# More on the $k$-Paging Problem

## MRU

The most perversel wrong online algorithm for the $k$-paging problem 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

For the paging problem, a *bookmark* is the memory of the name of some page that is not in the cache, but was once ejected. In case of fault, an algorithm can compare the name of the requested paging with its bookmarks, and determine whether it is one of the bookmarked pages, as well as any other information about that page that the algorithm has chosen to remember.

If $\mathcal{A}$ is an online paging algorithm, we say that $\mathcal{A}$ has $b$ bookmarks if $\mathcal{A}$ can remember the names of up to $b$ different ejected pages. We can then define a *trackless* algorithm to be a 0-bookmark algorithm.

The algorithm MBL is defined for $k = 2$. MBL a 1-bookmark online algorithm, which is a mixture of LRU and MRU. 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, MBL 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 (not bookmarked) 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.

## The Barely Random Algorithm

Let $k = 2$. Let $\frac{1}{2}LRU + \frac{1}{2}MBL$ be the barely random algorithm which flips a coin, just once. With probability one half, it behaves as LRU henceforth. With probability one half, it behaves as MBL henceforth.

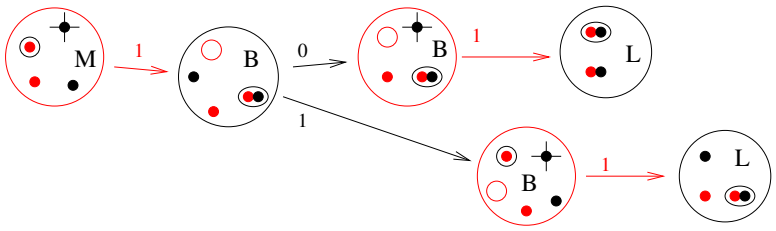



Figure 1: A fragment of the red-black graph for MBL

What is the competitiveness of $\frac{1}{2}LRU + \frac{1}{2}MBL$ against the oblivious adversary?

## The CKN Algorithm

The Chrobak-Koutsoupias-Noga (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.)