COT 6936: Topics in Algorithms

Giri Narasimhan

ECS 254A / EC 2443; Phone: x3748

giri@cs.fiu.edu

http://www.cs.fiu.edu/~giri/teach/COT6936_S10.html

https://online.cis.fiu.edu/portal/course/view.php?id=427

Online Problems

- Should I buy a car/skis/camping gear or rent them when needed?
- Should I buy Google stocks today or sell them or hold on to them?
- Should I work on my homework in Algorithms or my homework in OS or on my research?
- Decisions have to be made based on past and current request/task

Paging Problem

- Given 2-level storage system
  - Limited Faster Memory (k pages) "CACHE"
  - Unlimited Slower Memory
- Input: Sequence of page requests (Infinite, Online)
- Assumption: "Lazy" response (Demand Paging)
  - If page is in CACHE, no changes to contents
  - If page is not in CACHE, make place for it in CACHE by replacing an existing page
- Need: A "page replacement" algorithm

Well-known Page Replacement Algorithms

- LRU: evict page whose most recent access was earliest among all pages
- FIFO: evict page brought in earliest
- LIFO: evict page brought in most recently
- LFU: evict page least frequently used

Comparing online algorithms?

Game between Cruel Adversary and your algorithm

- Analyze: time? performance?
  - Input length?
  - Performance depends on request sequence
    - Probabilistic models? Markov Decision process
- Competitive analysis [Sleator and Tarjan]
  - Compare with optimal offline algorithm (OPT)
    - OPT is clairvoyant; no prob assumptions; "worst-case"
- Algorithm A is α-competitive if there exists constants b such that for every σ:
  - $cost_A(\sigma) \leq \alpha\, cost_{OPT}(\sigma) + b$

Optimal Algorithm for Paging

- MIN (Longest Forward Distance): Evict the page whose next access is latest.
- Cost: # of page faults
- Competitive Analysis: Compare
  - # of page faults of algorithm A with
  - # of page faults of algorithm MIN
- We want to compute the competitiveness of LRU, LIFO, FIFO, LFU, etc.

Lower Bound for any algorithm

- Cannot achieve better than k-competitive!
  - No algorithm is α-competitive for $\alpha < k$
    - Fix algorithm A,
    - Construct a request sequence σ, and
    - Show that: $cost_A(\sigma) \geq k\ cost_{OPT}(\sigma)$
- Sequence σ will only have k+1 possible pages
  - make 1..k+1 the first k+1 requests
  - make next request as the page evicted by algorithm A (Adversary Model)
    - A will fault on every request
    - OPT? Will fault every k requests

Upper Bound: LRU is k-Competitive

- Observation: if any subseq has k+1 distinct pages, MIN (any alg) faults at least once
  - Let p be 1st request; p is in CACHE (LRU & MIN)
  - Let T be any subsequence of σ with exactly k faults for LRU & with p accessed just before T.
  - LRU cannot fault on same page twice within T
    - Otherwise it will have faults on k+1 different pages
  - LRU cannot fault on p within T
    - Otherwise it will have faults on k+1 different pages
  - Thus, p followed by T requests k+1 distinct pages and MIN must fault at least once in T

LRU is k-competitive

- Partition σ into subsequences as follows:
  - Let s_0 include the first request, p, and the first k faults for LRU
  - Let s_i include subsequence after s_{i-1} with the next k faults for LRU
  - Argument applies for $T = s_i$, for every $i > 0$
  - If both algorithms start with empty CACHE or identical CACHE, then it applies to $i = 0$ also
  - Otherwise, LRU incurs k extra faults
- Thus, $cost_A(\sigma) \leq k\ cost_{OPT}(\sigma) + k$

Other Page Replacement Algorithms

- FIFO is k-competitive (Homework!)
- MFU and LIFO?