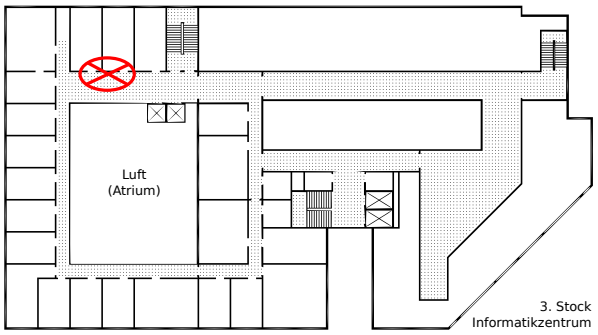
Abteilung Algorithmik **Summer term 2016**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Phillip Keldenich

Online Algorithms
2nd Homework Assignment, 11th of May, 2016

Solutions are due on Wednesday, the 25th 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 k-Server Problem): In this exercise, we consider the k-server problem: A company offers a service - for instance a mobile car repair service. Without prior knowledge, the company gets a sequence of service requests that have to be served in the order they come in. Each request comes from a certain location - in our example, this would be the locations of broken-down cars in a road network. In order to service requests, the company controls k mobile servers (for instance repair cars). For each request, one server has to move from its current location to the location of the request, incurring a cost of d (where d is the distance travelled by the server). In this model, travelling and servicing a request take no time. For a sequence of requests, the goal is to miminize the total distance travelled (the sum of the distances travelled by each server).

The algorithm Greedy always chooses the cheapest possibility to service the current request: it always uses the server closest to the requested location. Show that, even for $k = 2$, this greedy strategy is not c-competitive for any constant c. **(15 points)**

Exercise 2 (Deterministic lower bound for list update algorithms): In the big tutorial, we presented self-organizing datastructures, especially self-organizing linked lists. Recall that, in the list update problem, one has to maintain a list of n elements, minimizing the cost of a request sequence $\sigma = (\sigma_i)$ consisting of queries Query(x) for some element x in the list (in this exercise, assume that x is always present in the list). Each request for element x at position i in the list incurs a cost of i. After each request, we may move the requested element x further to the front of the list for free (this is called a *free exchange*). We may also swap any number of pairs of adjacent elements that occur before x in the list. Each such swap has an additional cost of 1 (this is called a *paid exchange*).

We discussed the algorithm Move-To-Front and demonstrated its 2-competitiveness. Show that there can be no deterministic list update algorithm $\mathcal{A}$ that achieves a competitive factor for a constant $c < 2$.

Hint: On a list of fixed size n, consider an arbitrarily long worst-case sequence σ consisting of Query(x)-requests only and compare the performance of $\mathcal{A}$ to that of an algorithm that first sorts the list in a suitable fashion. **(25 points)**

Exercise 3 (Randomized adversary models): In competitive analysis of a randomized online algorithm $\mathcal{A}$, one compares the expected cost of $\mathcal{A}$ to the cost of an adversary. However, unlike the situation for deterministic algorithms, there is more than one natural way to define the adversary.

In the lecture, the concept of an *oblivious adversary* was used in the analysis of the randomized marking algorithm RMA: An oblivious adversary has to prepare the entire input sequence before the randomized online algorithm runs and he cannot predict the random choices made by the algorithm. However, this adversary knows the entire sequence in advance and thus has to pay the price of the optimal offline solution.

In this exercise, we will consider a stronger kind of adversary, a so-called *adaptive online adversary*. An adaptive online adversary $\mathcal{D}$ does not have to fix the entire input sequence in advance. Instead, $\mathcal{D}$ can construct the request sequence σ one element at a time. When constructing σ_i, $\mathcal{D}$ knows all decisions $\mathcal{A}$ has made so far, including the responses of $\mathcal{A}$ to all requests up to σ_{i-1}. However, $\mathcal{D}$ still cannot predict random choices $\mathcal{A}$ will make in the future. The costs of $\mathcal{D}$ are measured based on a response sequence γ computed by $\mathcal{D}$. $\mathcal{D}$ has to produce its response γ_i together with σ_i (without knowing how $\mathcal{A}$ will react to σ_i). Thus, $\mathcal{D}$ can be considered an online algorithm itself.

In the lecture, the randomized marking algorithm RMA was shown to be $2H_k$-competitive against any oblivious adversary. Show that RMAs competitive ratio against any adaptive online adversary can not be better than $\frac{(k-1)k+1}{2k}$. **(20 points)**