Memory Management:

Page Replacement Policies: FIFO, Random

CS 571: Operating Systems (Spring 2020)

Lecture 8c

Yue Cheng

Some material taken/derived from:

- Wisconsin CS-537 materials created by Remzi Arpaci-Dusseau.

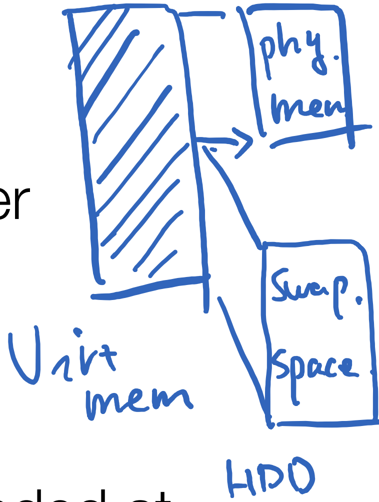
What to Evict?

Page Replacement

Mechanism

swapping partition

- Page replacement completes the separation between the logical memory and the physical memory
  - Large virtual memory can be provided on a smaller physical memory



- Impact on performance
  - If there are no free frames, two page transfers needed at each page fault!

victim

- We can use a modify (dirty) bit to reduce overhead of page transfers – only modified pages are written back to disk

Page Replacement Policy

- Formalizing the problem
  - Cache management: Physical memory is a cache for virtual memory pages in the system
  - Primary objective:
    - High performance
    - High efficiency
    - Low cost
  - Goal: Minimize cache misses
    - To minimize # times OS has to fetch a page from disk
    - -OR- maximize cache hits

Average Memory Access Time

- Average (or effective) memory access time (**AMAT**) is the metric to calculate the effective memory performance

$$AMAT = (P_{Hit} \cdot \underbrace{T_M}_{}) + (P_{Miss} \cdot T_D)$$

- T_M: Cost of accessing memory
- T_D: Cost of accessing disk
- P_{Hit}: Probability of finding data in cache (hit)
  - Hit rate
- P_{Miss}: Probability of not finding data in cache (miss)
  - Miss rate

An Example

- Assuming
  - T_M is 100 nanoseconds (ns), T_D is 10 milliseconds (ms)
  - P_{Hit} is 0.9, and P_{Miss} is 0.1

dominating factor ↓

- `AMAT = 0.9*100ns + 0.1*10ms = 90ns + 1ms = 1.00009ms`
  - Or around 1 millisecond

90%

- What if the hit rate is 99.9%?
  - Result changes to 10.1 microseconds (or `us`)
  - Roughly 100 times faster!

First-In First-Out (FIFO)

First-in First-out (FIFO)

- Simplest page replacement algorithm

- Idea: items are evicted in the order they are inserted

- Implementation: FIFO queue holds identifiers of all the pages in memory
  - We replace the page at the head of the queue
  - When a page is brought into memory, it is inserted at the tail of the queue

FIFO Replacement Policy

- Idea: items are evicted in the order they are inserted
- Example workload: 0 1 2 0 1 3 0 3 1 2 1 ...

FIFO Replacement Policy

- Idea: items are evicted in the order they are inserted
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

Access	Hit/Miss?	Evict	Resulting Cache State
0			
1			
2			
0			
1			
3			
0			
3			
1			
2			
1			

assume cache size 3

FIFO Replacement Policy

- Idea: items are evicted in the order they are inserted
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

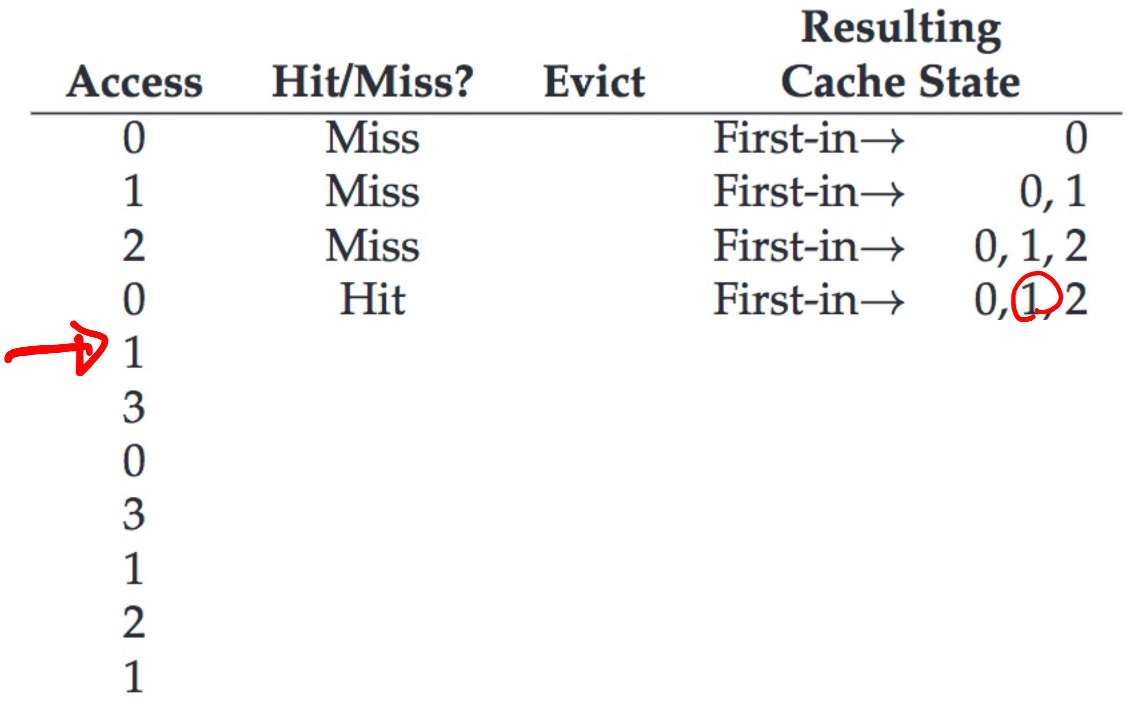
compulsory misses

assume cache size 3

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		First-in→ 0
1	Miss		First-in→ 0, 1
2	Miss		First-in→ 0, 1, 2
0			
1			
3			
0			
3			
1			
2			
1			

FIFO Replacement Policy

- Idea: items are evicted in the order they are inserted
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

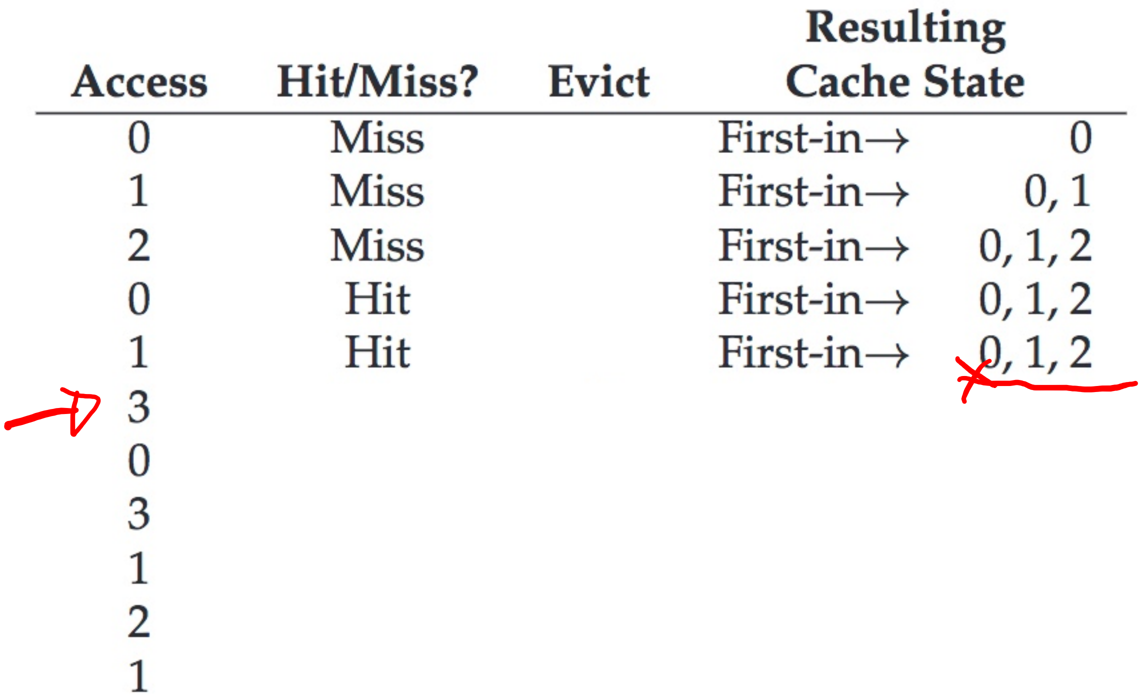


Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		First-in→ 0
1	Miss		First-in→ 0, 1
2	Miss		First-in→ 0, 1, 2
0	Hit		First-in→ 0, 1, 2
1			
3			
0			
3			
1			
2			
1			

assume cache size 3

FIFO Replacement Policy

- Idea: items are evicted in the order they are inserted
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

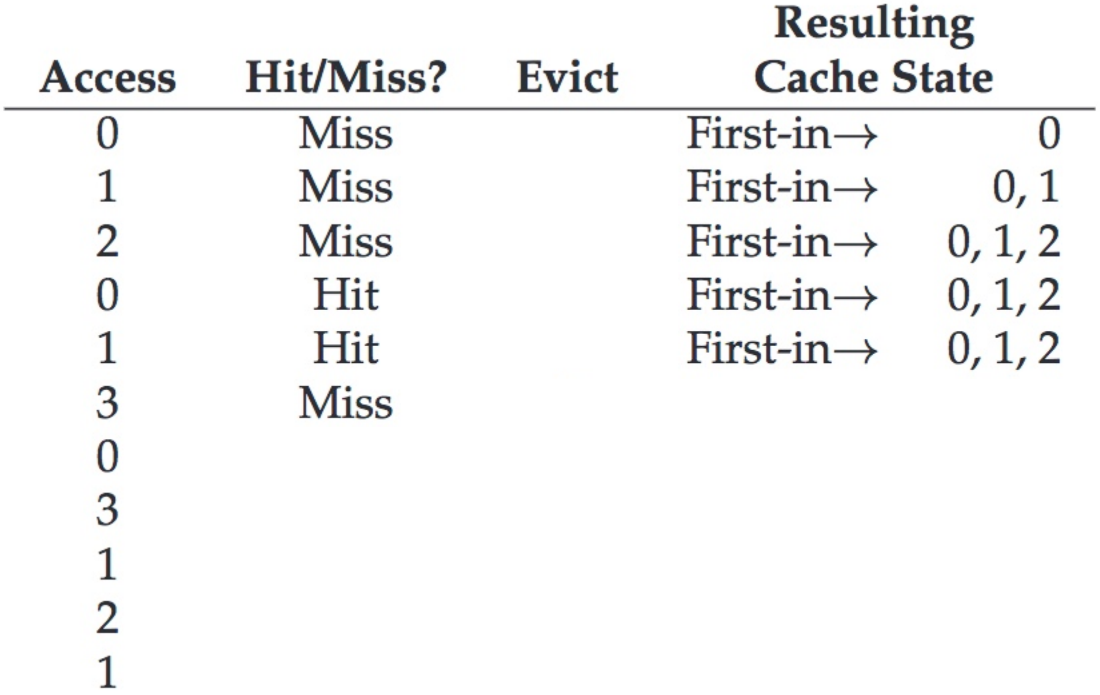


Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		First-in→ 0
1	Miss		First-in→ 0, 1
2	Miss		First-in→ 0, 1, 2
0	Hit		First-in→ 0, 1, 2
1	Hit		First-in→ 0, 1, 2
3			
0			
3			
1			
2			
1			

assume cache size 3

FIFO Replacement Policy

- Idea: items are evicted in the order they are inserted
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

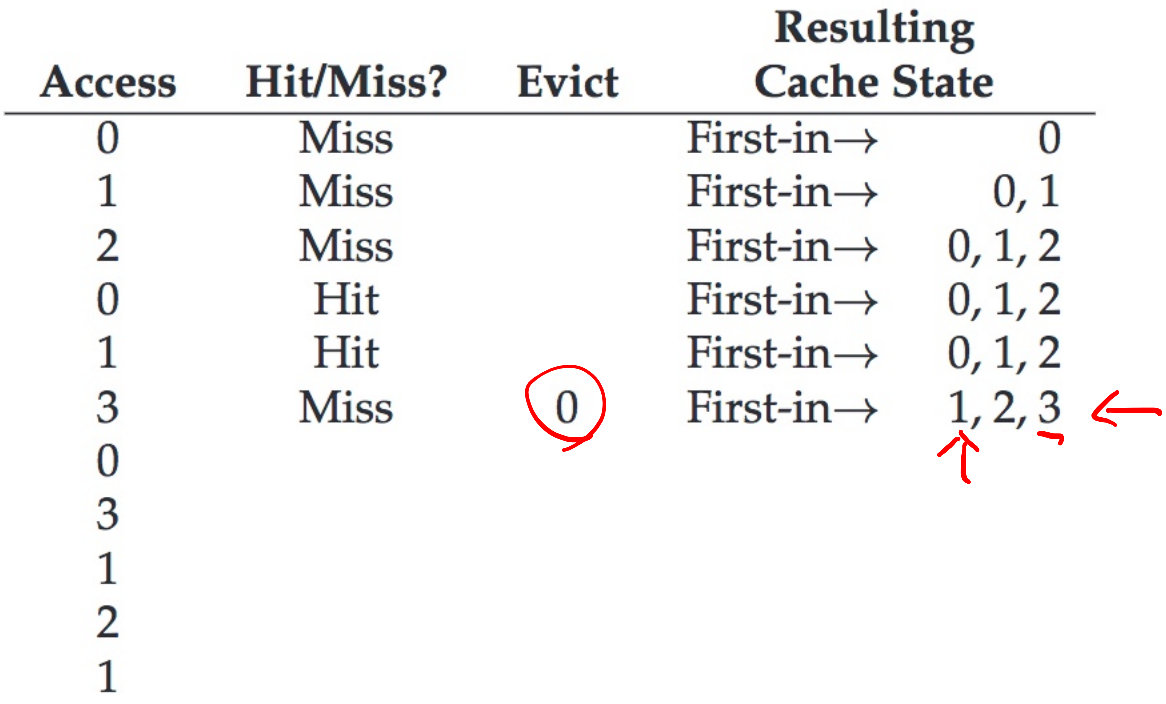


Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		First-in→ 0
1	Miss		First-in→ 0, 1
2	Miss		First-in→ 0, 1, 2
0	Hit		First-in→ 0, 1, 2
1	Hit		First-in→ 0, 1, 2
3	Miss		
0			
3			
1			
2			
1			

assume cache size 3

FIFO Replacement Policy

- Idea: items are evicted in the order they are inserted
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

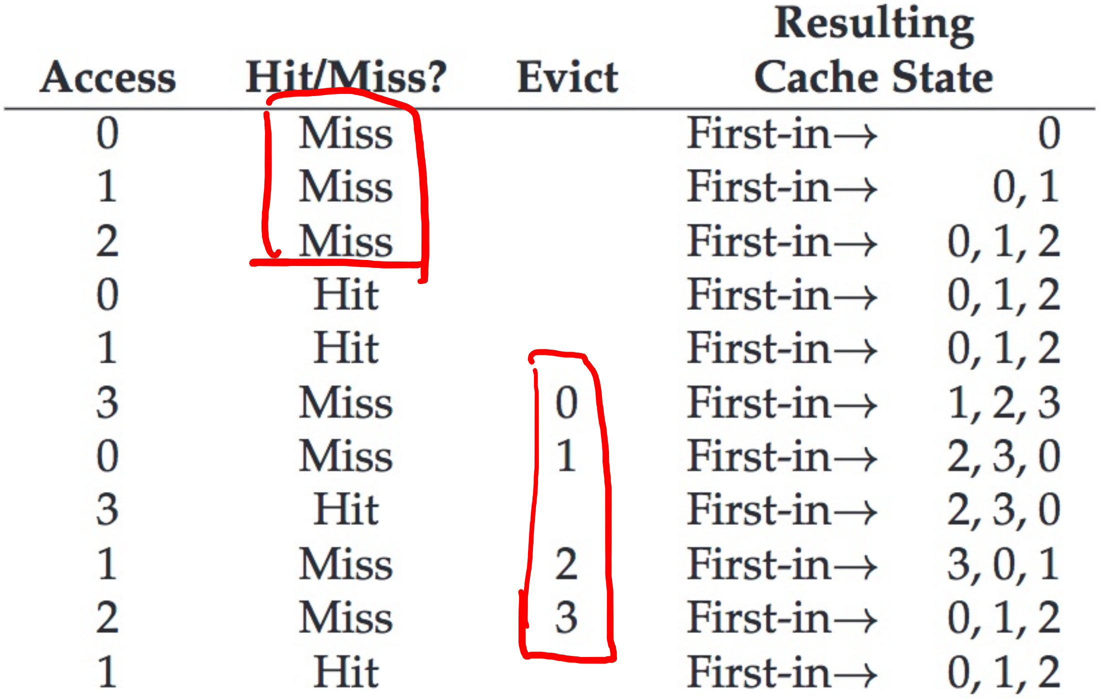


Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		First-in→ 0
1	Miss		First-in→ 0, 1
2	Miss		First-in→ 0, 1, 2
0	Hit		First-in→ 0, 1, 2
1	Hit		First-in→ 0, 1, 2
3	Miss	0	First-in→ 1, 2, 3
0			
3			
1			
2			
1			

assume cache size 3

FIFO Replacement Policy

- Idea: items are evicted in the order they are inserted
- Example workload: 0 1 2 0 1 3 0 3 1 2 1



Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		First-in→ 0
1	Miss		First-in→ 0, 1
2	Miss		First-in→ 0, 1, 2
0	Hit		First-in→ 0, 1, 2
1	Hit		First-in→ 0, 1, 2
3	Miss	0	First-in→ 1, 2, 3
0	Miss	1	First-in→ 2, 3, 0
3	Hit		First-in→ 2, 3, 0
1	Miss	2	First-in→ 3, 0, 1
2	Miss	3	First-in→ 0, 1, 2
1	Hit		First-in→ 0, 1, 2

assume cache size 3

FIFO Replacement Policy

- Idea: items are evicted in the order they are inserted

- Issue: the “oldest” page may contain a heavily used data
  - Will need to bring back that page in near future

FIFO Replacement Policy

- FIFO: items are evicted in the order they are inserted
- Example workload: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

(a) size 3

Access	Hit	State (after)
1		
2		
3		
4		
1		
2		
5		
1		
2		
3		
4		
5		

(b) size 4

Access	Hit	State (after)
1		
2		
3		
4		
1		
2		
5		
1		
2		
3		
4		
5		

FIFO Replacement Policy

- FIFO: items are evicted in the order they are inserted
- Example workload: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

(a) size 3 3 hits.

Access	Hit	State (after)
1	no	1
2	no	1,2
3	no	1,2,3
4	no	2,3,4
1	no	3,4,1
2	no	4,1,2
5	no	1,2,5
1	yes	1,2,5
2	yes	1,2,5
3	no	2,5,3
4	no	5,3,4
5	yes	5,3,4

(b) size 4

Access	Hit	State (after)
1		
2		
3		
4		
1		
2		
5		
1		
2		
3		
4		
5		

FIFO Replacement Policy

- FIFO: items are evicted in the order they are inserted
- Example workload: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

(a) size 3

Access	Hit	State (after)
1	no	1
2	no	1,2
3	no	1,2,3
4	no	2,3,4
1	no	3,4,1
2	no	4,1,2
5	no	1,2,5
1	yes	1,2,5
2	yes	1,2,5
3	no	2,5,3
4	no	5,3,4
5	yes	5,3,4

(b) size 4

Access	Hit	State (after)
1	no	1
2	no	1,2
3	no	1,2,3
4	no	1,2,3,4
1	yes	1,2,3,4
2	yes	1,2,3,4
5	no	2,3,4,5
1	no	3,4,5,1
2	no	4,5,1,2
3	no	5,1,2,3
4	no	1,2,3,4
5	no	2,3,4,5

Belady's Anomaly

Δ

Working set size. (WSS)

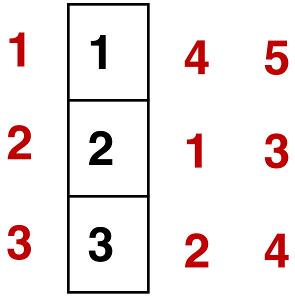
Sz(WSS) = 5 {1, 2, 3, 4, 5}.

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
  - Size-3 (3-frames) case results in 9 page faults
  - Size-4 (4-frames) case results in 10 page faults

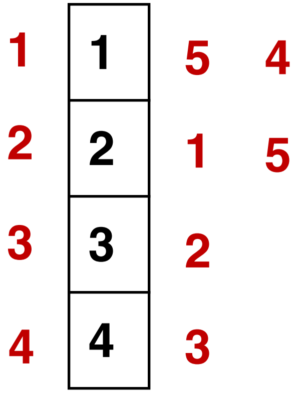
Δ

- Program runs potentially slower w/ more memory!

- Belady's anomaly
  - More frames ➔ more page faults for some access pattern



9 page faults



10 page faults

Random

Random Policy

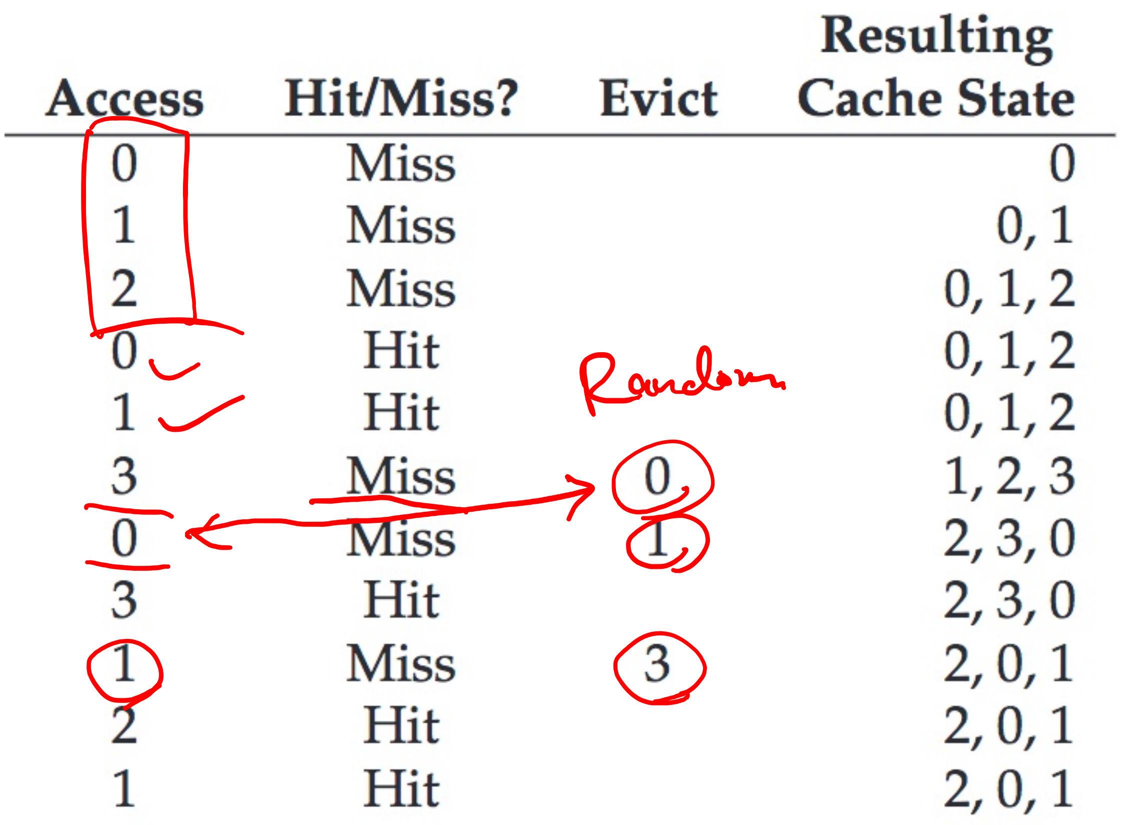
- Idea: picks a random page to replace

- Simple to implement like FIFO

- No intelligence of preserving locality

Random Policy

- Idea: picks a random page to replace
- Example workload: 0 1 2 0 1 3 0 3 1 2 1



Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0, 1
2	Miss		0, 1, 2
0	Hit		0, 1, 2
1	Hit		0, 1, 2
3	Miss	0	1, 2, 3
0	Miss	1	2, 3, 0
3	Hit		2, 3, 0
1	Miss	3	2, 0, 1
2	Hit		2, 0, 1
1	Hit		2, 0, 1

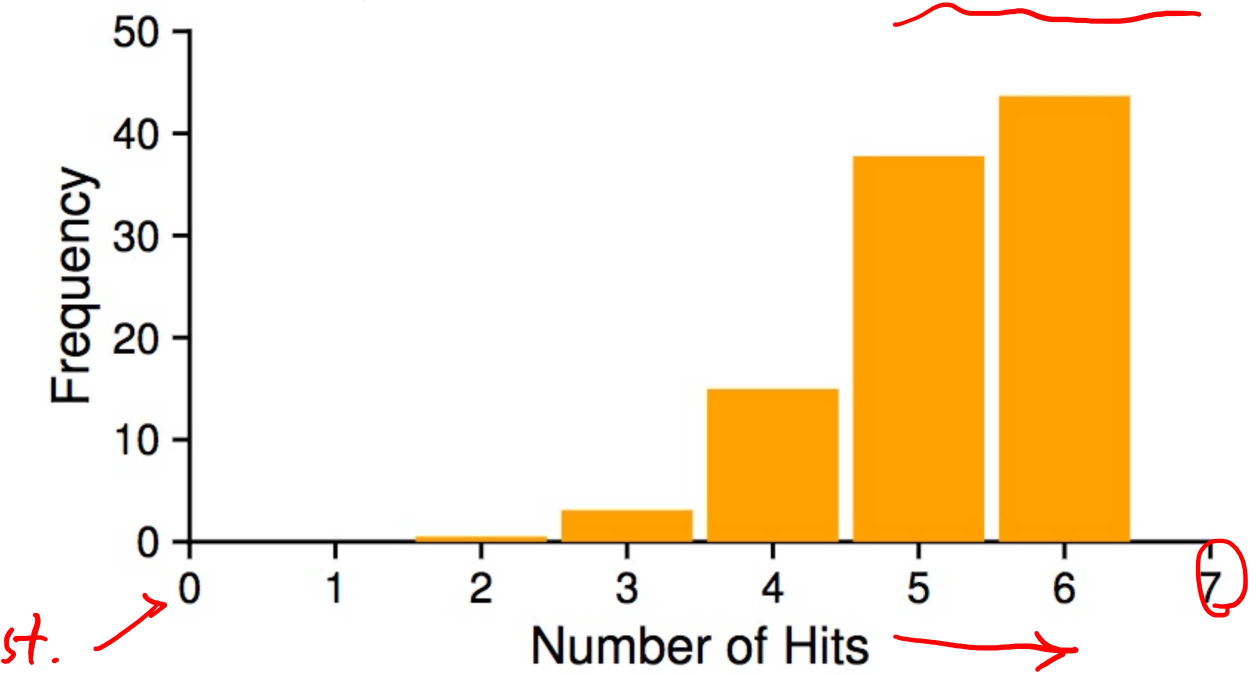
Random

assume
cache size 3

How Random Policy Performs?

- Depends entirely on how lucky you are
- Example workload: 0 1 2 0 1 3 0 3 0 1 2 1

Random performance over 10000 trials



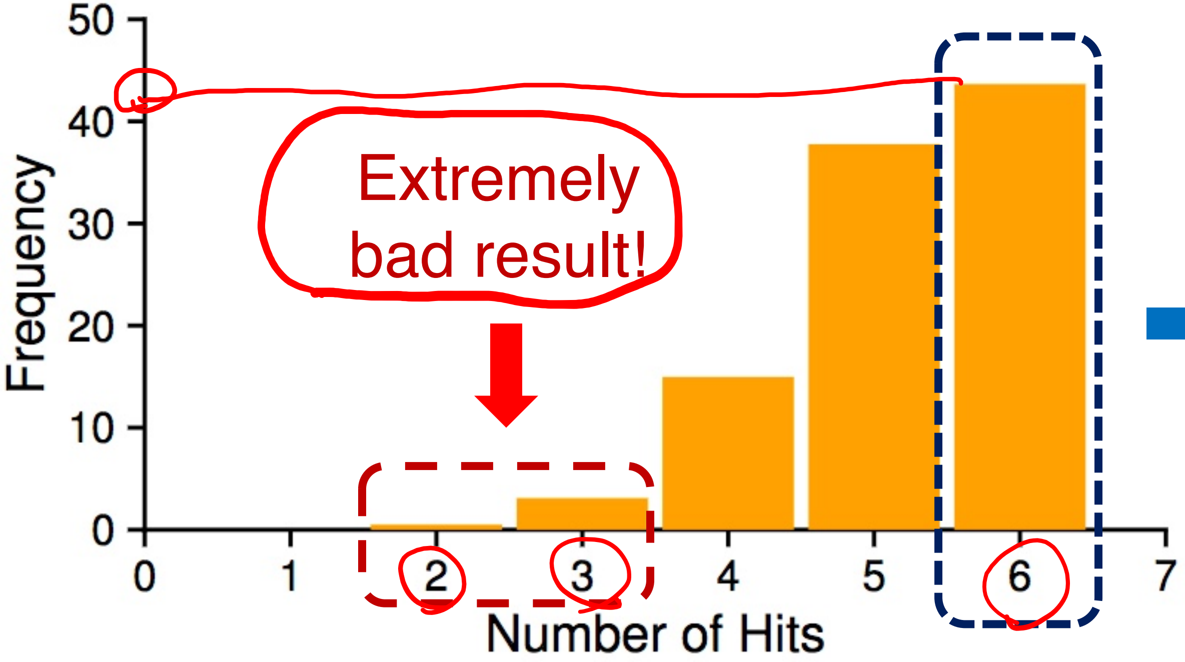
worst.

best.

How Random Policy Performs?

- Depends entirely on how lucky you are
- Example workload: 0 1 2 0 1 3 0 3 0 1 2 1

Random performance over 10000 trials



Extremely bad result!

Same as optimal