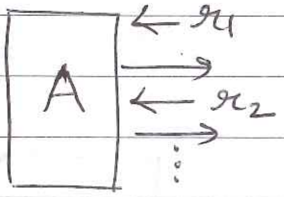
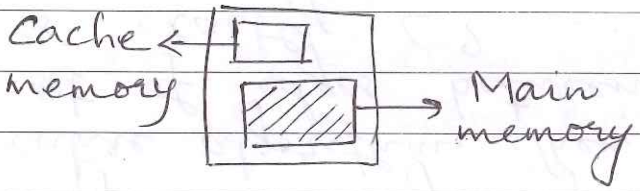
Lecture 20 Online Paging

All the algorithms that we have seen so far receive their entire input right at the beginning. In online algorithms - this is not the case.
- here we receive and process the input in partial amounts.

An online algorithm receives a sequence of requests for service. It must service each request before it receives the next one.
So the algorithm can be visualized as:



Example. Paging in a 2-level memory storage system



The cache memory can hold only a limited number of items.

Here each request is for a memory item.
If this item is in the cache, then we incur no cost in processing this request.
If this item is not in the cache, then we need to fetch it from the main memory at unit cost. In addition, one of the items in the cache is evicted, i.e. deleted or removed, to make room for this item. We want to design a paging algorithm that minimizes the number of misses.
Miss : requested item is not in the cache
Hit : requested item is in the cache

So the problem is: without knowing future requests the algorithm has to decide which item in the cache should be evicted to make room for the newly requested item.

What eviction rule should be used?

- first-in-first-out : throw out the page or item that entered the cache earliest
- least-frequently-used : estimate the number of times each page has been requested in some time interval and throw out the page with the least frequency
- other rules are least-recently-used & so on.

Let us show that for any online paging algorithm A, ∃ a sequence of arbitrary length such that A misses on every request.

Let $f_A(r_1, \ldots, r_N)$ = number of misses by A on $r_1, r_2, \ldots, r_N$.

Show a sequence $(r_1, \ldots, r_N)$ such that $f_A(r_1, \ldots, r_N) = N$.

Let r_1 = item not in the cache
r_2 = item evicted from the cache to service r_1
⋮

→ here we are assuming that the total number of items = k+1, where cache size = k.

What about offline algorithms?

- An offline algorithm knows the entire request sequence in advance and can tailor its response accordingly.

Consider the following eviction rule for an offline algorithm: evict that item whose request comes furthest in the future. Call this algo. MIN.

Let $f_{OPT}(r_1, \ldots, r_N)$ = number of misses suffered by the optimal offline algorithm on $(r_1, \ldots, r_N)$.

We will show that for any request sequence $(r_1, \dots, r_N)$, we have $f_{OPT}(r_1, \dots, r_N) \leq \lceil N/k \rceil$

* Let us show that the algorithm MIN suffers $\leq \lceil N/k \rceil$ misses on any request sequence $(r_1, \dots, r_N)$.

We will partition the request sequence into rounds such that a round consists of k different items being requested. The algo. MIN suffers a miss on the first request of every round, pays unit cost for it and evicts the item not requested in this round. So no more misses in this round.

Hence the total number of misses $\leq \lceil N/k \rceil$.

How do we measure the performance of an online algorithm A?

→ an online paging algorithm A is c-competitive if for every sequence of requests $(r_1, \dots, r_N)$ we have $f_A(r_1, \dots, r_N) \leq c \cdot f_{OPT}(r_1, \dots, r_N) + b$, where b is a constant independent of N (b may depend on k)

The least c for which A is c-competitive is called the competitive ratio of A.
The competitive ratio of any deterministic online paging algorithm $\geq k$ since $N \leq c \lceil \frac{N}{k} \rceil \Rightarrow c \geq k$.

Can we overcome the above negative result using randomization? Now A is a randomized algorithm.

- On a miss, A makes a random choice of what item to evict. So $f_A(r_1, \dots, r_N)$ is a random variable and on different days, $f_A(r_1, \dots, r_N)$ takes different values. So $E[f_A(r_1, \dots, r_N)]$ is our handle on this.

In proving the negative result on the competitive ratio of deterministic paging algorithms, we used an adversary to generate the sequence of requests $(r_1, \ldots, r_N)$ such that $f_A(r_1, \ldots, r_N) = N$.

- In the randomized world, what is the power that we would give an adversary?

Oblivious Adversary: This adversary has no advance knowledge of the random choices made by A.

We may assume that the request sequence $(r_1, \ldots, r_N)$ is fixed by the adversary in advance and the execution of the algorithm does not alter this.

Algorithm A is c-competitive if $\exists$ a value b (independent of N) such that

$$E[f_A(r_1, \ldots, r_N)] \leq c \cdot f_{OPT}(r_1, \ldots, r_N) + b$$

for every $(r_1, \ldots, r_N)$.

Let us now consider a randomized algorithm for the paging problem. This algorithm is called the Marker algorithm.

The Marker algorithm proceeds in rounds.

- At the beginning of each round, every page in the cache is unmarked.

- Repeat

    * if the requested page is in the cache then mark the page if it is unmarked.

```
        else 1) fetch the page from the main memory
             2) evict an unmarked page chosen
                uniformly at random
             3) mark the incoming page
    until there is a miss (i.e, requested page is not
      in the cache) and all the pages in the
      cache are marked.
- Unmark all the pages in the cache and
  start the next round.
```

To analyze the competitive ratio of the Marker algorithm, we need to bound

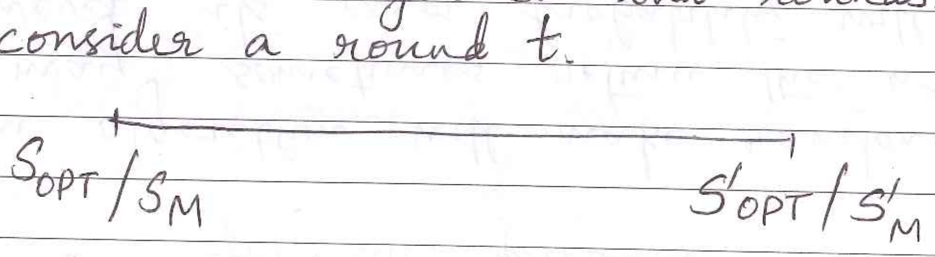
$$\max_{(r_1,\dots,r_N)} \frac{E[f_{Marker}(r_1,\dots,r_N)]}{f_{OPT}(r_1,\dots,r_N)}.$$

So what we need to do now are the following:

(i) lower bound $f_{OPT}(r_1,\dots,r_N)$

(ii) upper bound $E[f_{Marker}(r_1,\dots,r_N)]$.

Job (i). Let OPT be the optimal offline algorithm. Recall that we partitioned the execution of the Marker algorithm into rounds. Let us consider a round t.



Here S_{OPT} is the set of pages in OPT's cache at the start of round t and S'_{OPT} is the set of pages in OPT's cache at the end of round t. Similarly for S_M and S'_M: these are wrt Marker algorithm.

What is S'_M? This is the set of pages or items in Marker algorithm's cache at the end of round t.

- We know from Marker algorithm that this is exactly the set of pages requested in round t.

We want to lower bound the number of misses suffered by OPT in round t. Let x denote this number.

Let l_t be the number of items requested in this round that were not requested in the previous round. Let $d_1 = |S_{OPT} - S_M|$ and

$$d_2 = |S'_{OPT} - S'_M|.$$

Observe that $x \geq l_t - d_1$. This is because none of the above l_t items was in S_M. At most d_1 of them can be in S_{OPT}

Similarly, observe that $x \geq d_2$. This is because d_2 of the items requested in this round have already been replaced - these replacements occurred due to misses by OPT.

Hence $x \geq \dfrac{l_t + d_2 - d_1}{2}$

Adding the above inequalities for all the rounds, the total number of misses suffered by OPT

$$\geq \sum_t \frac{l_t}{2} + \underbrace{\frac{d_{final} - d_{init}}{2}}_{\text{note that this is a number between } -k/2 \text{ and } k/2}$$

We will see how to upper bound the expected number of misses by Marker in the next lecture.