| infOnAlg-01a: Online Algorithms | Lecture 03 | Date: 07.11.2018 |
|---|---|---|
| Instructor: Sebastian Berndt | | Scribe: Monty Santarossa |

# 1 Caching

Caching describes a problem, where the data of $n \in \mathbb{N}$ different pages in stored in the big but slow memory and can be loaded into a faster but much smaller cache of size $k \in \mathbb{N}$. If a page is requested that is currently not in the cache, this is called a page fault and leads to the requested page being loaded into the cache. If the cache is full during a page fault, one or multiple pages in the cache have to be discarded. The aim is to discard the pages in the cache in a way that minimizes the overall number of page faults.

**Problem 1 (Caching)**

***Given:*** *Pages* $1, \dots, n$ *in memory, cache of size* $k \in \mathbb{N}$ *with* $k < n$*, Request Sequence* $\vec{P} = p_1, p_2, \dots, p_m$ *with* $p_i \in \{1, \dots, n\}$.

***Find:*** *Minimal costs (e.g. number of pages loaded into the cache)* OPT *for the request sequence.*

We assume the cost for loading a page from the memory into the cache to be 1.

# 2 Marking Algorithms

For any given request sequence $\vec{P} = p_1, p_2, \dots, p_m$ we can define a partition of $\vec{P}$ into different layers $l_0, l_1, \dots, l_k$ with $k \leq m$ such that:

- $l_0 = 0$
- $l_i$ is the longest connected subsequence $p_{l_{i-1}+1}, \dots, p_{l_i}$ where $|\{p_{l_{i-1}+1}, \dots, p_{l_i}\}| = k$, e.g. $|\{p_{l_{i-1}+1}, \dots, p_{l_i+1}\}| = k+1$.

We define the intervall $[l_i + 1, l_{i+1}]$ to be *phase* $i+1$.

## Example

$k = 3$

Request Sequence $\vec{P} =$ $\overbrace{1,2,4,2,1,}^{\text{phase 1}}\overbrace{\underbrace{3,5,3}_{marked}\,,2,5,}^{\text{phase 2}}\overbrace{1,2,3,}^{\text{phase 3}}\overbrace{\;4}^{\text{phase 4}}$

$l_0$ $l_1$ $t$ $l_2$ $l_3$ $l_4$

At time $t \in [l_i + 1, l_{i+1}]$ a page is **marked** if it belongs to $\{p_{l_{i+1}}, \dots, t\}$. An online algorithm is called **marking algorithm** if it never discards a marked page.

**Alg: Flush when full (FWF):** If page is full, discard all pages.

**Theorem 1** *FWF is a marking algorithm.*

**Proof** The elements in the cache of FWF are exactly the marked pages. □

**Theorem 2** *Every marking algorithm is absolutely k-competitive.*

**Proof** For $k = 1$ every algorithm is optimal. For $k > 1$ consider sequence $\vec{P}$ and $L$ phases $[l_0 + 1, l_1], [l_1 + 1, l_2], \ldots, [l_{L-1} + 1, l_L]$. The cost for every marking algorithm can be at most $l \cdot k$. This happens, if we have $k$ page faults per phase.

Let us now consider the lower bounds for OPT. For this we construct *shifted phases*, where shifted phase $i$ is defined as $[l_i + 2, l_{i+1} + 1]$.

Request Sequence $\vec{P} = \overbrace{1, 2, 4, 2, 1}^{\text{phase 1}}, \overbrace{3, \underbrace{5, 3, 2, 5}_{\text{sh. phase 1}}}^{\text{phase 2}}, \overbrace{1, \underbrace{2, 3}_{\text{sh. phase 2}}}^{\text{phase 3}}, \overbrace{4}^{\text{phase 4}}$

As one can observe $|\{p_j : j \in \{[l_i + 2, l_{i+1} + 1\}, p_j \neq p_{l_i+1}\}| = k$. At the beginning of each shifted phase $[l_i + 2, l_{i+1} + 1]$ the page $p_{l_i+1}$ is in the cache. Hence, *every* algorithm has at least one page fault per shifted phase. Since there are at least $L - 2$ shifted phases and the first phase leads to $k$ faults, the following approximation holds:

$$\text{OPT} \geq k - L - 2 \underset{k \geq 2}{\geq} L \implies \frac{\text{cost of marking alg.}}{\text{OPT}} \leq \frac{L \cdot k}{L} = k$$

□

**Corollary 3** *FWF is absolute k-competitive.*

**Theorem 4** *No deterministic online algorithm has competitive ratio $r < k$ for the caching problem.*

**Proof** We construct the request sequence $\vec{P}$ of length $l + k$, where $l$ will be chosen later. The sequence contains $k + 1$ pages $p^{(1)}, \ldots, p^{(k+1)}$. Let $A$ be any online algorithm. Now we construct the sequence the following way:

**For** $i = 1, \ldots, k + 1$: $p_i = p^{(i)}$

**For** $i = k + 1, \ldots, l + k - 1$: If $A$ discards $p^{(i)}$ at time $i$: $p_{i+1} = p^{(i)}$

For this sequence the costs of $A$ are exactly $l + k$, as each request leads to a fault.

Let us now consider the offline algorithm *Furthes in Future* (FIF), that discards the page that will not be requested the longest. We note that $OPT \leq FIF$ (in fact it even holds that $OPT = FIF$). On the first $k + 1$ pages of our constructed sequence $\vec{P}$, FIF has $k + 1$ page faults.

Following time $k + 1$, the pages $\overbrace{p_{k+1}, \ldots, p_{2k+1}}^{\leq k-1 \text{ pages}}$ are requested. FIF has no faults for requests $p_{k+2}, \ldots, p_{2k}$, but might fault on request $p_{2k+1}$. Following this, FiF again has no faults for requests $p_{2k+2}, \ldots, p_{3k}$, but might fault on request $p_{3k+1}$. Therefore, the follwing approximation holds:

$$FIF \leq k + 1 + \lceil \frac{l}{k} \rceil \leq k + 2 + \frac{l}{k}$$

Thus, we have: $\frac{A}{\text{OPT}} \geq \frac{l+k}{k+2+\frac{l}{k}}$. By choosing $l > \frac{(r-1)\cdot k+2r}{1-\frac{r}{k}}$ we get $\frac{l+k}{k+2+\frac{l}{k}} \geq r$ for all $r < k$. □

**Corollary 5** *FWF is an optimal deterministic online algorithm for the caching problem.*