The Paging Problem

There are k cache locations, each of which can hold one page. If a page in the cache is requested, that is called a *hit*. If a page not in the cache is requested, that is called a *fault*. In case of a fault, a page already in the cache must be ejected, and the requested page moved in. This costs 1. A hit costs 0. To smooth the analysis, we assume that initially the cache is full.

One optimal paging strategy is called *longest forward distance* (LFD). In case of a fault, eject a page which will never be used again, if there is one. Otherwise, eject the page whose future use is farthest away.

Phases

A *phase* is defined to be maximal substring of the request sequence which begins with a fault of the adversary and contains no other fault of the adversary. A phase can have any length, but can contain at most k distinct pages.

Prove that LFD is optimal.

The Cruel Adversary

The *cruel adversary* initially requests an arbitrary page not in the cache. Subsequently, the cruel adversary always requests the page that the algorithm just ejected. Prove that no deterministic online algorithm for the paging problem can be less than k-competitive.

LRU

The most popular online algorithm is called *least recently used* (LRU). In case of a fault, eject the page whose past use was least recent. LRU is deterministic, trackless, and requires $O(k)$ memory. Prove that LRU is k-competitive.

MRU

The least intelligent online algorithm is called *most recently used* (MRU). In case of a fault, eject the page whose past use was most recent. MRU is deterministic, trackless, and requires $O(k)$ memory. Prove that MRU is not competitive.

MBL

The algorithm MBL is defined for $k = 2$. MBL is a mixture of LRU and MRU, with a bookmark. MBL has three states, which we call M, B, and L. In state B, MBL has one bookmarked page which is not in the cache. In state M or state L, there is no bookmarked page.

In any state, if there is a request to the more recently used page, the algorithm does nothing. In any state, if there is a request to the less recently used page in the cache, the algorithm moves to state M, and the bookmark, if any, is erased.

The complicated part of MBL is when there is a request to a page which is not in the cache.

1. If MBL is in state M and there is a request to a new page, the **more recently used** page is ejected and bookmarked, and the state changes to B.

2. If MBL is in state B and there is a request to the bookmarked page, the **less recently used** page is ejected and not bookmarked. The state changes to M.

3. If MBL is in state B and there is a request to a new page, the bookmark is erased, the less recently used page is ejected and not bookmarked, and the state changes to L.

4. If MBL is in state L and there is a request to a new page, the less recently used page is ejected and not bookmarked, and the state changes to M.

Prove that MBL is 2-competitive.

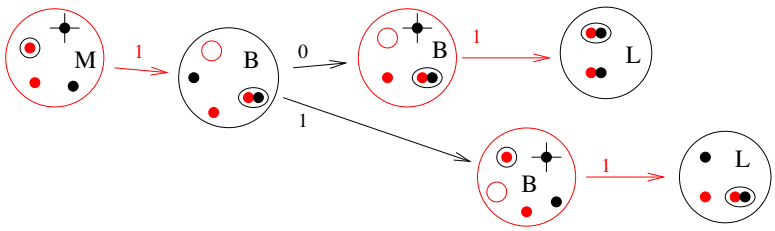


Figure 1: A fragment of the red-black graph for MBL

The Oblivious Forgetful Adversary

The *oblivious forgetful adversary* has no memory of past requests. The adversary only requests from a set of $k+1$ pages. At each step, this adversary requests one of those pages at random.

Prove that every randomized paging algorithm is exactly H_k-competitive against the oblivious forgetful adversary.

The Forgetful Random Paging Algorithm

The *forgetful random paging algorithm* (RAND) ejects a page uniformly at random whenever there is a fault. (See §4.2 of your textbook.) What is the worst-case competitiveness of RAND against an oblivious adversary?

Marking Algorithms

The randomized algorithm MARK is trackless, but not memoryless. At any given time, one or more pages in the cache are marked. Initially, all pages in the cache are marked.

If there is a hit, the requested page is marked. (If it is already marked, nothing happens.)

If there is a fault and not all pages in the cache are marked, one unmarked cache page, chosen uniformly at random, is ejected. The requested page is brought into the page and marked.

If there is a fault and all pages in the cache are marked, all marks are erased. Then, one cache page, chosen uniformly at random, is ejected. The requested page is brought into the page and marked.

Prove that MARK is $2H_k$-competitive.

Prove that, for $k = 2$, MARK is 2-competitive.[1]

The Barely Random Algorithm

Let $k = 2$. The *barely random algorithm* flips a coin, just once. With probability one half, it behaves as LRU henceforth. With probability one half, it behaves as MBL henceforth.

What is the competitiveness of the barely random algorithm against the oblivious adversary?

[1] That does not fit the formula, since $2H_2 = 3$.

The CKN Algorithm

The CKN algorithm is a randomized algorithm for the paging problem for $k = 2$. Each page in the cache is either *unmarked*, *marked*, or *double marked*. The more recently used page is always unmarked.

Roughly, the more marks a page has, the more likely it is to be ejected. The algorithm is defined as follows. Let r be the requested page.

1. If r is in the cache and unmarked, do nothing.
2. If r is in the cache and is marked or double marked, unmark r.
3. If r is new and both of the pages in cache are unmarked, eject one of those pages chosen uniformly at random, mark the other, and bring r into the cache unmarked.
4. If r is new and one of the pages in cache is marked:
   (a) With probability $\frac{5-\sqrt{13}}{2}$, eject the marked page and bring r into the cache unmarked. The previously unmarked page in cache remains unmarked.
   (b) With probability $\frac{\sqrt{13}-3}{2}$, eject the unmarked page, double mark the marked page, and bring r into the cache unmarked.
5. If r is new and one of the pages in cache is double marked, eject the double marked page, and bring r into the cache unmarked. The previously unmarked page in cache remains unmarked.

Is the CKN algorithm trackless? Find the competitiveness of the CKN algorithm. (Hint: it's irrational.)

Optimal Randomized Algorithms

Find a $\frac{3}{2}$-competitive randomized paging algorithm for the case that $k = 2$. Is your algorithm trackless?

Find an $\frac{11}{6}$-competitive randomized paging algorithm for the case that $k = 3$. Is your algorithm trackless? This problem is hard.