Department of Mathematics and Computer Science
University of Southern Denmark, Odense

November 17, 2016
KSL

Exercises for Week 47 on Online Algorithms

a topic in DM534 – Introduction to Computer Science

Kim Skak Larsen
Fall 2016

The exercises are of significantly varying difficulty. They are organized in order of topic: ski rental, machine scheduling, and bin packing. Thus, if you find any problem too hard, go to the next topic before you return to the harder problems. A number of extra assignments in different categories are listed at the end.

Ski Rental

Exercise 1

Prove that no matter which other algorithm than the one from the lecture notes we define for ski rental, the algorithm will perform worse, i.e., the competitive ratio will be strictly higher than $\frac{19}{10}$.

Start by analyzing the algorithms "Buy on day 5" and "Buy on day 15" to see what happens. The skis still cost 10 units to buy and 1 unit per day to rent.

Exercise 2

Let us consider the smaller variant of the ski rental problem, where skis cost 4 units. Clearly, the best algorithm we can design is "Buy on day 4", which has competitive ratio $\frac{7}{4}$.

We introduce a new element into our code: A coin flip! This is known as a *randomized* algorithm.

Consider the algorithm that rents the first two days, flips a coin on the third day and buys/rents based on that, and buys on the fourth day if it didn't buy already.

Obvious worst-case sequences to consider giving our algorithm are

it's a new day, it's a new day, it's a new day, come home

and

it's a new day, it's a new day, it's a new day, it's a new day

Separately for each sequence, compute our algorithm's *expected* cost (the average of the two possibilities resulting from the coin flip on day 3), compute OPT's cost, and then the ratio.

For which sequence do we get the worst expected cost? (That's the sequence the adversary will give us.)

Is that better or worse than the $\frac{7}{4}$-competitive deterministic algorithm?

Exercise 3

Implement the above algorithm in PYTHON and verify your calculations by computing the average cost for each sequence over 1000 runs of the algorithm.

Coin flips can be implemented like this:

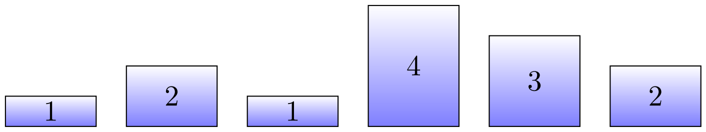
```
from random import random

if random() < 0.5:
    print("head")
else:
    print("tail")
```

Machine Scheduling

Exercise 4

For $m = 3$, which schedule does the List Scheduling algorithm, LS, produce on the following input sequence:



Exercise 5

In the lecture, we proved that the machine scheduling algorithm, LS, could not perform better than $2 - \frac{1}{m}$. We now consider only two machines. Thus, $m = 2$, and the ratio is then $\frac{3}{2}$. Just because LS cannot perform better, it could be that some other algorithm could. Prove (for $m = 2$) that this is not the case. You must design an input, where *no* algorithm, no matter what decisions it makes, can do better than $\frac{3}{2}$ times OPT. You only need sequences with two and three jobs and a case analysis with only two cases, depending on what an algorithm does with the second job that is given.

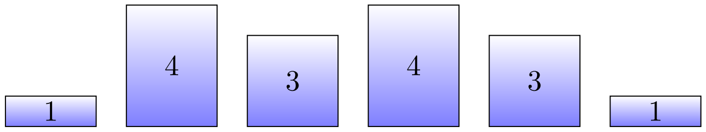
Bin Packing

Exercise 6

Consider the first bin packing example given in the lecture, where the First-Fit algorithm, FF, uses four bins. Show that OPT only needs three.

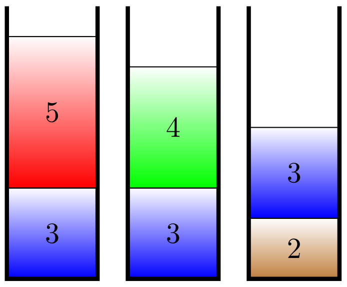
Exercise 7

How does the First-Fit algorithm, FF, behave on the input sequence below? Item sizes are given in multiples of $\frac{1}{6}$.



Exercise 8

Why can the following configuration *not* have been produced by the bin packing algorithm FF? Item size are given in multiples of $\frac{1}{9}$.



Exercise 9

For bin packing, one can prove the upper bound that FF is 1.7-competitive. However, this is a quite hard proof. In this exercise, we will try to improve (raise) the lower bound.

In the lecture, we saw an example demonstrating that FF can be as bad as $\frac{3}{2} = 1.5$ times OPT.

Let that example inspire you, and try to use items of the following three sizes:

$$\frac{1}{7} + \frac{1}{1000}, \frac{1}{3} + \frac{1}{1000}, \frac{1}{2} + \frac{1}{1000}$$

Find a sequence where FF performs $\frac{5}{3}$ times worse than OPT.

Now try using

$$\frac{1}{43} + \frac{1}{1000}, \frac{1}{7} + \frac{1}{1000}, \frac{1}{3} + \frac{1}{1000}, \frac{1}{2} + \frac{1}{1000}$$

to get a lower bound close to the 1.7 upper bound.

Exercise 10

It is very easy to implement FF in PYTHON, if there are no efficiency requirements: just use a list to hold the current level in the bins, and for each item, search for the first bin with enough space. If you make sure there are enough bins from the beginning, then there are no special cases. And you simple count the number of non-empty bins at the end to get the result.

Implement FF.

Try to define your own algorithm, from scratch or as a variant of FF. Test your own algorithm up against FF and try to determine which one is best; for instance on uniformly distributed sequences, i.e., each item size is chosen like this:

```
from random import seed, random

seed(42)
.
.
item = random()
```

You don't have to use `seed`; the intention is simply to provide a *reproducible* random sequence, i.e., `seed` is called once and the seed value initializes the sequence.

Extra Exercises

If you extra have time or interest, you can consider the following problems.

Exercise 11

[Hard] In the set-up from Exercise 3, consider choosing what to do on the third day with a probability different from $\frac{1}{2}$. What is the best probability to choose? What does the expected ratio become?

Exercise 12

A core problem for an online machine scheduling algorithm is whether to make a level or a skewed schedule. For instance, just considering $m = 2$ and the sequences 1, 1 and 1, 1, 2, we don't know what the best option is for the second job:

placing it on the *same* machine as the first job *or not*. (Because this depends on the future: do we receive a third job of size 2 or not?)

Try to randomize the decision as you did earlier for ski rental. Assuming you get one of these two sequences, can you get an expected ratio smaller than $\frac{3}{2}$?

Exercise 13

Finish left-over discussions from the study group on online algorithms.