Institut for Matematik og Datalogi
Syddansk Universitet, Odense

29. november 2019
KSL/JFB/RF

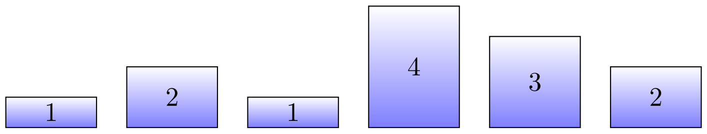
Eksaminatorier DM534 Uge 49

Online Algorithms

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.

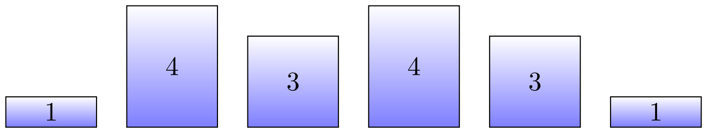
2. For $m = 3$, which schedule does the List Scheduling algorithm, Ls, produce on the following input sequence:



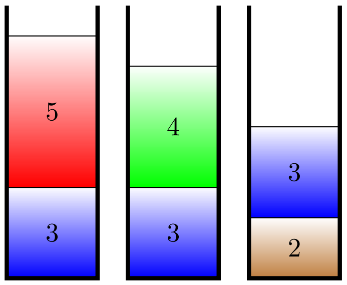
3. In the lecture, we proved that the machine scheduling algorithm, Ls, could not perform better than $2 - \frac{1}{m}$. We now consider only two machines. Thus, $m = 2$, and the ratio is then $\frac{3}{2}$. Just because Ls cannot perform better, it could be that some other algorithm could. Prove (for $m = 2$) 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.

4. Consider the first bin packing example given in the lecture (slide 19), where the First-Fit algorithm, Ff, uses four bins. Show that Opt only needs three.

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



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



7. 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:
$$\frac{1}{7}+\frac{1}{1000}, \quad \frac{1}{3}+\frac{1}{1000}, \quad \frac{1}{2}+\frac{1}{1000}$$
   Find a sequence where FF performs $\frac{5}{3}$ times worse than OPT.

   Now try using
$$\frac{1}{43}+\frac{1}{10000}, \quad \frac{1}{7}+\frac{1}{10000}, \quad \frac{1}{3}+\frac{1}{10000}, \quad \frac{1}{2}+\frac{1}{10000}$$
   to get a lower bound close to the 1.7 upper bound.

8. It is very easy to implement FF in JAVA, if there are no efficiency requirements: just use an array 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.

In Java, one way to create pseudorandom floating point numbers uniformly distributed in the interval $[0, 1[$ is via the class `java.util.Random` and its `NextFloat` method. You may want to look at the tutorial here: `http://www.functionx.com/java/Lesson18.htm`. If you want to generate the same sequence of pseudorandom numbers in different invocations of your program (for instance to compare two algorithms on the same input sequences), you must set the seed in the random number generator at the start of the program (otherwise it is automatically set to a different seed at each invocation). This is also described in the tutorial.

Cryptology

9. Suppose a public RSA key is $PK = (1517, 13)$. Which of the following is the RSA encryption of the message 43?

   (a) $1517^{43} \pmod{13}$

   (b) $43^{13} \pmod{1517}$

   (c) $13^{43} \pmod{1517}$

   For the right answer, we want to use the algorithm for fast modular exponentiation (page 33 on the slides). How many times during the recursive execution is the "**if** k is odd" case encountered, and how many times is the "**if** k is even" case encountered? [Do not include the base cases $k = 0$ and $k = 1$ in the counts.]

10. Why is a cryptographically secure hash function used in connection with RSA digital signatures?

11. With RSA, why would you never use the value 2 as one of of the two primes p and q?

12. In RSA, why must the message being encrypted be a non-negative integer strictly less than the modulus?

13. If you have time, for fun try breaking these two encrypted messages:

    - This English message was encrypted using a Caesar cipher. Decrypt it.

      YMNX HWDUYTLWFR NX JFXD YT IJHNUMJW.

      Discuss which techniques you used.

    - This was entitled "Cold Country". It was encrypted using a monoalphabetic substitution cipher. A monoalphabetic substitution cipher works similarly to a Caesar cipher. However, instead of just shifting the alphabet a fixed amount to get the mapping defined for each letter, the key is a permutation of the alphabet, so that you decide according to this key what letter "A" maps to, what letter "B" maps to, etc. If the alphabet has 29 letters, the number of keys is now 29! Why? The original message here was in English, so there are only 26 letters. How many possible keys are there?

      TOWWJPHJC ZY RXW PHOTWYR ZYPHJC ZJ RXW SFOPC. UFYR FB ZR ZY QFIWOWC SZRX ZQW RXFMYHJCY FB BWWR CWWD.

      Discuss which techniques you used.