In-Class Worksheet

CS 181 Advanced Algorithms — Spring 2026

In **online caching / paging**: we have a cache of size k and N pages. At each time step, a page is requested. If it is already in cache, we get a *hit*. Otherwise we get a *miss* and must load the page into the cache, evicting some page if the cache is full. The goal is to minimize the total number of misses over the entire request sequence.

The **LRU** (Least Recently Used) algorithm always evicts the page whose last access was furthest in the past.

1. How good is LRU? Fix a cache size $k = 4$ and a page universe $\{1, 2, 3, 4, 5\}$. Construct a request sequence on which LRU does poorly compared to what we could have done in hindsight. In other words, make it so that

$$\frac{\text{cost(LRU)}}{\text{cost(OPT)}}$$

   is as large as possible.

2. We will show that LRU is k-competitive.

   Consider any request sequence. We will partition the request sequence into consecutive *phases* as follows. A **phase** is a maximal subsequence containing at most k distinct pages. The first phase begins with the initial page request.

   Example: Consider the following request sequence. In this example $k = 4$.

$$\underbrace{1,\ 2,\ 3,\ 1,\ 8,\ 2,\ 3}_{\text{phase 1}},\ \underbrace{5,\ 1,\ 2,\ 6}_{\text{phase 2}},\ \underbrace{3,\ 4,\ 2,\ 5}_{\text{phase 3}},\ 7\ldots$$

3. Partition the following request sequence into phases, where the cache has size 3.

$$1,\ 9,\ 1,\ 3,\ 9,\ 1,\ 3,\ 9,\ 1,\ 3,\ 9,\ 4,\ 5,\ 9,\ 4,\ 5,\ 4,\ 9,\ 5,\ 4\ldots$$

4. What is an upper bound on the number of faults that the LRU algorithm can incur in a single phase?

5. Suppose that a request sequence contains m phases. How many faults must be incurred (even by the clairvoyant optimum)?
