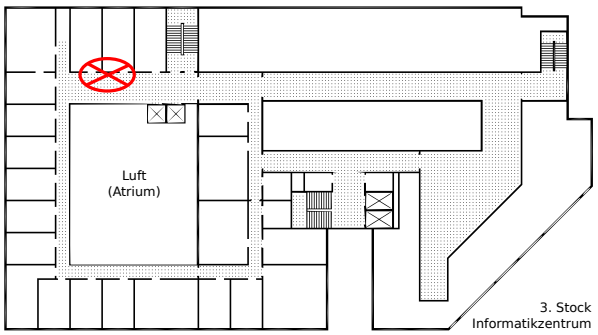
Abteilung Algorithmik **Summer term 2018**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Phillip Keldenich

Online Algorithms
4th Homework Assignment, 11th of June 2018

Solutions are due Monday, the 25th of June 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 (Bounded Size Bin Packing): In the tutorial, we have considered the online problem Bin Packing where one is given a sequence $a_1, \ldots, a_n$ of items with weights $a_i \in (0, 1]$ and has to pack these items into as few bins of capacity 1 as possible. In the cases we have seen, the biggest losses for online algorithms were caused by large items in the input sequence. In this exercise, we consider the case where the weight a_i of each item is bounded by a constant $\alpha \in (0, 1)$ known to the algorithm, i.e., all items satisfy $a_i \leq \alpha$.

Consider the algorithm Next Fit presented in the tutorial. For each value $\alpha \in (0, 1)$, determine (with proof) the asymptotic competitive ratio $c_{NF}^{\infty}(\alpha)$ of Next Fit if all items satisfy $a_i \leq \alpha$. **(12 points)**

Exercise 2 (Bin Packing with Unlimited Capacity): In this exercise, we consider a variant of Bin Packing that has applications in task scheduling. In this variant of Bin Packing, we receive a sequence of items $a_i \in (0, \infty)$; this modifies the original problem setting by allowing items of arbitrary size. Moreover, we have a fixed number of m bins with unlimited capacity; as in the original Bin Packing problem, we have to pack each incoming item into a bin that we have to fix before the next item is given to us. The goal is to distribute the weight onto the bins as evenly as possible. More formally, we want to minimize the total weight W packed into the fullest bin.

In this exercise, we consider the following algorithm Greedy for this problem. Initially, all bins are empty. On receiving the ith item a_i, Greedy packs a_i into the emptiest bin; ties are broken arbitrarily. For an example of the algorithm, refer to Figure 1.

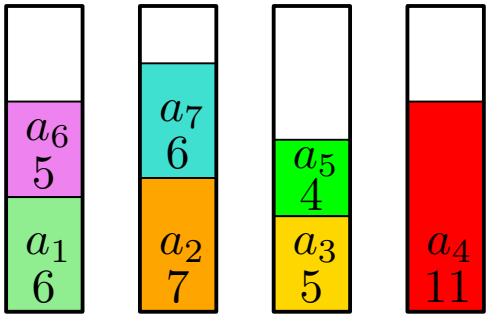


Figure 1: The packing that results from applying GREEDY to $m = 4$ bins and the input sequence $(6, 7, 5, 11, 4, 5, 6)$. The quality of the solution produced by GREEDY is $W_G = 13$ (the fullest bin contains $7 + 6 = 13$ units of weight). The optimal solution has quality $W_O = 11$.

a) Prove that GREEDY cannot be better than $(2 - \frac{1}{m})$-competitive. *Hint: Consider a sequence that begins with small items and ends with a big item.*

b) Prove that GREEDY is $(2 - \frac{1}{m})$-competitive. *Hint: Consider the last item a_T placed into the fullest bin and the value $W_G - a_T$.*

(10+15 points)

Exercise 3 (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

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

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.

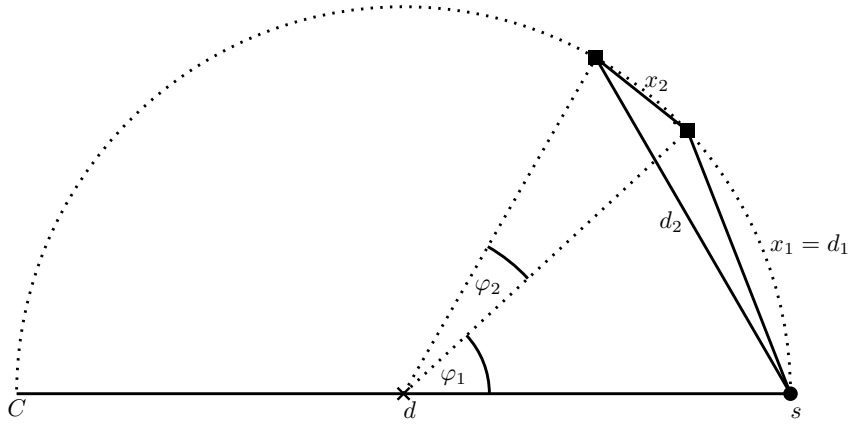


Figure 2: The first two steps of looking around a corner

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 a common programming language of your choice, implement a program 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* to find the best achievable competitive ratio for $d = 30$. Hint: For $d = 40$, the factor c satisfies $2.0015 < c < 2.0016$.

(1+2+12+10 points)