Announcements

- Exam 2 on F
  - Please let me know if you need to take the exam at an alternate time and did not need to for exam 1.
  - Please complete the exam instructions assignment on gradescope before the exam
- No HW this week.

Last Time

- Greedy Algorithms
  - Find decision Procedure
  - Repeatedly make best available choice
  - Repeat until done
- Interval Scheduling
- Proofs are Important
- Exchange arguments

Exchange Argument

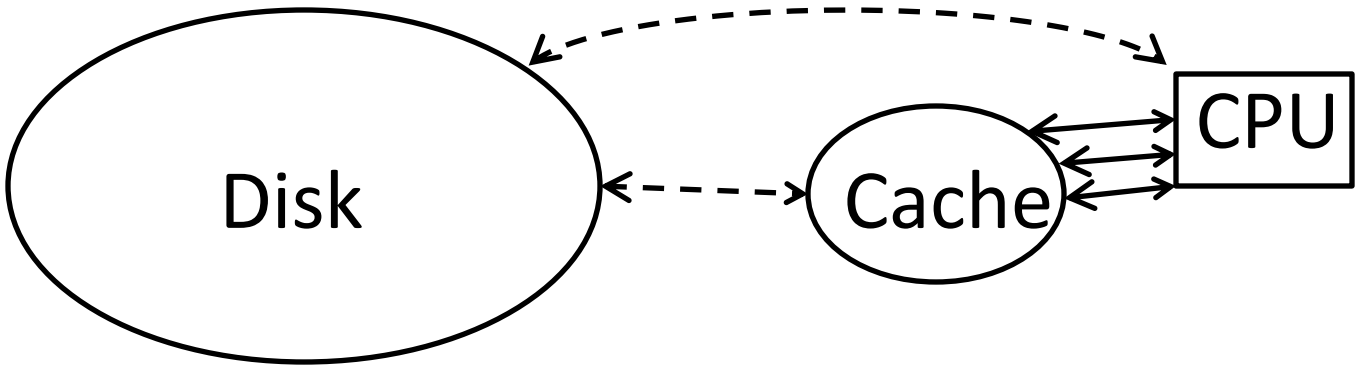
- Greedy algorithm makes a sequence of decisions D_1, D_2, D_3,…,D_n eventually reaching solution G.
- For arbitrary solution A:
- Find sequence of solutions
  $A=A_0, A_1, A_2,\ldots,A_n = G$
  so that:
  - $A_i \leq A_{i+1}$
  - A_i agrees with $D_1,D_2,\ldots,D_i$
- Use A_i to construct A_{i+1}.

Today

- Optimal Caching
- Huffman Codes

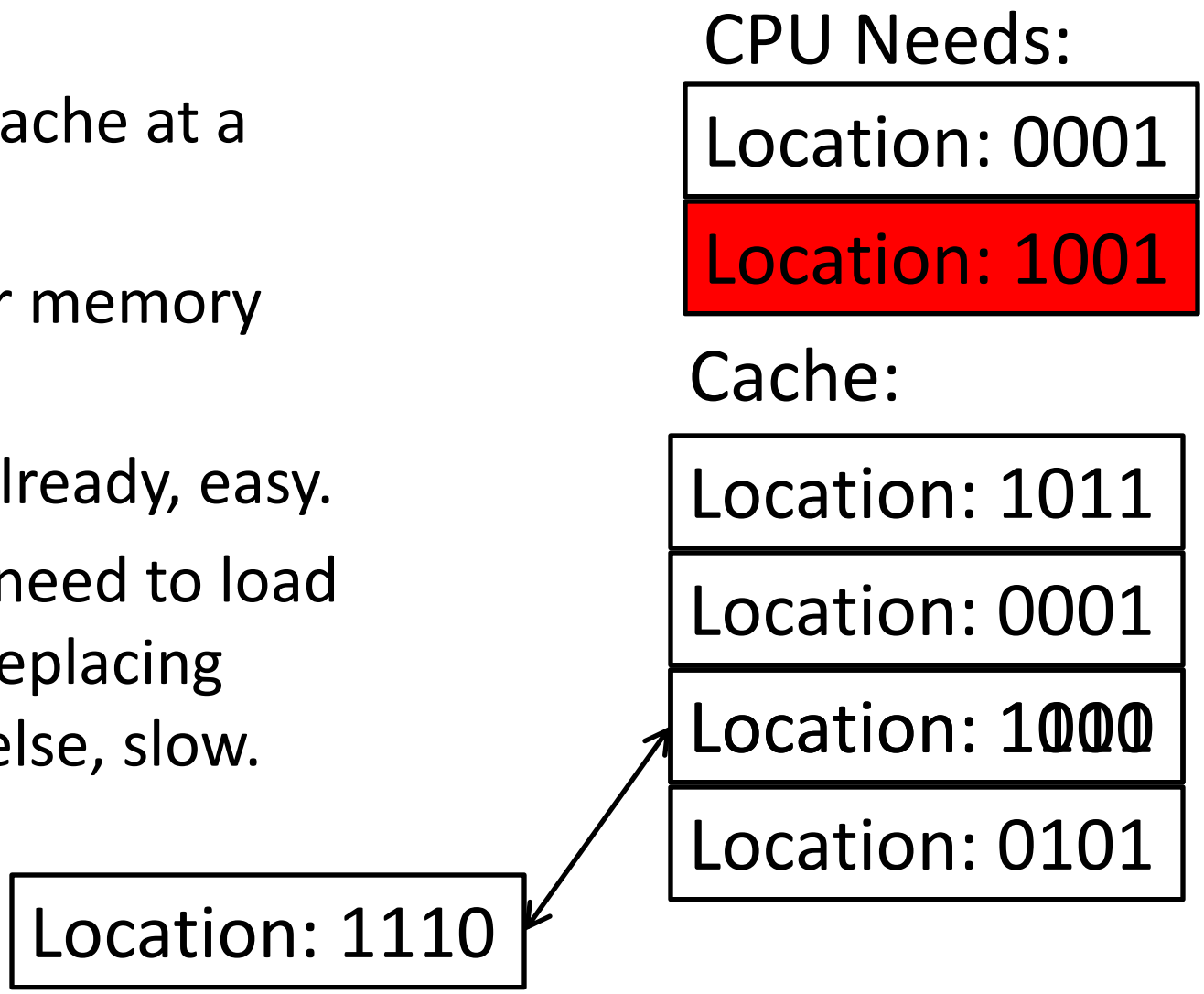
Optimal Caching (not in textbook)

- Communication between disk and CPU is slow.
- Have much smaller cache close to CPU.
- Store only a bit in cache at a time.
- If need to access some other location, will need to load into cache (slow).



Model

- k words in cache at a time.
- CPU asks for memory access.
- If in cache already, easy.
- Otherwise, need to load into cache replacing something else, slow.



Optimal Caching

Problem: Given sequence of memory accesses and cache size, find a cache schedule that involves fewest possible number of swaps with disk.

CPU	A	B	A	C	A	D	E	C	B	C	A	C
Cache 1												
Cache 2												

8 Cache misses.

Observation

- No need to get new entries in cache ahead of time.
- Only make replacements when new value is called for.
- Only question algorithm needs to answer is which memory cells to overwrite.

Question

What is a good candidate greedy procedure for deciding which cell to overwrite?

Furthest In The Future (FITF)

- For each cell consider the next time that memory location will be called for.
- Replace cell whose next call is the furthest in the future.

	X	A	Y	A	B	B	X	C
A								
B								
C								

Proof of Optimality

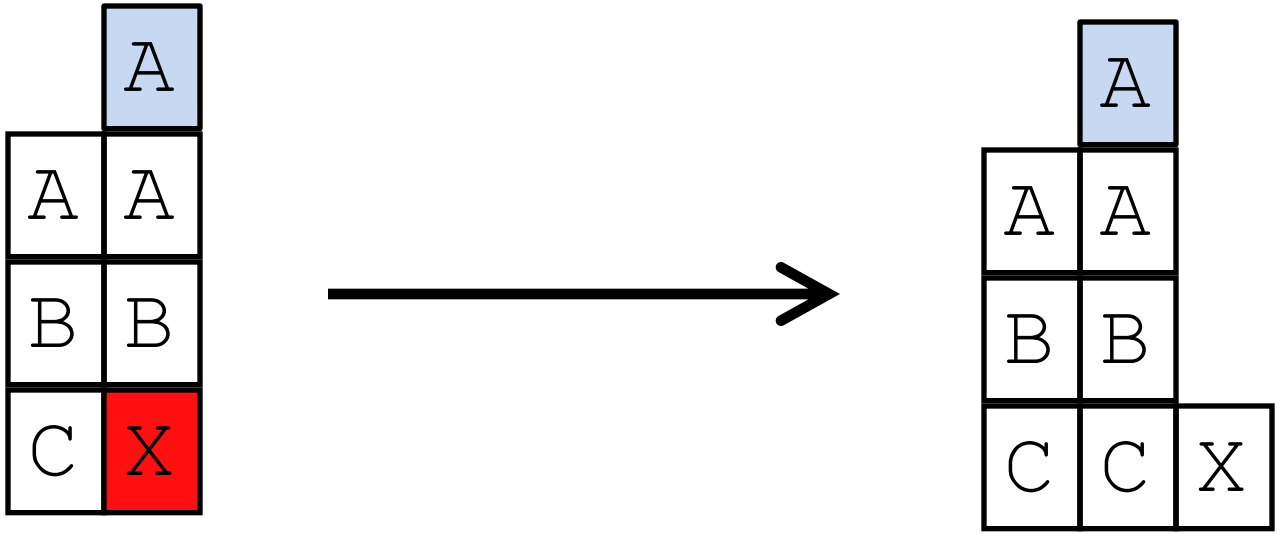
- Exchange argument
- n^{th} decision: What to do at n^{th} time step.
- Given schedule S that agrees with FITF for first n time steps, create schedule S' that agrees for n+1 and has no more cache misses.

Case 1: S agrees with FITF on step n+1

Nothing to do. S’ = S.

Case 2: S Makes Unnecessary Replacement

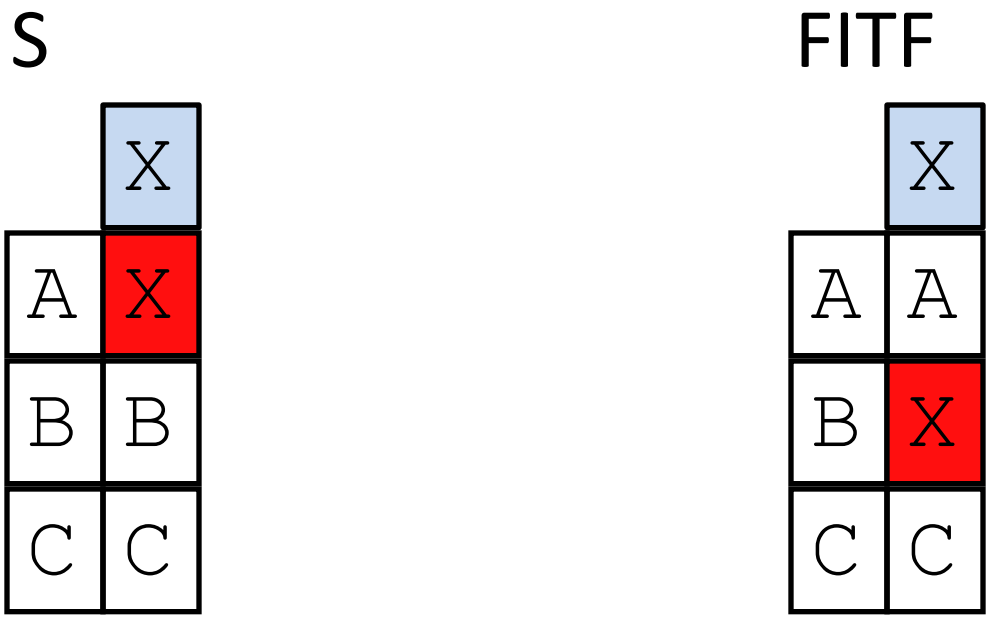
If S replaces some element of memory that was not immediately called for, put it off.



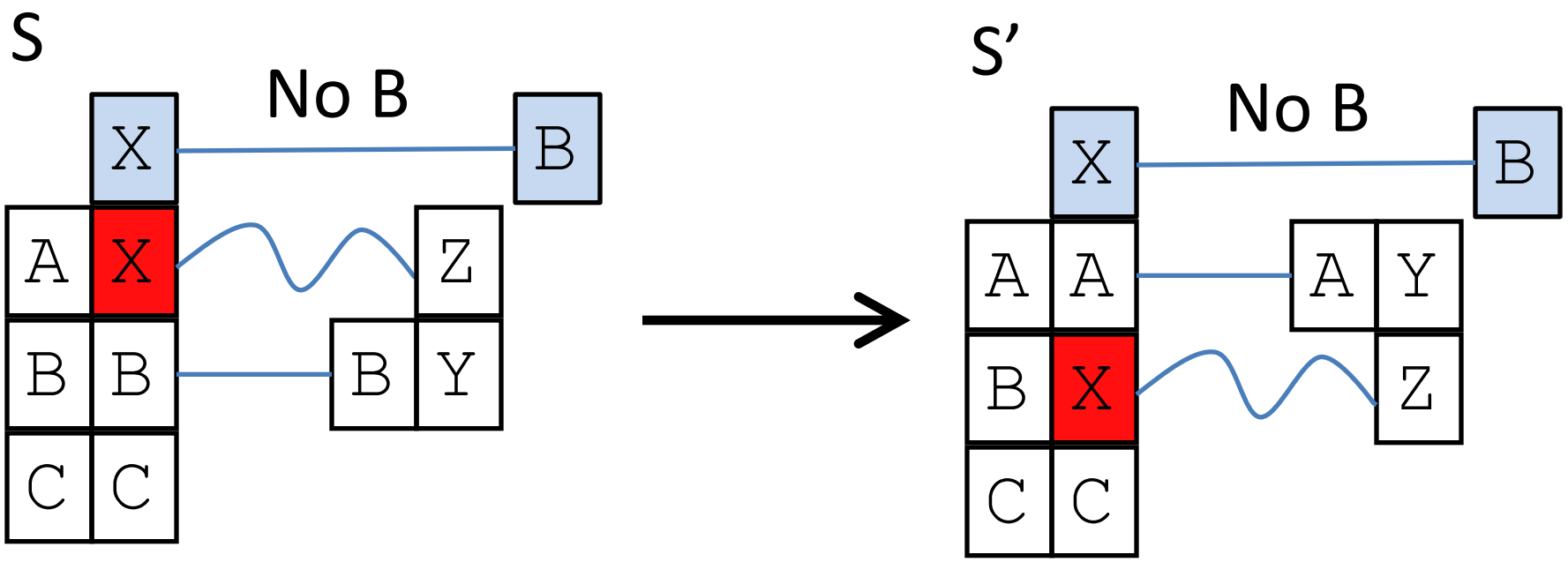
Can assume that S only replaces elements if there's a cache miss.

Case 3

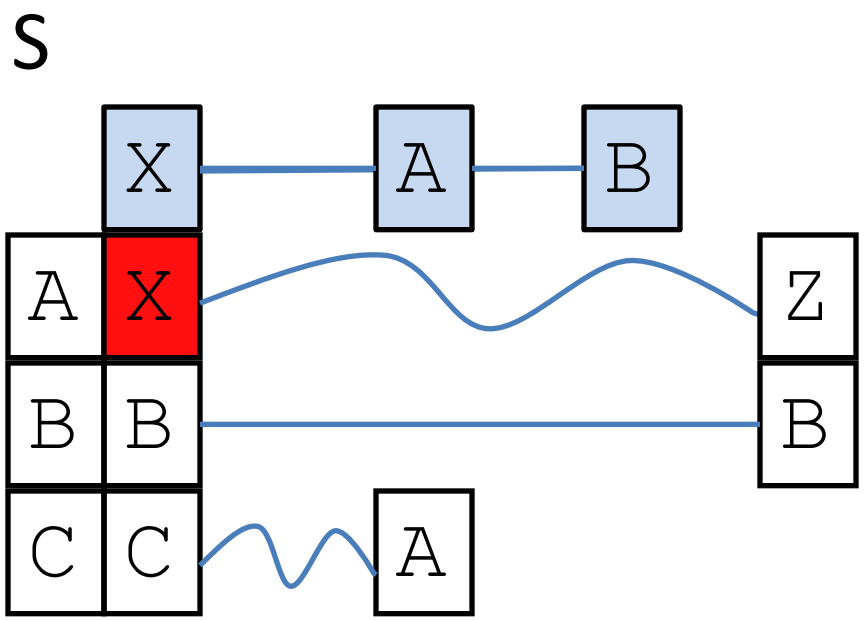
The remaining case is that there is a cache miss at step n+1 and that S replaces the *wrong* thing.



Case 3a: S throws out B before using it

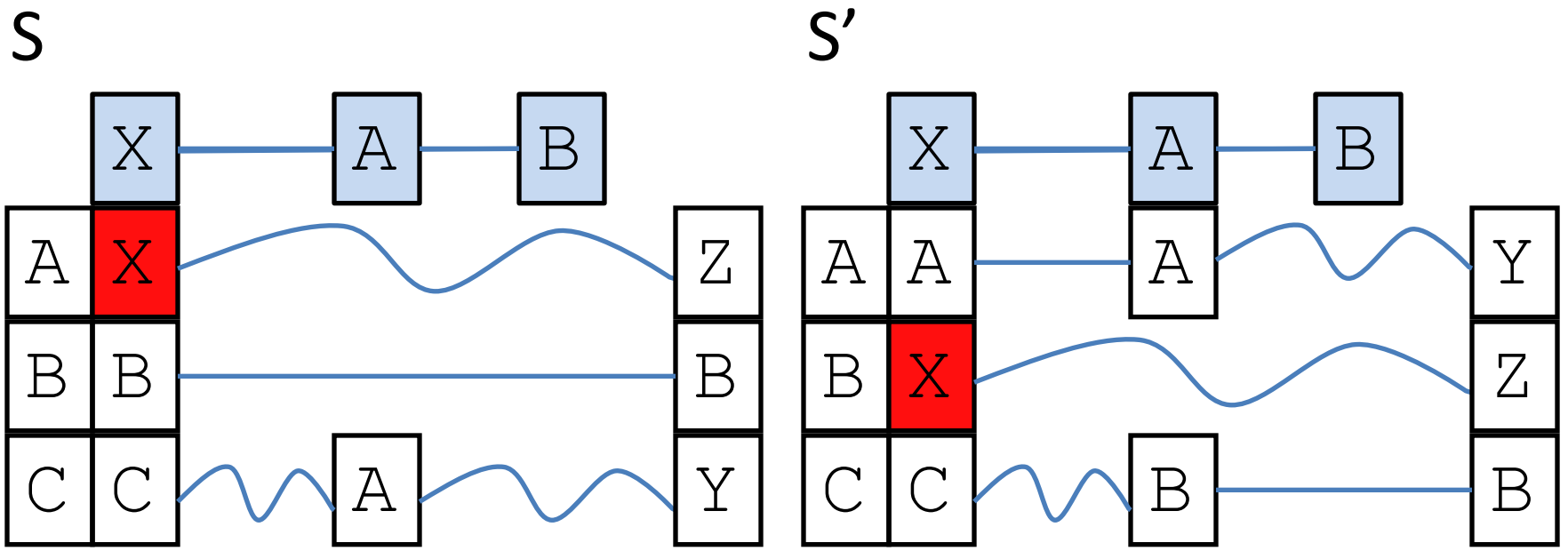


Case 3b: S keeps B until it is used



- B is FITF
- A is used sometime before B.
- A must be loaded into memory somewhere else.

Case 3b: S keeps B until it is used



Instead of replacing A and then bringing it back, we can replace B and then bring it back.

Least Recently Used

Unfortunately, FITF requires that you know exactly what future memory accesses are needed. This makes it hard to use in practice.

Instead, people often throw out the Least Recently Used (LRU) memory location. This is *not* always optimal, but it can be shown to be competitive with the optimal.