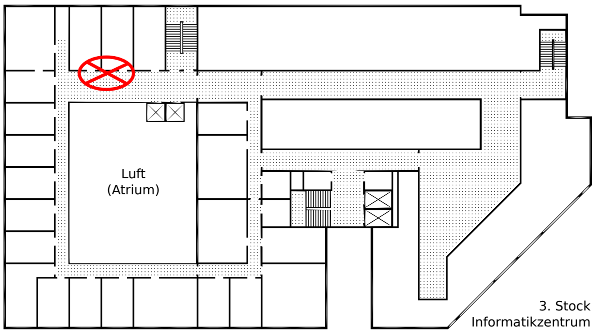
Abteilung Algorithmik **Summer term 2018**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Phillip Keldenich

Online Algorithms
1st Homework Assignment, 23rd of April 2018

Solutions are due Monday, the 7th of May 2018, until 1:15 PM in the homework cupboard. You can also hand in your solution in person before the small tutorial begins. If you cannot hand in the homework in person, you can also hand it in via e-mail to both `j.heroldt@tu-bs.de` *and* `keldenich@ibr.cs.tu-bs.de`. Please clearly label your solutions using your name and matriculation number.



Exercise 1 (The BahnCard Problem: Optimal Offline Algorithm): In the big tutorial, we considered the BahnCard Problem $BC(C, \beta, T)$ with cost C, cost reduction β and validity duration T. We proved that, in the worst case, 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 σ consisting of n chronologically ordered ticket requests $(t_1, c_1), \ldots, (t_n, c_n)$, 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.

Prove that your algorithm is correct and that its running time is $\mathcal{O}(n)$. **(20 points)**

Exercise 2 (The BahnCard Problem: Online Algorithm SUM): For the BahnCard problem $BC(C, \beta, T)$, we presented the online algorithm SUM. Recall that a request is called a *reduced* request if SUM possesses a BahnCard for that request and *regular* otherwise, and the *break-even price* c^* is $\frac{C}{1-\beta}$.

Input: Sequence $\sigma = \big((t_i, c_i)\big)_{1\leq i\leq n}$ of travel requests, T, β, C
Output: $\gamma = \big(\gamma_i\big)_{1\leq i\leq n} \in \{0,1\}^n$, where $\gamma_i = 1$ means buying a BC at request i

```
if We already own a BC at request i then
    Output γ_i = 0
else
    if The cost of all regular requests in (t_i − T, t_i] is at least c* then
        Output γ_i = 1
    else
        Output γ_i = 0
    end if
end if
```

Algorithm 1: Online algorithm SUM for the BahnCard problem

Let $\sigma = (t_1, c_1) \ldots (t_n, c_n)$ be a sequence of travel requests. Moreover, let $\tau_1, \ldots, \tau_k$ be the times where the optimal offline solution buys a BahnCard and consider the *phases* $[0, \tau_1), [\tau_1, \tau_2), \ldots, [\tau_k, \infty)$. We prove that SUM is $(2-\beta)$-competitive by proving $c_{\mathrm{SUM}} \leq (2-\beta)\cdot c_{\mathrm{OPT}}$ for each phase individually.

a) Recall that we call a time interval $I = [b, e)$ *expensive* if the sum of all costs for travel requests with time $t_i \in I$ is at least c^*, and *cheap* otherwise. Prove that for each phase $[\tau_i, \tau_{i+1})$ with $1 \leq i \leq k$, the interval $[\tau_i, \tau_i + T)$ is expensive. Moreover, let $\tau_{k+1} := \infty$. Prove that any subinterval of $[\tau_i + T, \tau_{i+1})$ of length at most T is cheap.

b) Prove that, for the first phase $I = [0, \tau_1)$, $c_{\mathrm{SUM}} \leq c_{\mathrm{OPT}}$.

c) Prove that $c_{\mathrm{SUM}} \leq (2-\beta)\cdot c_{\mathrm{OPT}}$ for a phase $I = [\tau_i, \tau_{i+1})$ if SUM does not buy a BahnCard in phase I.

d) Finally, prove that $c_{\mathrm{SUM}} \leq (2-\beta)\cdot c_{\mathrm{OPT}}$ for a phase $I = [\tau_i, \tau_{i+1})$ if SUM buys a BahnCard in phase I. *Hint:* Decompose I into three intervals I_1, I_2, I_3 based on the time until which SUM possesses a BahnCard from the last phase and the time where SUM decides to buy a new BahnCard.

(3 + 4 + 8 + 10 points)

Exercise 3 (The k-Server Problem): In this exercise, we consider the k-server problem in the Euclidean plane. In this problem, we start with k servers at given points $p_1^0, \ldots, p_k^0 \in \mathbb{R}^2$. We are given a sequence of n requests coming from locations $r_j \in \mathbb{R}^2$. Each request r_j must be handled immediately by moving one of the servers from its current position to r_j. The cost for moving a server from a position $p \in \mathbb{R}^2$ to r_j correspond to the Euclidean distance $||r_j - p||_2$. In other words, the goal is to minimize the sum of distances travelled by all servers.

The algorithm Greedy always chooses the cheapest possibility to serve the current request. In other words, it always serves an incoming request using the server closest to the request. Show that, even for $k = 2$ servers, this greedy strategy is not c-competitive for any constant c. **(15 points)**