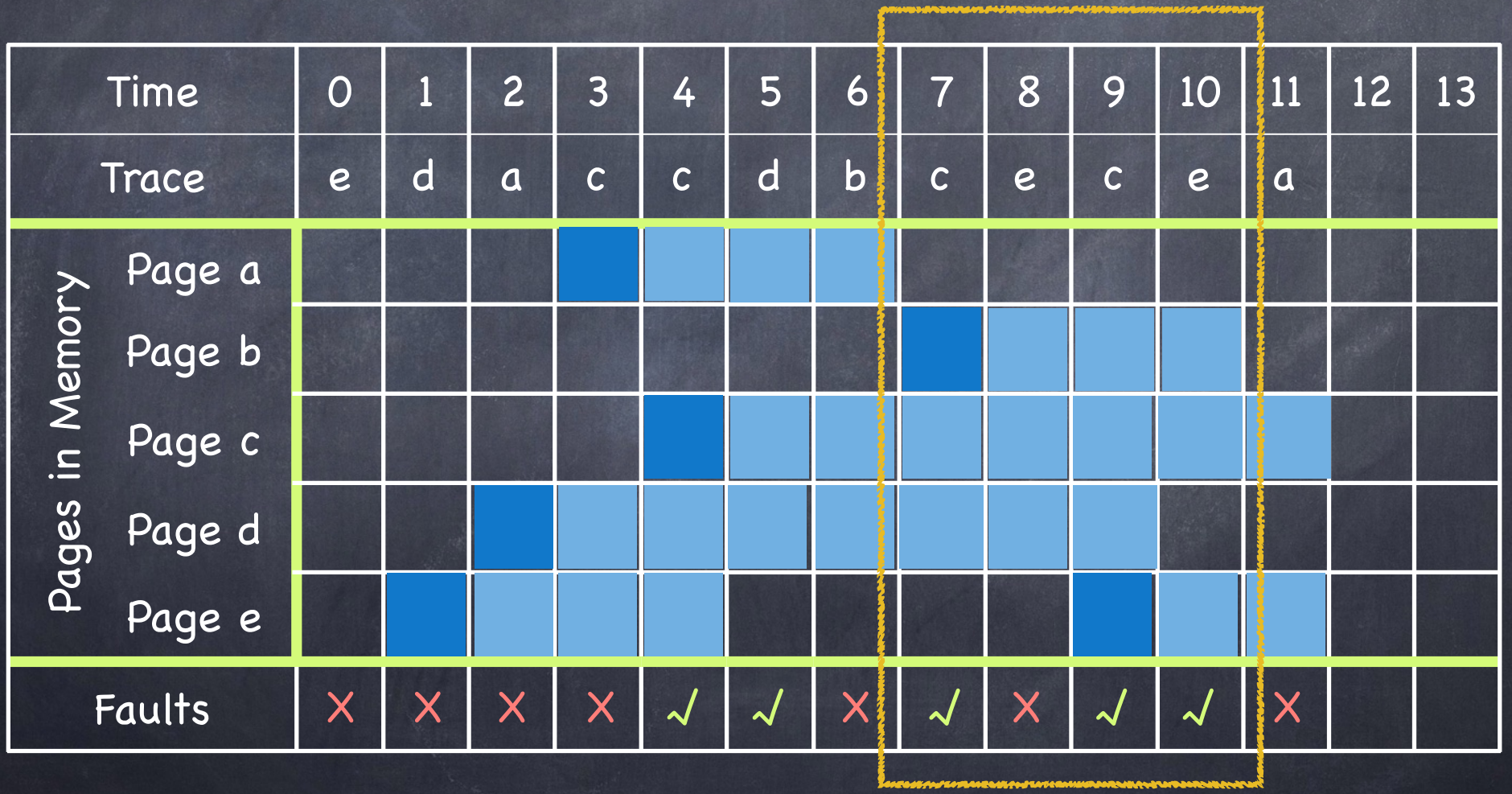
WS Page Replacement

$\Delta = 4$



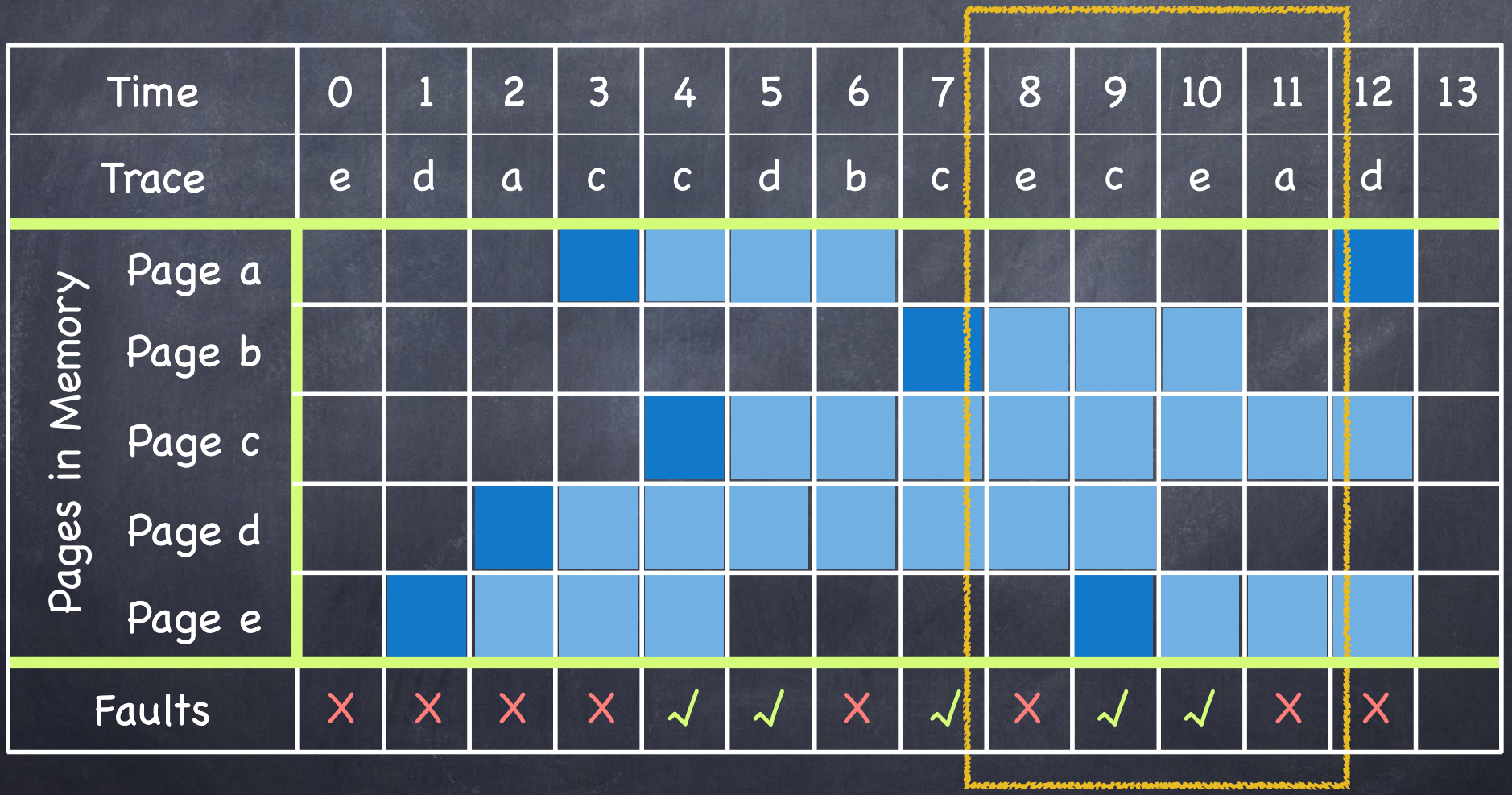
WS Page Replacement

$\Delta = 4$



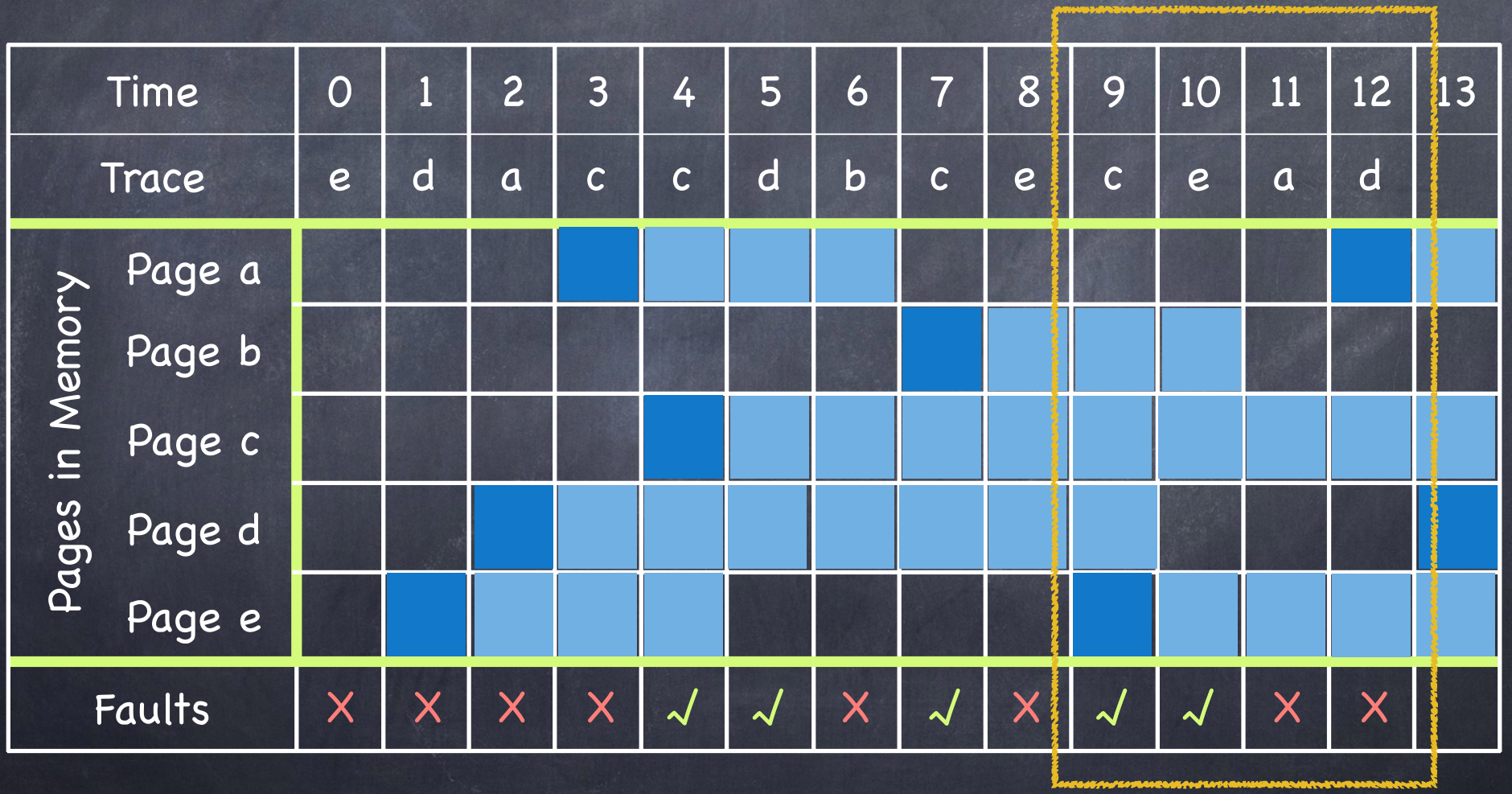
WS Page Replacement

$\Delta = 4$



WS Page Replacement

$\Delta = 4$



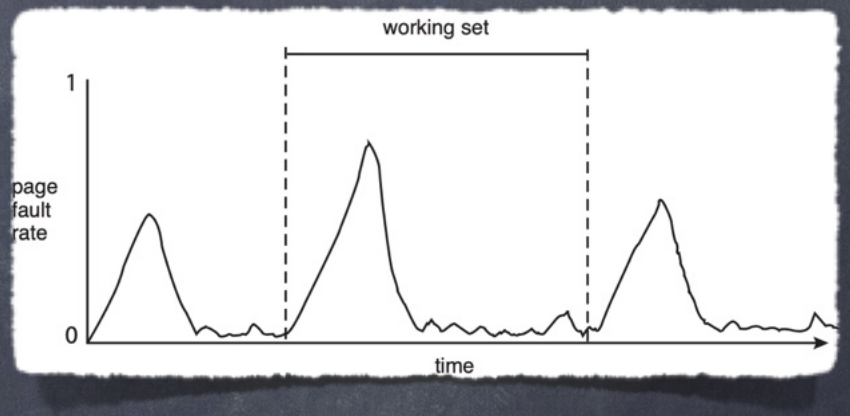
Approximating the Working Set

- Keep a k-bit tag in each page table entry (say, 2 bits)
- Set a timer interrupt to fire every Δ/k page references
  - if Δ = 10,000, then every 5000 references
  - can tell if page references in the last 15,000 references (2 bits + ref bit)
- On timer interrupt
  - Shift tag right one bit
  - Copy REF bit in tag's leftmost bit and clear REF
  - Add to a free list any page whose tag is zero
- When a frame is needed, use the free list (check also REF bit!)
  - if free list is empty, pick any frame

Note: Must scan all frames!

Working Sets and Page Fault Rates

- As the working set changes, the page fault rate increases
  - a steep increase in the page fault rate indicates a shift in locality, which may require a different WS



- Idea: Change the number of frames allocated to a process in response to changes to its Page Fault rate
  - as long as the working sets of all processes currently in memory does not exceed the size of physical memory, no thrashing

PFF (Page Fault Frequency) Algorithm

Keep time t_{last} of last page fault

On page fault:

if $t_{current} - t_{last} \leq \tau$, (threshold) then add faulting page to the working set

else unmap all pages not referenced in $[t_{last}, t_{current}]$

PFF Page Replacement

$$\tau = 2$$

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Trace	d	a	e	c	c	d	b	c	e	c	e	a	d	
Pages in Memory: Page a			■										■	
Pages in Memory: Page b								■						
Pages in Memory: Page c					■									
Pages in Memory: Page d		■												
Pages in Memory: Page e				■						■				
Faults	X	X	X	X			X		X			X	X	
$t_{\text{current}} - t_{\text{last}}$	0	1	1	1			3		2			3	1	