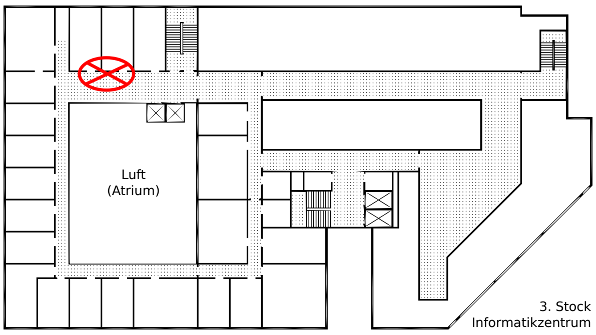
**Abteilung Algorithmik** **Summer term 2016**
**Institut für Betriebssysteme und Rechnerverbund**
**TU Braunschweig**

Prof. Dr. Sándor P. Fekete
Phillip Keldenich

# Online Algorithms
# 1st Homework Assignment, 27th of April, 2016

Solutions are due on Wednesday, the 11th of May, 2016, until 11:30 in the homework cupboard. You can also hand in your solution in person before the small tutorial begins. External students can hand in the homework via e-mail to `c.rieck@tu-bs.de`.



**Please make sure your homework submissions are clearly labeled using your name and/or matriculation number.**

**Exercise 1 (The Bahncard Problem I):** In the big tutorial, we presented the Bahncard Problem and proved that no online algorithm can perform better than $2 - \beta$ times the cost of an optimal offline algorithm. Construct an optimal offline algorithm that, for a given sequence $\sigma$ consisting of $n$ chronologically ordered ticket requests, produces an optimal solution in $\mathcal{O}(n)$ time.

You may make use of the following two facts:

- The optimal offline algorithm never has to buy a Bahncard while it still owns one.
- The optimal offline algorithm never has to buy a Bahncard at a time point that is not the time point of some ticket request.

**(20 points)**

**Exercise 2 (The Bahncard Problem II):** For the Bahncard problem, we presented the online algorithm SUM and presented a theorem by which SUM is $(2-\beta)$-competitive. Recall that a request is called a *reduced* request if SUM posseses a Bahncard for that request and *regular* otherwise, and the *break-even price* $c^*$ is $\frac{C}{1-\beta}$. For every triple $0 < \beta < 1, T > 0, C > 0$ of positive real parameters construct a worst-case input sequence $\sigma$ of constant size that makes the ratio $\frac{c_{SUM}}{c_{opt}}$ come arbitratily close to $(2-\beta)$. **(20 points)**

```
Input: Sequence $\sigma = \big((t_i, c_i)\big)_{1\le i\le n}$ of travel requests, $T$, $\beta$, $C$
Output: $\gamma = \big(\gamma_i\big)_{1\le i\le n} \in \{0,1\}^n$, where $\gamma_i = 1$ means buying a BC at request $i$
  if We already own a BC at request $i$ then
    Output $\gamma_i = 0$
  else
    if The cost of all regular requests in $(t_i - T, t_i]$ is at least $c^*$ then
      Output $\gamma_i = 1$
    else
      Output $\gamma_i = 0$
    end if
  end if
```

**Algorithm 1:** Online algorithm SUM for the Bahncard problem

**Exercise 3 (Paging):** In the paging problem, one has a cache that can contain up to $k$ pages, and a total of $n \ge k$ different pages. The input sequence $\sigma$ consists of requests for certain pages. In order to satisfy a request, the requested page must be brought into the cache if it is not already present; this is called a *fault* and incurs a cost of 1. In this case, a page that is currently in the cache must be *evicted*, i.e. removed from the cache to make room for the requested page. An online algorithm for the paging problem is fully defined by the strategy by which it chooses the page to be evicted.

Consider the following online algorithms on a paging problem with $n = 5$ pages and cache size $k = 4$:

**Least Recently Used (LRU)** The algorithm LRU always evicts the cached page for which the last request lies farthest in the past.

**First In First Out (FIFO)** The algorithm FIFO always evicts the *oldest* cached page, i.e. the page that has been in the cache for the longest time.

**Furthest in the future (OPT)** The optimum offline algorithm OPT always evicts the page for which the next request lies farthest in the future.

Perform the actions of each of the three algorithms, given an initial cache content $(1, 2, 3, 4)$ and request sequence $(5, 1, 2, 3, 5, 4, 5, 2)$. If the page to be evicted is not unique according to the strategy, always evict the page with the lowest page number. How many faults does each of the strategies produce? For each step, indicate which pages are currently in the cache. **(20 points)**