Complexité Avancée: TD1
Esquisse de correction

Aliaume Lopez

25/09/2019

Exercise 1 : Sorting in logspace

INPUT : less than n numbers that are between 0 and m, separated by •

OUTPUT : the same amount of numbers that are sorted, separated by •

A proposition of algorithm (at home, try to improve it).

```
L <- list of numbers (input tape)
max  = NONE                # the maximal number written in the output tape
min  = NONE                # the minimal number in { x in L / x > max }
num  = NONE                # number of times «min» appears in L
len  = 0                   # number of output that have been written
while len < |L| do
  min = findMinimum   # searches through L to find the minimum
  num = countMinimum  # searches through L to count
  for i = 0 to num do
    write min
    write ·
    len += 1
  done
  max = min
done
```

Exercise 2 : Space constructible functions and $\log_2(n)$ issue

The objective was to prove the following inclusion :

$$\mathsf{NSPACE}(f(n)) \subseteq \mathsf{DTIME}(2^{O(f(n))} + O(n)) \tag{1}$$

We first build a naïve proof, and then find the hypotheses that are needed.

Let M be a Turing machine computing using space $f(n)$ without output tape (remember that it is a decision problem). However in the definition of $\mathsf{SPACE}(f(n))$, there is an input tape that is read-only.

Beware the input tape

For L (logspace) it is *necessary* to consider an input tape... Can you see why ?

Therefore, a reachable configuration of the machine M on input x has size bounded by,

$$|C_M(x)| \leq |Q| \times |x| \times f(|x|) \times |\Sigma_M|^{f(n)} \triangleq S_M(x) \tag{2}$$

Where Q is the states of M, x is the input and, Σ_M is the alphabet of M.

The key ingredient

Now, to detect if M accepts x, we proceed in two steps

1. Compute all the confiurations of size bounded by $S_M(x)$
2. Use a graph reachability algorithm to see if there is an accepting configuration that is reachable from the initial configuration.

However, it is crucial to see that there are things that are not well defined in the previous statement.

— How do you take a transition from the Turing machine M in the graph ?

— The graph of possible configurations of M is possibly infinite, but you only consider configurations of a bounded size. How do you compute the size $S_M(x)$? (hint : the function f must be space-constructible).

By answering the two preceding items, you should be able to describe a machine M' that accepts the same language as M but runs in time bounded by $2^{O(f(n))} + O(n)$.

Why would you assume that $f \geq \log n$

If $f \geq \log n$ then M' is actually running in time $2^{O(f(n))}$.

Exercise 3 : Restrictions of the definition of $\mathsf{SPACE}(f(n))$

As always in this kind of situation, the solution is using a *timeout* or *spaceout*.

1. When you have a machine that runs in space $f(n)$ but may not halt, you can detect an infinite run because the number of configurations is finite. As in the previous exercise you can see that this number can be bounded by $S_M(x)$

$$S_M(x) \triangleq |Q| \times |x| \times f(|x|) \times |\Sigma_M|^{f(n)} \tag{3}$$

   Therefore it suffices to have a counter C written in binary to stop the execution when C is too big.

2. When you have a machine that runs in space $f(n)$ if it accepts, and with no constraint otherwise. To prevent from cycling, the previous trick will still work. However, we also need a counter to constraint the space usage, also known as *spaceout*.

Why would you assume that $f \geq \log n$ ... part two

Check that otherwise the space of your newly defined Turing machine will not be $O(f(n))$ ...

Why you do need two counters for the second question

Check that using the counter C to bound the *time* of the computation will result in possibly reaching configurations of size $2^{f(n)}$