Memory Management:

Page Replacement Policies: Belady's Optimal

CS 571: Operating Systems (Spring 2020)

Lecture 8c

Yue Cheng

Some material taken/derived from:

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

Belady's Optimal

OPT: The Optimal Replacement Policy

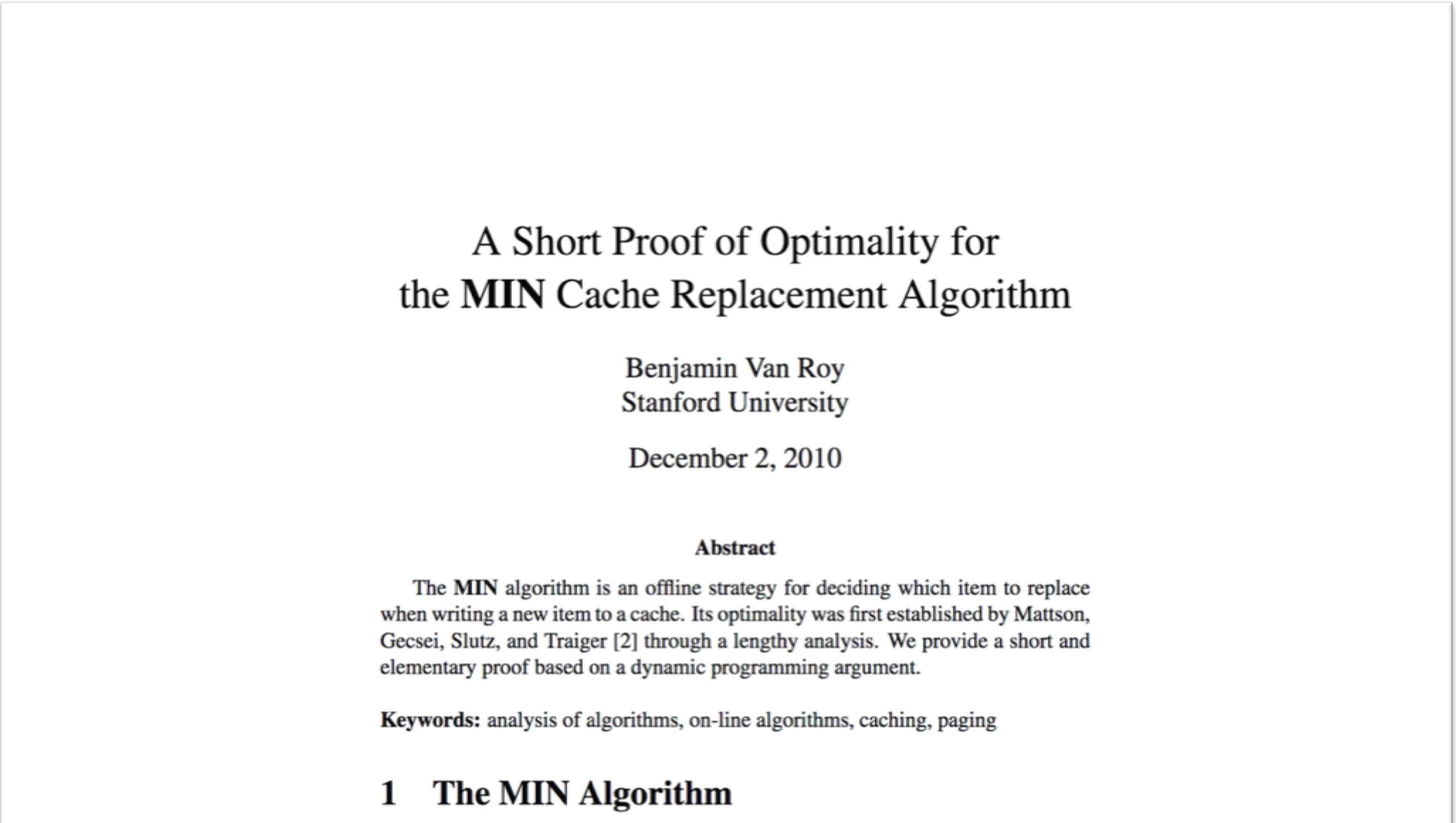
offline:

- Many years ago Belady demonstrated that there is a simple policy (OPT or MIN) which always leads to fewest number of misses
- Idea: evict the page that will be accessed furthest in the future
- Assumption: we know about the future
- Impossible to implement OPT in practice!

- But it is extremely useful as a practical best-case baseline for comparison purpose

Proof of Optimality for Belady's Optimal Replacement Policy

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.307.7603&rep=rep1&type=pdf



A Short Proof of Optimality for the **MIN** Cache Replacement Algorithm

Benjamin Van Roy
Stanford University

December 2, 2010

Abstract

The **MIN** algorithm is an offline strategy for deciding which item to replace when writing a new item to a cache. Its optimality was first established by Mattson, Gecsei, Slutz, and Traiger [2] through a lengthy analysis. We provide a short and elementary proof based on a dynamic programming argument.

Keywords: analysis of algorithms, on-line algorithms, caching, paging

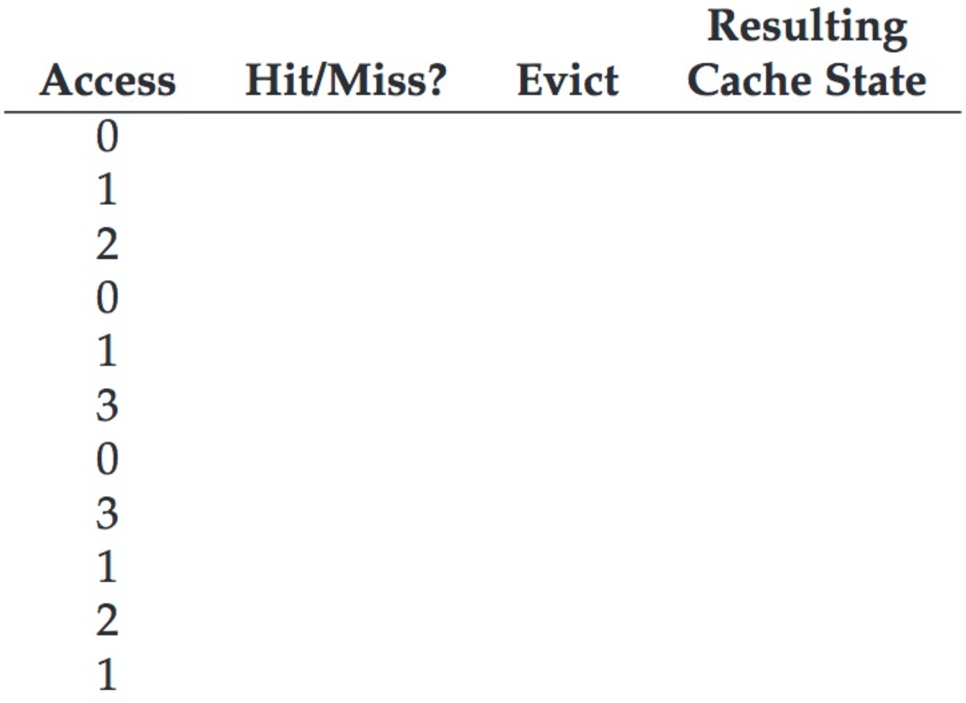
1 The MIN Algorithm

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- 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

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

compulsory misses.

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

assume cache size 3

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

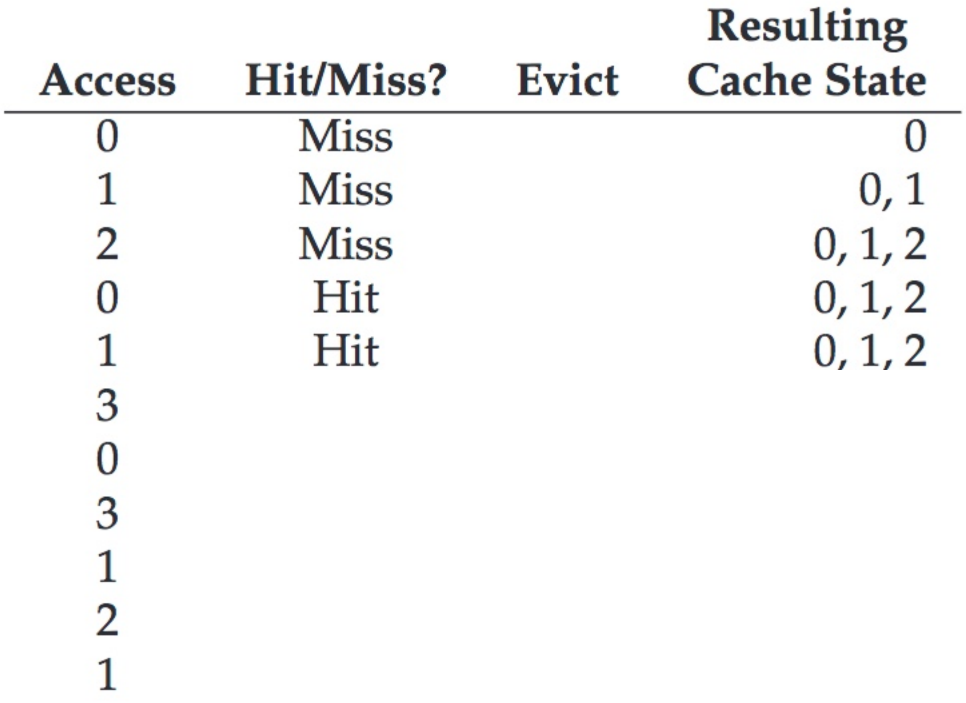
assume cache size 3

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	2	0 1 3
0 ✓			
3			
1 ✓			
2 ✓			
1			

most distant ref.

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- Example workload: 0 1 2 0 1 3 0 3 1 2 1



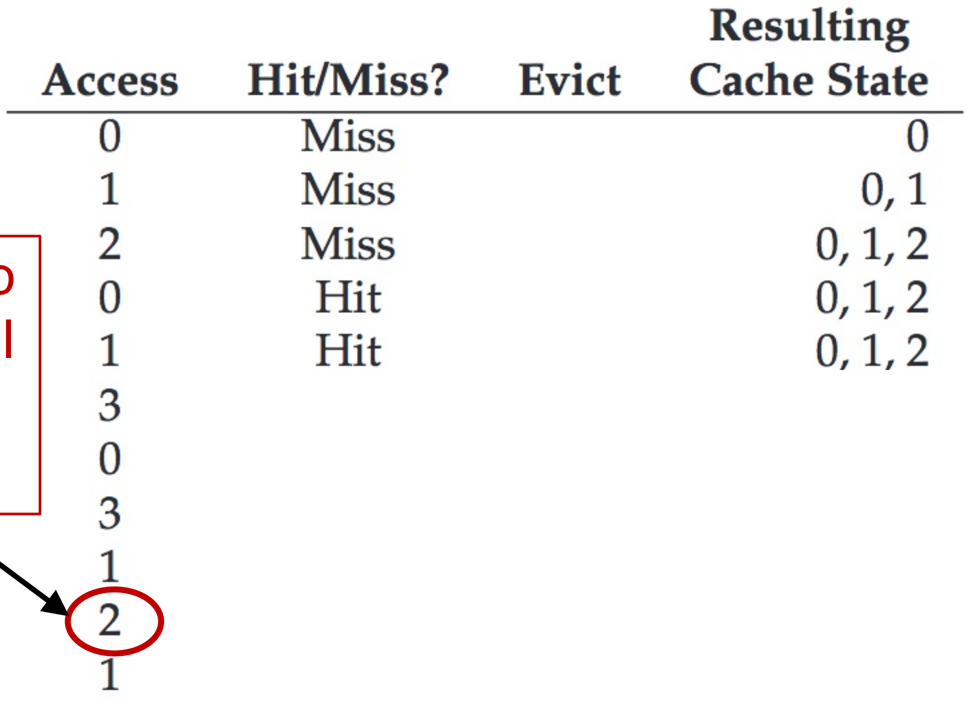
Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0, 1
2	Miss		0, 1, 2
0	Hit		0, 1, 2
1	Hit		0, 1, 2
3			
0			
3			
1			
2			
1			

assume
cache size 3

What to evict??

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- 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			
0			
3			
1			
2			
1			

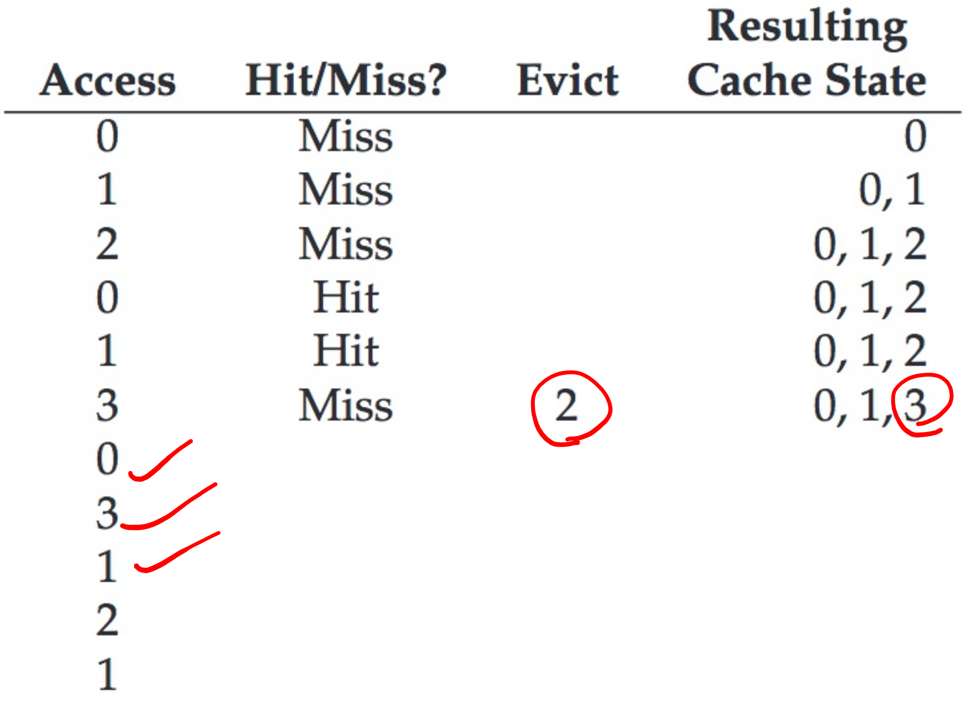
assume cache size 3

Page 2 happens to be the one that will be accessed furthest in future!

What to evict??

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- Example workload: 0 1 2 0 1 3 0 3 1 2 1



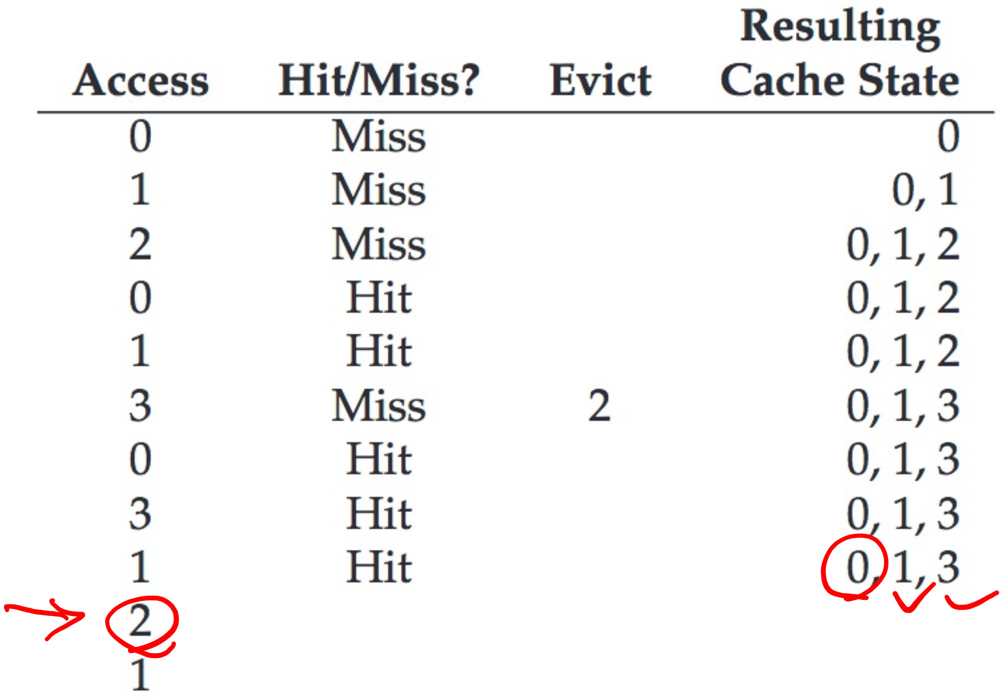
Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0, 1
2	Miss		0, 1, 2
0	Hit		0, 1, 2
1	Hit		0, 1, 2
3	Miss	2	0, 1, 3
0 ✓			
3 ✓			
1 ✓			
2			
1			

assume cache size 3

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

end of trace



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	2	0, 1, 3
0	Hit		0, 1, 3
3	Hit		0, 1, 3
1	Hit		0, 1, 3
2			
1			

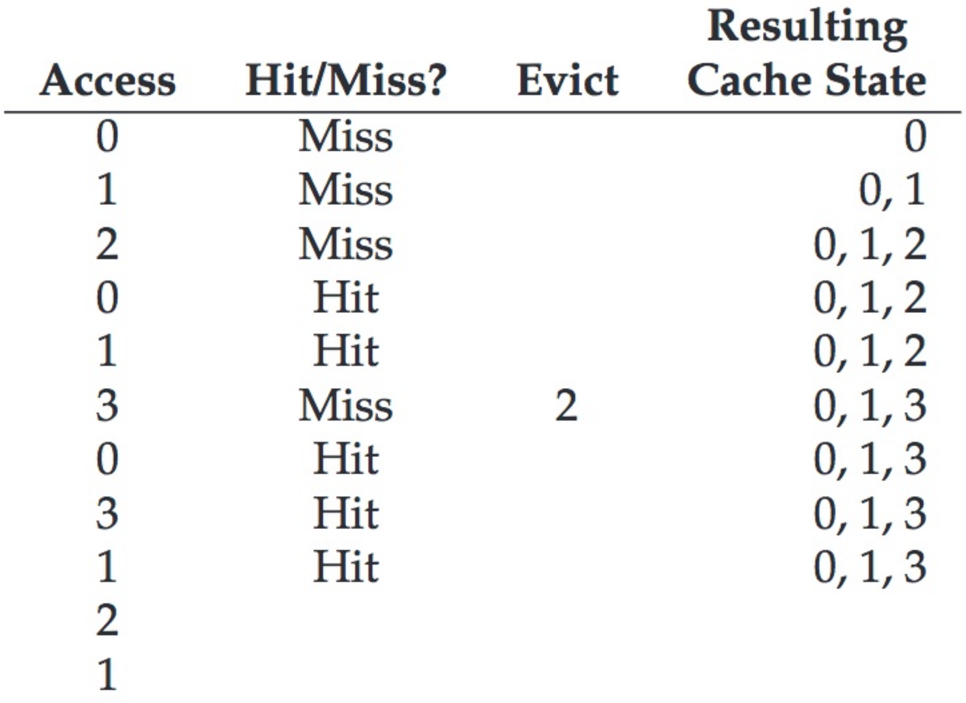
:3

assume cache size 3

random policy 50%

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- 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	2	0, 1, 3
0	Hit		0, 1, 3
3	Hit		0, 1, 3
1	Hit		0, 1, 3
2			
1			

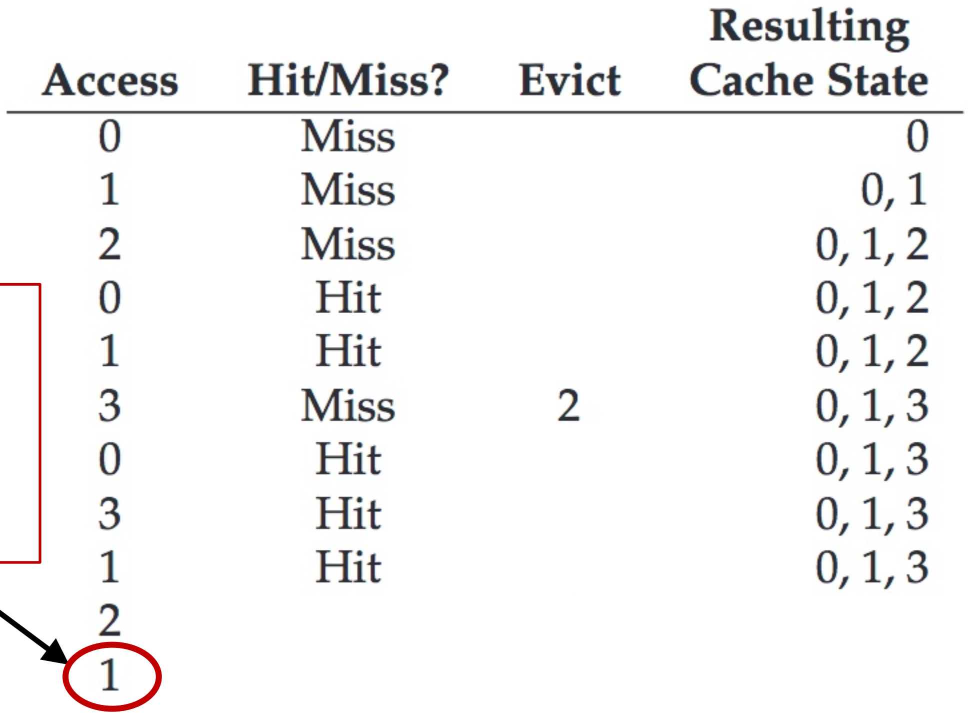
assume
cache size 3

What to evict??

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

assume cache size 3



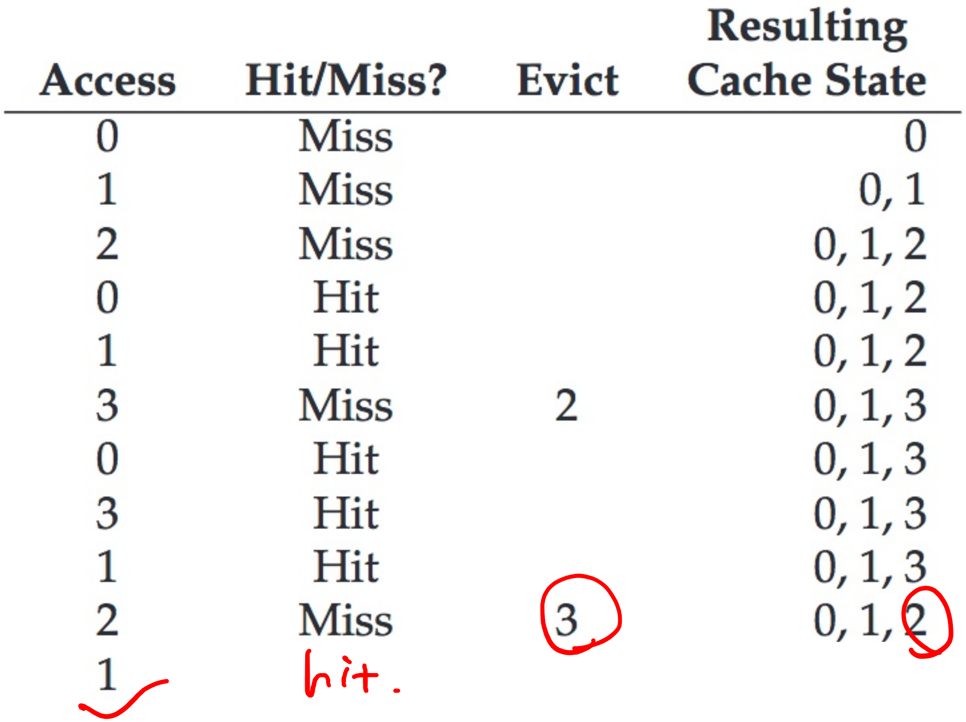
Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0, 1
2	Miss		0, 1, 2
0	Hit		0, 1, 2
1	Hit		0, 1, 2
3	Miss	2	0, 1, 3
0	Hit		0, 1, 3
3	Hit		0, 1, 3
1	Hit		0, 1, 3
2			
1			

Page 1 will be accessed right after page 2. Hence 1 is safe!

What to evict??

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

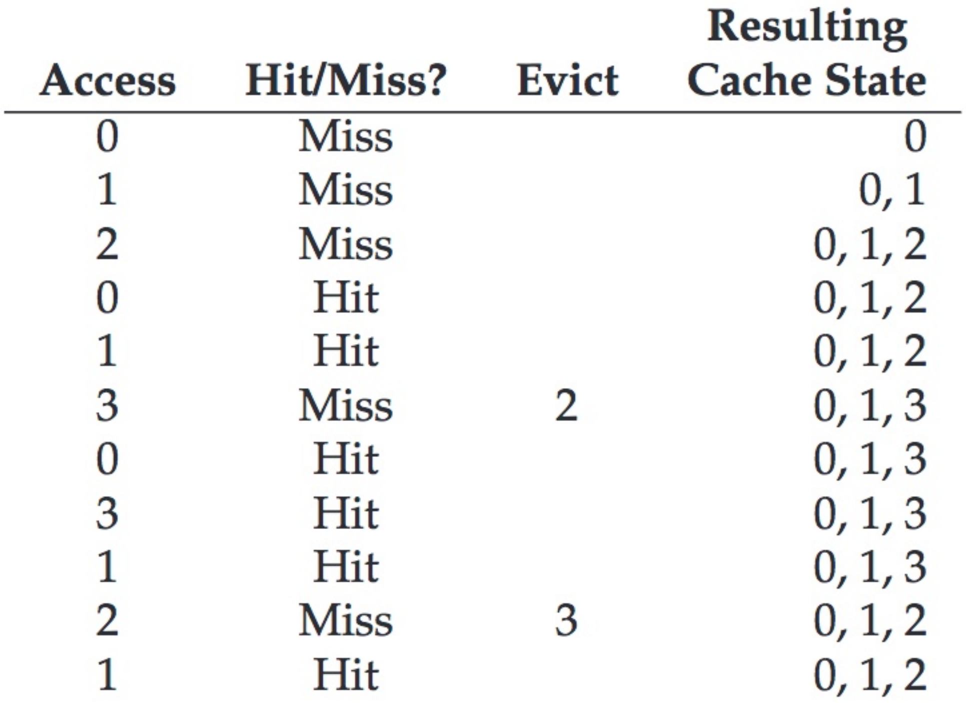


Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0, 1
2	Miss		0, 1, 2
0	Hit		0, 1, 2
1	Hit		0, 1, 2
3	Miss	2	0, 1, 3
0	Hit		0, 1, 3
3	Hit		0, 1, 3
1	Hit		0, 1, 3
2	Miss	3	0, 1, 2
1 ✓	hit.		

assume cache size 3

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- Example workload: 0 1 2 0 1 3 0 3 1 2 1

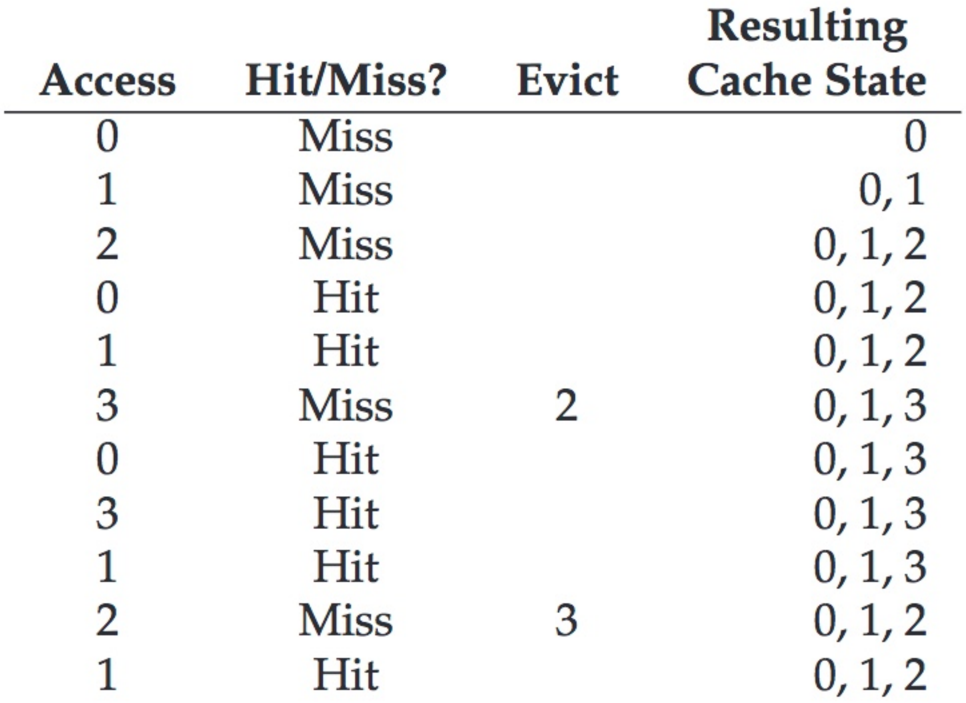


Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0, 1
2	Miss		0, 1, 2
0	Hit		0, 1, 2
1	Hit		0, 1, 2
3	Miss	2	0, 1, 3
0	Hit		0, 1, 3
3	Hit		0, 1, 3
1	Hit		0, 1, 3
2	Miss	3	0, 1, 2
1	Hit		0, 1, 2

assume
cache size 3

OPT the Optimal

- Idea: evict the page that will be accessed furthest in the future
- 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	2	0, 1, 3
0	Hit		0, 1, 3
3	Hit		0, 1, 3
1	Hit		0, 1, 3
2	Miss	3	0, 1, 2
1	Hit		0, 1, 2

assume
cache size 3

The optimal number of cache hits is **6** for this workload!