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

Prof. Dr. Sándor P. Fekete
Phillip Keldenich

Online Algorithms
Exercise 5
July 1, 2020

Hand in your solutions as PDF file until July 15, 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 (Dynamic Monotone Trees):
In the tutorial, we presented the dynamic binary search tree algorithms RotateToRoot, SingleRotation and Splay. All these algorithms have in common that they do not use additional memory compared to a regular binary search tree. Such strategies are also called *stateless*.
In this exercise, we consider a generalization of the FrequencyCount list update algorithm to trees. In a so-called *dynamic monotone tree*, each element stores a counter that maintains how often the element has been searched for. The counters are all initialized to 0. Whenever we search for an element, we increase its counter and rotate it up until its counter is not bigger than the counter of its parent. In other words, we maintain a search tree which is also in max-heap order with respect to the access counters.
Prove that dynamic monotone trees are not better than $\Omega(n)$-competitive. **(10 pts.)**

Exercise 2 (Looking Around a Corner):
In the lecture, we considered the problem of looking around a corner with scan costs. In this situation, a robot it located at a known distance d from a corner. Distances are measured in travel time, normalized by scan time, i.e., traveling distance d takes the same time as d scans. Behind the corner, at an unknown angle, there is some hidden object. The goal is to move to the corner, performing scans along the way until the object is seen, minimizing the time taken for traveling and scanning.
We restrict ourselves to traveling on a polygonal chain inscribed in a semicircle of diameter d defined by the corner C and our starting location s. Given any fixed competitive ratio c we want to achieve, we can compute the distances x_i traveled between the ith and the $(i-1)$st scan using a recursive formula as presented in the lecture:

$$x_{i+1} = c\underbrace{(1+d_i)}_{\text{offline cost}} - \underbrace{(1+i)}_{\text{scan cost}} - \underbrace{\sum_{j=1}^{i} x_i}_{\text{travel cost}} .$$

Here, d_i denotes the distance from the starting point to the ith scan point. For the first step, set $d_0 = 0$ and thus $x_1 = c - 1$. The distance d_i can be computed from the angles $\varphi_i = 2\arcsin(\frac{x_i}{d})$ covered by the ith segment as

$$d_i = d \sin\left(\frac{1}{2}\sum_{j=1}^{i}\varphi_j\right).$$

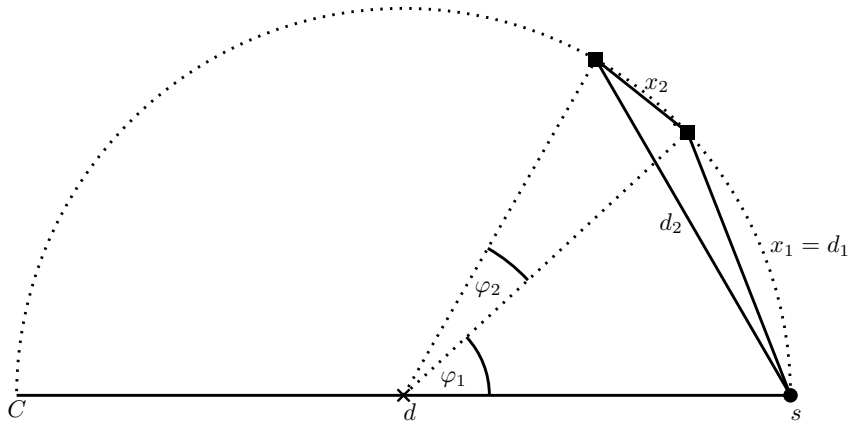

Figure 1: The first two steps of looking around a corner

Depending on the parameter c, this recursion either reaches the corner (after finitely many steps, the angle covered reaches or exceeds π) or collapses (distance we are allowed to travel becomes non-positive). In the first case, a competitive ratio of c is achievable by traveling on a semicircle; otherwise, it is not.

a) Why does the robot have to move to the corner?

b) Prove that moving to the corner directly is not c-competitive for any $c \geq 1$.

c) Using C++, Java or Python, implement a program or function that is able to check for a given pair c, d, whether a competitive ratio of c is achievable for the starting distance d.

d) Use your program in conjunction with the *bisection method* (binary search for the right value) to find the best achievable competitive ratio for $d = 30$. You may ignore rounding errors for the purpose of this exercise.
Check: For $d = 40$, the factor c satisfies $2.0015 < c < 2.0016$.

(5+5+12+8 pts.)