TD n°1 *Algorithmic Complexity*

Except in exercise 4, algorithms are given as programs in a programming language with integer variables and arrays. An upper bound of the cost of the basic operations are given in the table below. Exercise 4 asks you to prove that this table is correct. All that really matters is that they are in polynomial time.

Addition / Substraction of two integers x, y	$max(\log x, \log y)$
Comparison of two integers x, y	$min(\log x, \log y)$
Reading the content of a variable with value v	$\log v$ steps
Writing the content of a variable with value v	$\log v$ steps
Reading / writing in an array with total size s	s steps
Multiplication of two integers x, y	$max(\log x, \log y)^{\log_2 3}$
Euclidian division of two integers x, y	$max(\log x, \log y)^2$

Exercice 1. Majoritary element

Let w be a word on some alphabet A. A letter $a \in A$ is majoritary in w if more than half of the letters of w are a.

1.1 Give an algorithm which, on input a, *checks* whether a letter $a \in A$ is majoritary in a word w. What is its time complexity ? What is its space complexity ? *an approximation is enough*

1.2 Write an algorithm that *finds* which letter (if any) is majoritary in a word w. What is its time complexity ? What is its space complexity ?

1.3 Consider the following algorithm. Prove that if m is the majoritary element of w, then its output is m. What the time and space complexity of this algorithm ?

```
def get_candidate(w):
    candidate = None
    count = 0
    for current_letter in w:
        if count == 0:
            candidate = current_letter
        if current_letter == candidate:
            count = count + 1
        else:
            count = count - 1
    return candidate
```

1.4 Can you find a better algorithm than in question 2 for finding the majoritary element if it exists ? *indication : you may have to read the word twice*

Exercice 2. From Wang Tilings to polyominoes

Let X be a set of polyominoes, and S be a polyomino. POLYOMINO-TILING(S,X) is the problem of tiling S with translated copies of the elements of X (no rotations are allowed).

2.1 How to formulate this as a decision problem ? Define the encoding of its input.

2.2 Show that the problem is in NP : define a certificate for this problem, give the verification algorithm and check that it is in polynomial time.

2.3 Give a reduction from the problem WANG-RECTANGLE-TILING seen in the course. That is, explain :

— How to easily transformation any instance of WANG-RECTANGLE-TILING into an instance of POLYOMINO-TILING. *Your transformation has to be computable in polynomial time, but you don't need to detail the proof of its complexity.*

— A proof that the answer to WANG-RECTANGLE-TILING is "yes" for the instance x if only if it is "yes" for the instance $r(x)$ of POLYOMINO-TILING.

2.4 Prove that for any $S \subset \mathbb{Z}^2$, the problem POLYOMINO-TILING(S,-) of tiling S with copies of elements of a set of polyominoes X is in P.

2.5 Can you find a set P of polyominoes such that the problem POLYOMINO-TILING(-, P) of tiling a finite subset of $\mathbb{Z}^2$ with copies of elements of P is NP-complete ?

Exercice 3. Tiling a line with two bars

Let $L_n = \{0, \ldots, n-1\}$ be the n-cell, one-dimensional line. We want an algorithm to tile it with an a-cell long bar b_a and a b-cell long bar b_b.

3.1 How to formulate this question as a decision problem ? What are the inputs ? How can it be represented on the alphabet $\{0, 1, \#\}$; on how many tapes ? What is the size of the input ?

3.2 Give a necessary condition for L_n to be tileable with b_a and b_b.

3.3 Recall Euclid's algorithm, and calculate its complexity. *Hint : express what happens with the logarithms of a and b.*

3.4 Characterize the set of solutions in $\mathbb{Z}$ of the equation $au + bv = c$, and show that when $k \gcd(a,b) \geq \mathrm{lcm}(a,b)$, the equation $au + bv = k \gcd(a,b)$ has a non-negative solution.

3.5 Give a decision algorithm for the tileability of L_n with b_a and b_b ; does it have polynomial complexity ? *Hint : use the previous question to consider the case where* $\gcd(a,b) = 1$ *and* $n < ab$.

Exercice 4. Back to primary school

4.1 Give a Turing machine with 2 input tapes and 1 output tape which compares its two inputs (as integers) : given a and b as input, its output is 1 if $a \geq b$, and 0 otherwise. How many steps does it need to compare 3 and 5 ? How many steps does it need to compare n and m ?

4.2 Give a Turing Machine with 2 input tapes and 1 output tape which computes the sum of its inputs. How many steps does it need to add n and m ?

4.3 Give a Turing Machine with 2 input tapes and 1 output tape which computes the difference of its inputs, or zero if it is negative.

From now on, we give our Turing Machines as lists of instructions in english. We may use variables which can hold values for us ; each value is either an integer or a list of values. The time taken by each elementary operation is given by the table below :

Addition / Substraction of two integers x, y	$max(\log x, \log y)$
Comparison of two integers x, y	$min(\log x, \log y)$
Reading the content of a variable with value v	$\log v$ steps
Writing the content of a variable with value v	$\log v$ steps

4.4 How long does it take to multiply n by m with the algorithm you learnt in primary school ?

4.5 Notice that forall a, b, c, d, x, we have :

$$(ax + b)(cx + d) = acx^2 + [(a + b)(c + d) - ac - bd] + bd$$

. Suppose that multiplying two numbers x, y with $\log x = \log y$ takes time T, how long does it take to multiply two numbers n, m, with $\log n = \log m = 2 \log x$ using this formula ?.

4.6 Write an algorithm to multiply two numbers of any size using this formula. Let T_n be the maximum number of steps this algorithm needs in order to multiply two numbers x, y with $\max(\log x, \log y)$ What recurrence does its complexity T_n satisfy ? Check that $T_n = \Theta(n^{\log_2 3})$.

4.7 Write the primary-school algorithm to divide two integers, written in base 2. How many operations does it need ?

Exercice 5. Exact Cover

A pentomino is a polyomino with 5 cells. There are (up to rotation) 12 pentominoes. It is customary to try and tile figures with one copy of each pentomino. Exact-cover is a generalization of this problem.

Let U be a set, and P be a set of subsets of U. P has an exact cover of U if there is $P' \subset P$ which is a partition of U.

5.1 Take $U = \{1, 2, 3, 4, 5\}$, does $P = \{\{1, 4\}, \{1, 2, 3\}, \{3, 5\}, \{2, 3\}, \{5\}\}$ have an exact cover ?

5.2 Formulate the problem of covering a shape with one copy of each pentomino (no rotation allowed) as an exact cover problem.

5.3 Formulate the problem of covering a shape with one copy of each pentomino (with rotations allowed) as an exact cover problem.

5.4 Let X be a set of polyominoes, and $m : X \mapsto \mathbb{N}$. Formulate the problem of covering a shape with $m(x)$ copies of each pentomino x (with rotations allowed) as an exact cover problem.

5.5 Show that EXACT-COVER is NP-complete, by reducing from POLYOMINO-TILING.