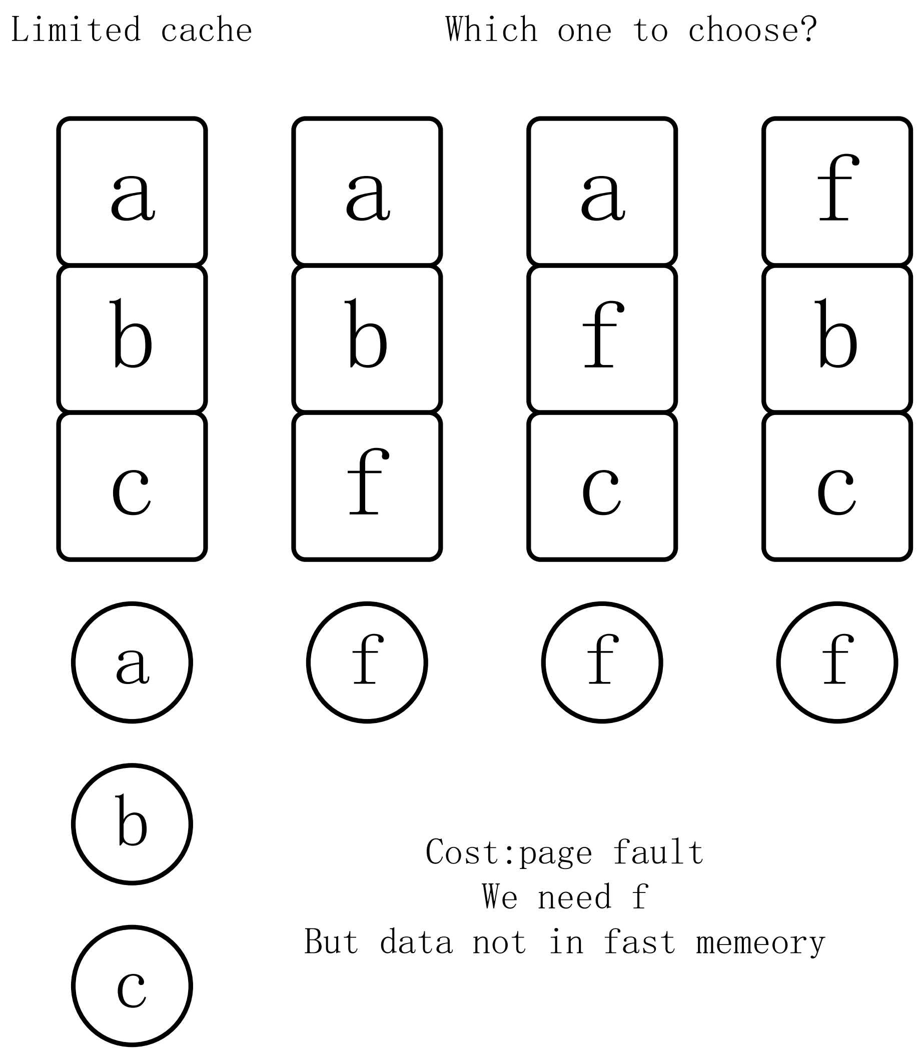
Lecture 9: Online Data-Page Replacement Algorithm

1. Operating System Memory Scheduling

- Two levels of memory in operating system
  - Small size fast memory (more expensive but small)
  - Large size slow memory (relatively cheaper but slow)
- Page fault: data accessed is not in fast memory
  - Upload the page containing the data needed.
- Objective: Design page replacement policy to minimize page fault.



2. Page Replacement Algorithm

- Offline/Clairvoyant: Knowing the page sequence to be accessed in advance.
  - FIF (the furthest in the future algorithm)
    - FIF is optimal offline.
    - When cache has a page fault, remove a page in the cache to let the new page in; Choose the removed page that will appear furthest in the future.
- Online/Non-Clairvoyant: Not knowing the page sequence to be accessed in advance.
  - Data are past data already given when making decision (or training)
  - Objective: Design page replacement policy so that the online page fault against offline page fault is minimized (in the worst/average case).
    - Given a page arrival sequence z, Cost (A, z) represents the # of page fault by algorithm A.
    - OPT (z) represents the minimum # of page faults by the best clairvoyant algorithm knowing the sequence z of page arrivals.
  - Competitive ratio $max_{all\ z}\{\frac{Cost(A,z)}{OPT(z)}\}$
  - Metrics: to minimize competitive ratio in the worst/average case
    - $min_A max_z\{\frac{Cost(A,z)}{OPT(z)}\}$
    - $\underset{z,|z|=n}{min_A} \frac{\sum_{|z|=n} Cost(A,z)}{\sum_{|z|=n} OPT(z)}$

3. Evaluation of offline & online Algorithms

- Lower Bound k for Any Deterministic Algorithm
  - Denote a deterministic online page replacement algorithm as A; Denote page sequences as z.
  - $\forall \mathrm{A}, \exists \mathrm{z}, \frac{Cost(A,z)}{OPT(z)} \geq k$
  - Proof
    - Let $\mathrm{Z} = \{p_1, p_2, \dots, p_{k+1}\}$ be a set of $\mathrm{k}+1$arbitrary pages. Assuming without loss of generality that A and OPT initially have $p_1, p_2, \dots, p_k$ in their fast memories.
    - Consider the following request sequence. No matter what A throw out of cache, it will be next input. Online algorithm A has a page fault on every request.
    - Suppose that OPT has a fault on some request $\sigma(\mathrm{t})$. When serving $\sigma(\mathrm{t})$, OPT remove a page is not requested during the next $\mathrm{k}-1$ request. Thus, on any k consecutive requests, OPT has at most one fault and A has k faults.
    - In summary, $\forall \mathrm{A}, \exists \mathrm{z}, \frac{Cost(A,z)}{OPT(z)} \geq k$
- FIF is Always Optimal
  - Proof (In lecture note of Fan Xin)
- LRU with competitive ratio k
  - On page fault when a new page is to be added, the pages to keep is the most recently used ones.

- The least recently used one is removed.

- Proof of competitive ratio k
  - Break input into subsequences $\sigma_1, \sigma_2, \dots, \sigma_b$ such that σ_i is the longest sequence where k pages appeared in the input.
  - LRU faults by ⩽ k times each block, total $k + (b-1)k$ times in the input sequence.
  - Any algorithm faults by at least once at each block transition, total at least $k + (b-1)$ faults.
  - Competitive ratio is $\frac{k+(b-1)k}{k+(b-1)}$ ⩽ k.