Algorithms Group **Summer 2022**
Department of Computer Science - IBR
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Michael Perk

Online Algorithms
Homework Assignment 4
June 27, 2022

Solutions are due on Monday, the 04th of July, 2022, until 15:00 in the homework cupboard. You can also hand in your solution in person before the big tutorial begins or via e-mail to `kramer@ibr.cs.tu-bs.de` with CC to `mperk@ibr.cs.tu-bs.de`.

Exercise 1 (Cow Path Problem):
Suppose that you are a hungry cow on a path. You know that at some distance $D \geq 1$ along the path there is a pasture with tasty grass; however, you do not know the distance D or the direction in which you have to go. In order to reach the pasture, you start by traveling into one of the two directions for one unit of distance. If you do not find your food, you return to the start and then travel two units of distance into the other direction. If you still have not found your food, you return to the start and then travel four units of distance into the first direction, and so on. You repeat this procedure, doubling the distance each time you return back to the start, until you finally reach your goal.

a) Prove that this strategy is 9-competitive, i.e., that you never have to travel more than $9D$ units of distance until you reach the goal. Moreover, prove that this is tight, i.e., that there is no constant $c < 9$ such that this strategy is c-competitive.

b) Prove that the restriction $D \geq 1$ is vital, i.e., that without it, there is no c-competitive strategy for any constant c.

(15+5 pts.)

Exercise 2 (Move To Front):
In this exercise, we consider the MoveToFront algorithm for the List Update Problem. After each request s_i, the algorithm moves the requested item s_i to the front of the list. For example, after searching for 2 in $[1, 4, 2, 3]$, the list becomes $[2, 1, 4, 3]$.

a) Prove that there is no constant $c < 2$ such that Move To Front is c-competitive.

b) Prove that Move To Front is 2-competitive.

Hint: Use the number of inversions in Move To Front's list w.r.t. OPT's list after request i as a potential function $\phi(i)$. An inversion is a pair x, y of elements such that x comes before y in Move To Front's list, but after y in OPT's list.

In order to prove $c_{\text{MTF}}(i) + \phi(i) - \phi(i-1) \leq 2c_{\text{OPT}}(i)$ for a request s_i, consider the number of items k that come before s_i in both OPT's and Move To Front's list, the number of items m that come before s_i in OPT's list but after s_i in Move To Front's list, and the number of items ℓ that come before s_i in Move To Front's list but after s_i in OPT's list.

(Bonus: 10+15 pts.)

Exercise 3 (List Update Algorithms):
In the previous exercise, we considered the list update problem. We are maintaining a set of keys in a linked list and are given a sequence of queries σ for keys. For each query, we iterate through the list starting from the first element. Each element we access in our search costs us 1 unit. Searching for 2 in $[2, 1, 4, 3]$ costs 1, in $[1, 2, 3, 4]$ costs 2 and in $[1, 3, 4, 2]$ costs 4 units. After finding the queried element, we can move it to any point further to the front without extra cost. After finding 2 in $[1, 3, 2, 4]$, which costs 3 units, we may thus change the list to $[2, 1, 3, 4]$ or $[1, 2, 3, 4]$ or keep $[1, 3, 2, 4]$. We showed that the algorithm Move To Front is 2-competitive. Prove that no deterministic list update algorithm can be c-competitive for any $c < 2$. **(Bonus: 20 pts.)**