Algorithms Group **Summer 2020**
Department of Computer Science - IBR
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Phillip Keldenich

Online Algorithms
Exercise 1
May 5, 2020

Hand in your solutions as PDF file until May 20, 2020, 11:30 AM via e-mail to `v.sack@tu-bs.de`, with CC to `keldenich@ibr.cs.tu-bs.de`. If you cannot turn your solution into a PDF file (for example by writing it in LaTeX or Word), you can also submit photographs or scans. In that case, be careful to keep the file size acceptable (about 3 MB per page) by using appropriate compression and resolution; however, make sure that your solutions are still readable.

Exercise 1 (Memory):
In this exercise, we consider a single-player version of the card game MEMORY. There is a deck of n *pairs* of cards lying face down on the table in front of the player. The player has to remove the cards by uncovering matching pairs of cards in as few moves as possible.
In each move, the player selects a first card and turns it around to see its face. After that, he selects and uncovers a second card. If the two cards match, they are removed. Otherwise, the cards are turned face down and remain on the table.
Knowing all cards' positions, the optimal offline algorithm removes the cards in n moves.

(a) Design an online algorithm for MEMORY that requires at most $2n - 1$ moves.

(b) Prove that there is no online algorithm that requires at most $2n - 2$ moves.

(c) In the two-player game, a player gets another free move after uncovering a matching pair of cards. In the single player variant, this corresponds to reducing the costs of a move to 0 if that move successfully removes a matching pair of cards. Give a c-competitive online algorithm for some constant $c > 1$ for this version of MEMORY, or prove that no such algorithm exists.

(5+10+5 pts.)

Exercise 2 (Potential Functions and Amortized Analysis):
Consider an abstract online problem where an online algorithm A faces an online sequence $r = r_1 r_2 \dots r_n$ of *requests*. As a response to each request r_i, A has to perform an action $A(i)$ without knowing the next request r_{i+1}. Each such action incurs a cost $c_A(i) \in \mathbb{R}$. Analogously, the optimal offline algorithm OPT performs actions as response to requests with costs $c_{\mathrm{OPT}}(i)$.

In the analysis of online algorithms, it is often impossible to bound the cost of an online algorithm by proving $c_A(i) \leq c \cdot c_{\mathrm{OPT}}(i)$ for each request i. Therefore, we need a way to distribute the costs of an expensive action of A across several requests.
One way of doing this is by considering a *potential function* $\Phi_r : \{1, 2, \ldots, n\} \to \mathbb{R}_{\geq 0}$ with $\Phi_r(0) = 0$. This potential function acts as a savings account that is not allowed to become negative and that accumulates saved costs to pay for later expensive actions.

(a) Prove the following. If for every request sequence r, there is a potential function Φ_r such that

$$c_A(i) + \Phi_r(i) - \Phi_r(i-1) \leq c \cdot c_{\mathrm{OPT}}(i),$$

then A is c-competitive, i.e., $\sum_{i=1}^{n} c_A(i) \leq c \sum_{i=1}^{n} c_{\mathrm{OPT}}(i)$.

(b) Consider the problem Read Into Buffer, where we want to read a non-empty stream s of unknown length into a buffer that is stored in memory as contiguous array of size at most $2|s|$. Reading a symbol from s into the buffer has a cost of 1. The optimal offline algorithm allocates an array of size $|s|$ once and thus has a cost of $|s|$.

In the online scenario, if the buffer is full, it has to be reallocated and the old contents have to be copied to the new buffer. For every symbol already in the buffer, this incurs an additional cost of 1. Thus, reading the kth symbol from s costs either 1 (not full) or k (buffer full).

Devise a 3-competitive algorithm for Read Into Buffer, using a potential function to prove the competitive ratio.

(5+15 pts.)