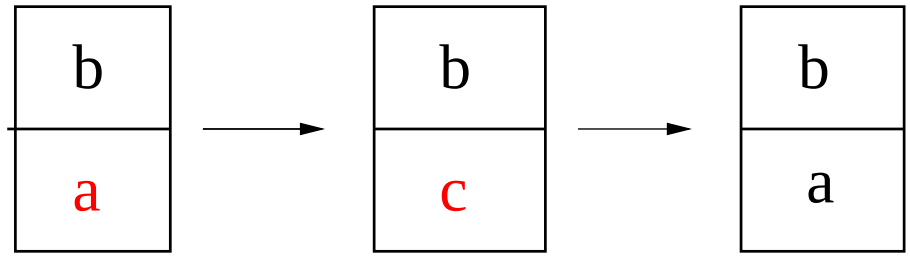
Paradigm: Greedy

Offline Caching

R. Inkulu
http://www.iitg.ac.in/rinkulu/

Description



$$I = \{a, b, c\}, k = 2$$

$$Q : a, b, c, b, c, a, b$$

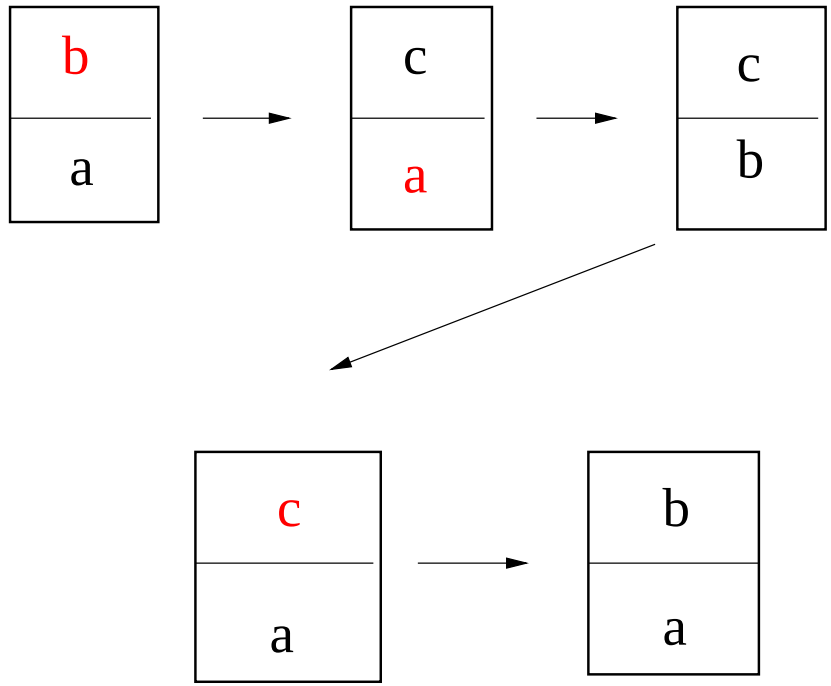
eviction schedule: a, c

number of cache misses $= 2$

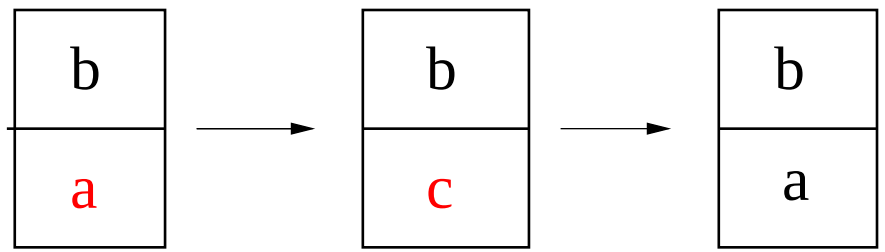
Two level memory hierarchy comprising of main memory and cache wherein the main memory contains a set I of integers and cache can hold up to k integers such that $k < |I|$. Devise an eviction schedule that causes minimum number of cache misses to satisfy an input ordered sequence Q.

> We do not consider loading an element from main memory to an empty entry in cache as a cache miss.

Greedy strategy that falters



evict least recently used: number of cache misses = 4

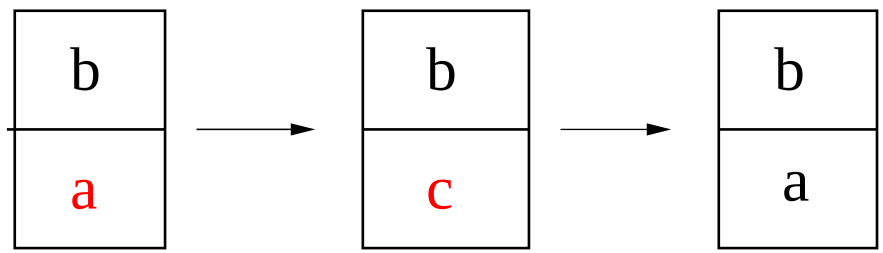


an optimal solution: number of cache misses = 2

$Q : b, a, c, b, c, a, b.$

Greedy strategy for further exploration

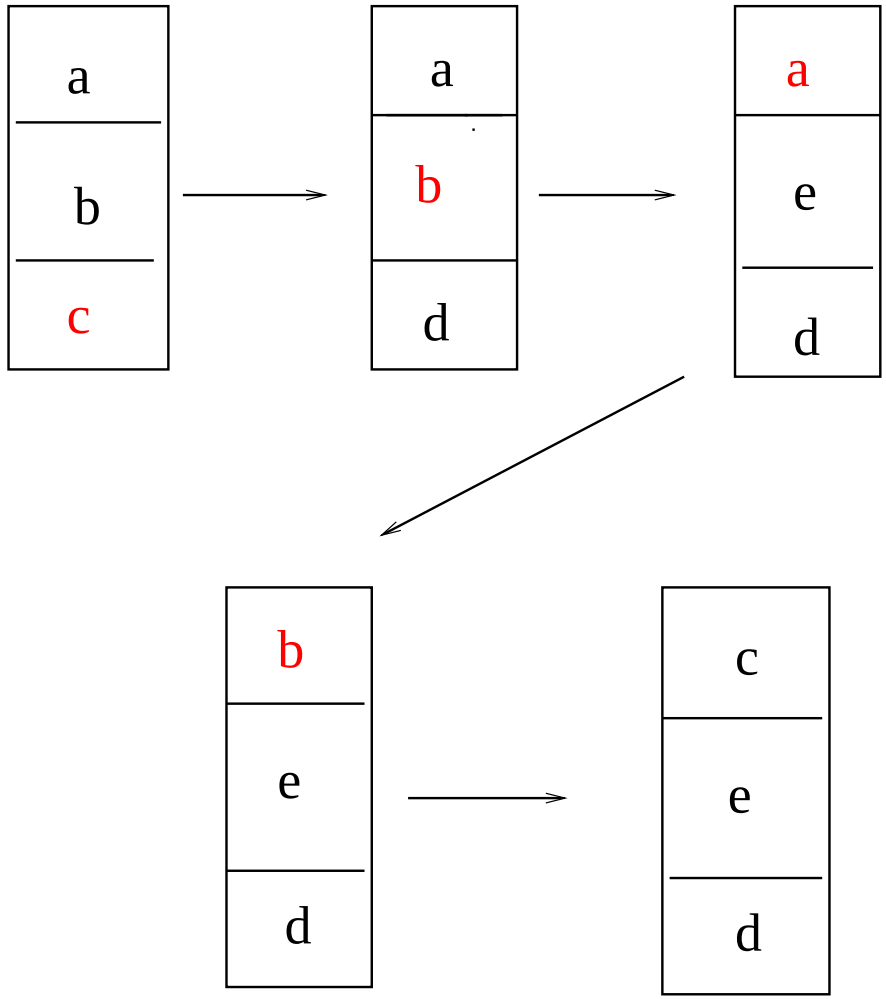
Evict an element which occurs late (farthest) in future in the remaining part of input sequence.



an optimal solution: number of cache misses = 2

$$Q : a, b, c, b, c, a, b.$$

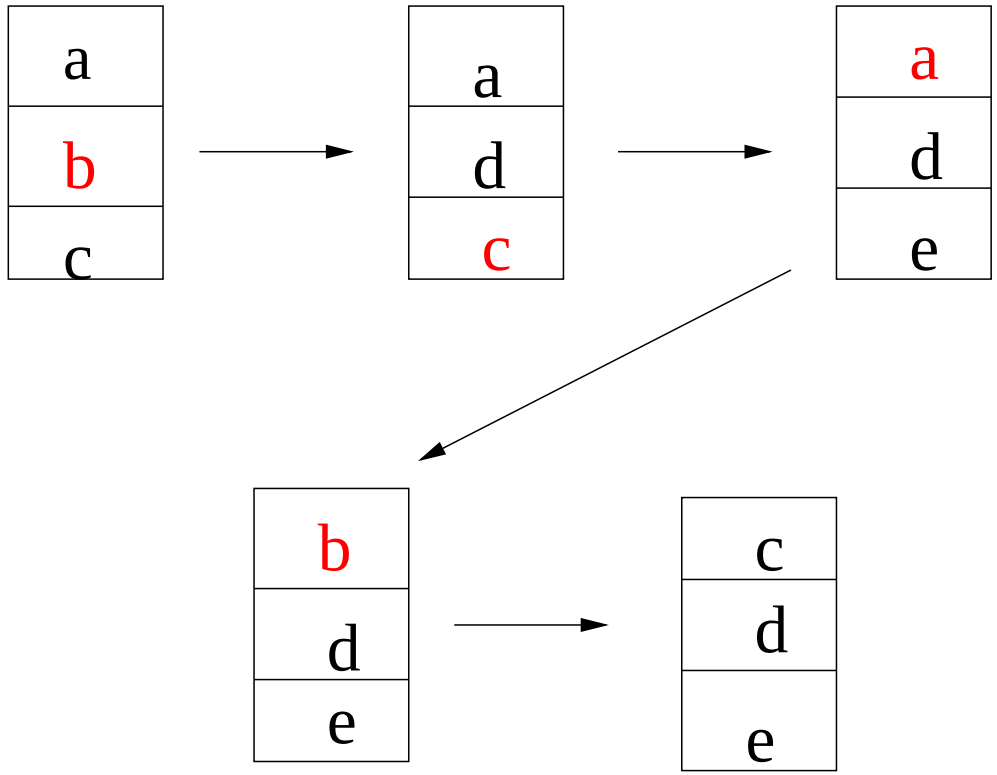
Multiple optimal solutions possible



processed with the above greedy algorithm

$$Q : a, b, c, d, a, d, e, a, d, b, c.$$

Multiple optimal solutions possible (cont)



processed with some other algorithm

$Q : a, b, c, d, a, d, e, a, d, b, c.$

Correctness

Greedy algorithm G does not incur more misses than any other optimal algorithm O.

- Let $a_1, \ldots, a_j, a_{j+1}, \ldots, a_n$ be the input sequence. If the eviction schedule S of O agrees with the eviction schedule S_G of G after processing a_j, then there exists an eviction schedule S' that agrees with S_G after processing a_{j+1} and incurs no more cache misses than S does.
- Applying this strategy inductively for at most n times, proves the optimality of S_G.

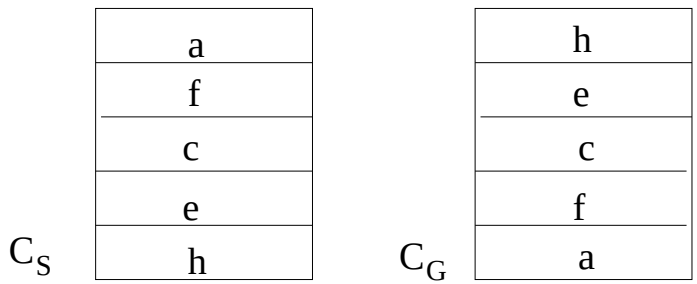
Correctness

Let C_S, C_G be the caches of same size associated with algorithms O and G respectively. Suppose after processing a_j in $a_1, \ldots, a_j, \ldots, a_n$:

$$S : b_1, b_2, \ldots, b_i$$

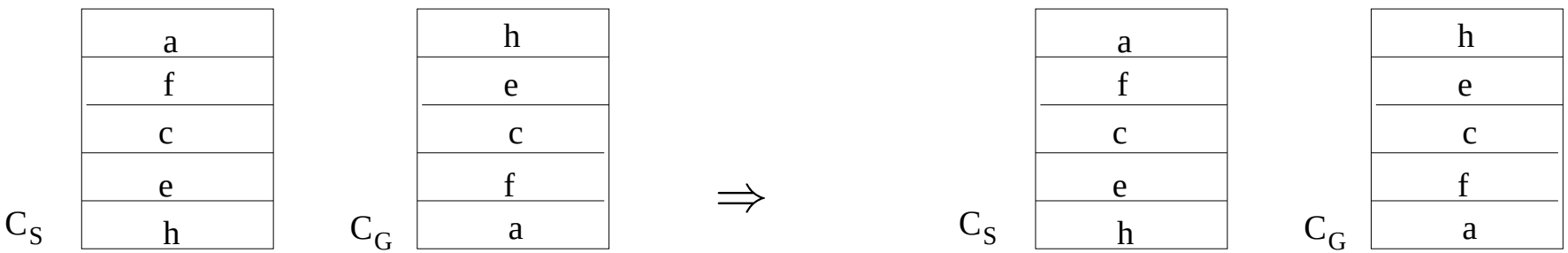
$$S_G : b_1, b_2, \ldots, b_i$$

Then the contents of C_S and C_G are same after processing a_j.



Correctness: Case (i)

$$Q : a_1, a_2, \ldots, a_j, a_{j+1} = c, \ldots, a_k, a_{k+1}, \ldots, a_n$$



after processing a_j:

$S : b_1, b_2, \ldots, b_i$

$S_G : b_1, b_2, \ldots, b_i$

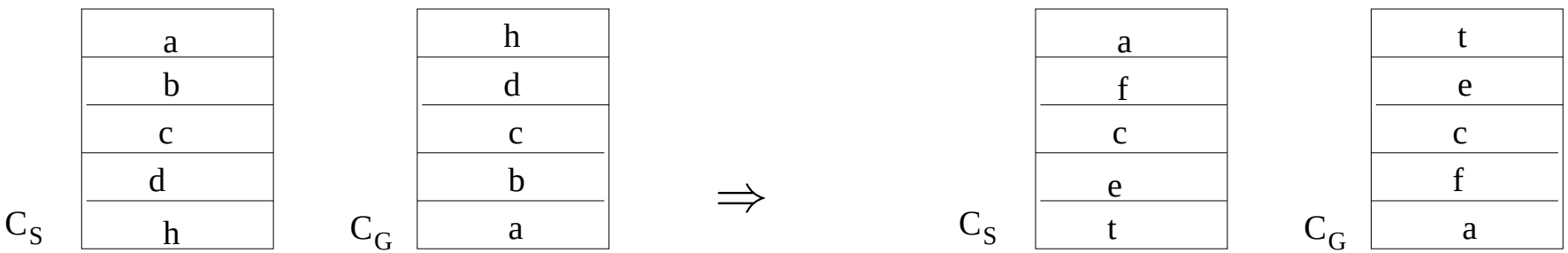
after processing a_{j+1}:

$S : b_1, b_2, \ldots, b_i$

$S_G : b_1, b_2, \ldots, b_i$

Correctness: Case (ii)

$$Q : a_1, a_2, \ldots, a_j, a_{j+1} = t, \ldots, a_k, a_{k+1}, \ldots, a_n$$



after processing a_j:

$S : b_1, b_2, \ldots, b_i$

$S_G : b_1, b_2, \ldots, b_i$

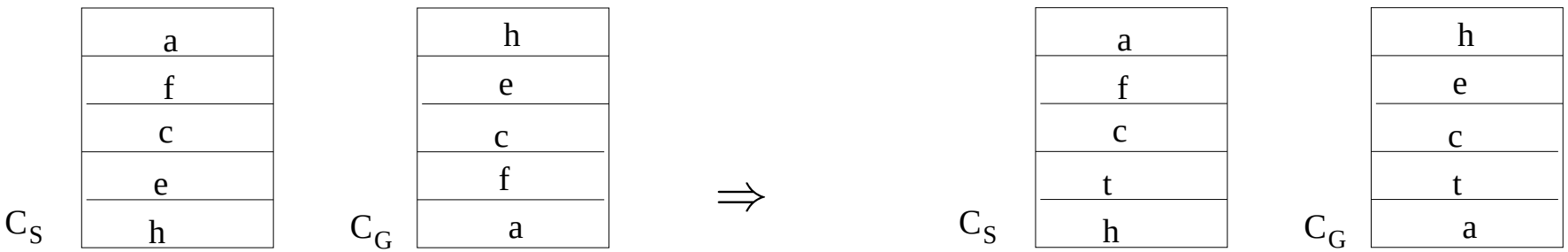
after processing a_{j+1}:

$S : b_1, b_2, \ldots, b_i, h$

$S_G : b_1, b_2, \ldots, b_i, h$

Correctness: Case (iii)

$Q : a_1, a_2, \ldots, a_j, a_{j+1} = t, \ldots, a_k, a_{k+1}, \ldots, a_n$



after processing a_j:

$S : b_1, b_2, \ldots, b_i$

$S_G : b_1, b_2, \ldots, b_i$

after processing a_{j+1}:

$S : b_1, b_2, \ldots, b_i, e$

$S_G : b_1, b_2, \ldots, b_i, f$

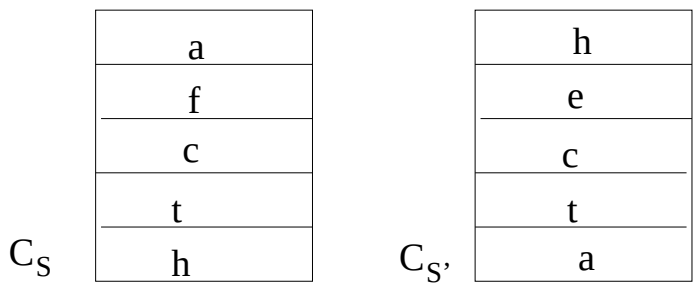
The element f is farthest to a_{j+1} in the remaining sequence, when compared to e.

Correctness via *exchange argument*: Handling Case (iii)

$Q : a_1, a_2, \ldots, a_j, a_{j+1}, \ldots, a_k, a_{k+1}, \ldots, a_n$

Construct S' which behaves same as G until a_{j+1} is processed; and, after procesing the whole of Q, $|S'| \leq |S|$.

The strategy would be to make the contents of $C_{S'}$ same as C_S as early as possible; till that state is achieved, we need to ensure S' does not incur more cache misses than S.



after processing a_{j+1}:

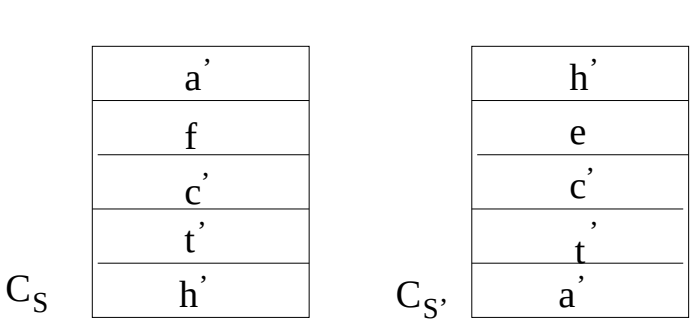
$S : b_1, b_2, \ldots, b_i, e$

$S' = S_G : b_1, b_2, \ldots, b_i, f$

Correctness: Subcase (iii)(a)

$Q : a_1, a_2, \ldots, a_j, a_{j+1}, \ldots, a_k, a_{k+1}, \ldots, a_n$

Cache misses due to elements in $a_{j+2}, \ldots, a_{k-1}$ causing eviction of elements other than f in S, and none of these elements equal to e; $a_k \neq e$ causing the eviction of f in C_S:

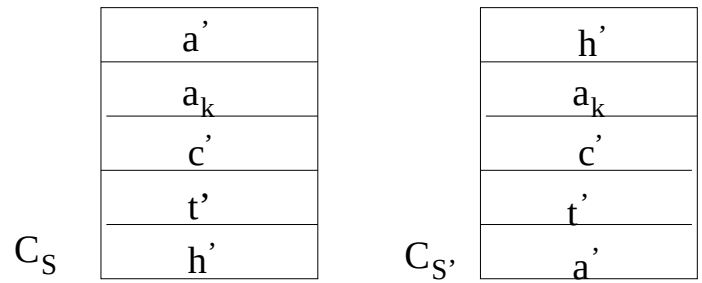


after processing a_{k-1}:

$S : b_1, b_2, \ldots, b_i, e, ..., $

$S' : b_1, b_2, \ldots, b_i, f, ..., $

$\Rightarrow$



after processing a_k:

$S : b_1, b_2, \ldots, b_i, e, ..., f$

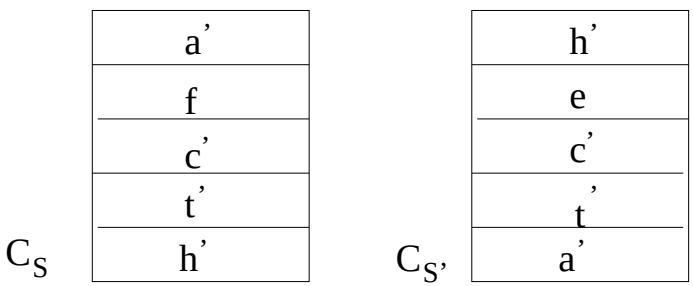
$S' : b_1, b_2, \ldots, b_i, f, ..., e$

$|S| = |S'|$

Correctness: Subcase (iii)(b)

$Q : a_1, a_2, \ldots, a_j, a_{j+1}, \ldots, a_k = e, a_{k+1}, \ldots, a_n$

Cache misses due to elements in $a_{j+2}, \ldots, a_{k-1}$ causing eviction of elements other than f in S, and none of these elements equal to e; $a_k = e$ causing the eviction of f in C_S:

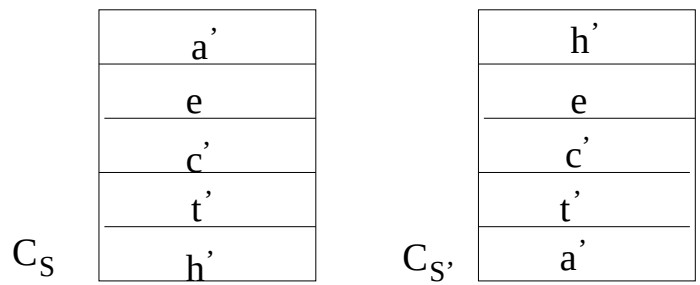


after processing a_{k-1}:

$S : b_1, b_2, \ldots, b_i, e, ...,$

$S' : b_1, b_2, \ldots, b_i, f, ...,$

$\Rightarrow$



after processing a_k:

$S : b_1, b_2, \ldots, b_i, e, ..., f$

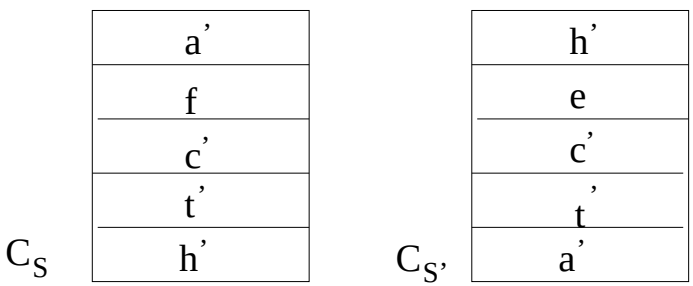
$S' : b_1, b_2, \ldots, b_i, f, ...$

$|S| - 1 = |S'|$

Correctness: Subcase (iii)(c)

$Q : a_1, a_2, \ldots, a_j, a_{j+1}, \ldots, a_k, a_{k+1}, \ldots, a_n$

Cache misses due to elements in $a_{j+2}, \ldots, a_{k-1}$ causing eviction of elements other than f in S, and none of these elements equal to e; and $a_k = e$ causing the eviction of c' in C_S:

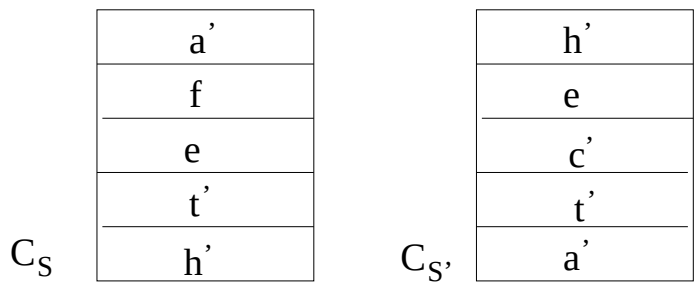


after processing a_{k-1}:

$S : b_1, b_2, \ldots, b_i, e, ...,$

$S' : b_1, b_2, \ldots, b_i, f, ...,$

$\Rightarrow$



after processing a_k:

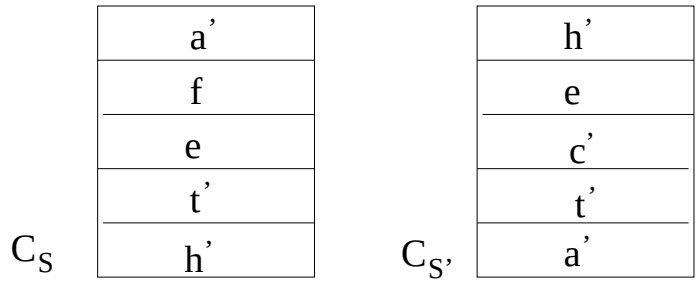
$S : b_1, b_2, \ldots, b_i, e, ..., c'$

$S' : b_1, b_2, \ldots, b_i, f, ...,$

$|S| - 1 = |S'|$

Correctness: Subcase (iii)(c) (cont)

$$Q : a_1, a_2, \ldots, a_j, a_{j+1}, \ldots, a_k, a_{k+1}, \ldots, a_l, a_{l+1}, \ldots, a_n$$



after processing a_k:

$$S : b_1, b_2, \ldots, b_i, e, ..., c'$$

$$S' : b_1, b_2, \ldots, b_i, f, ...,$$

It is guaranteed that $|S'| \leq |S|$:

For $a_l = f$, evict c' in S'.

Whenever f is evicted in S, evict c' in S'.

Does not matter if nothing of this sort happens as $|S| - 1 = |S'|$.

Correctness: Subcase (iii)(d)

$Q : a_1, a_2, \ldots, a_j, a_{j+1}, \ldots, a_k, a_{k+1}, \ldots, a_n$

Cache misses due to elements in $a_{j+2}, \ldots, a_{k-1}$ causing eviction of elements other than f in S, and none of these elements equal to e; and $a_k = f$.

This case never occur as e must occur before we encounter f.

In other words, by the time we encounter f, three subcases together ensure $C_S = C_{S'}$.

Analysis

Homework!