Automata, Computability, and Complexity

Hamza Bouhelal, Mohamed Reda Arsalane, Mahdi Ouchrahou

SHEET #10:

Exercice 1:

We first search for a decider of ALL_{DFA} . DFA accepts Σ^* if and only if all reachable states from the start state, q0, are accepting. We consider the tree of the computation branches of a DFA on an input string, with root q0. A decider for ALL_{DFA} is given by:

on input <M>

1) if input does not follow specifications, reject.
2) From root q0 (the initial state), check using depth first search, on the computation tree, if a non-accepting state is reachable in one of the computation paths. If a non-accepting state is found, reject.
3) if no non-accepting state is found, accept

The algorithm carries breadth-first search on the computation tree, therefore checks if a non-accepting state is reached. Computation paths are finite so it is obvious that we have a decider

We now analyse the running time analysis of each step:

Step 1) and 3) is obviously executed once. Step 2), in worst case, we go through all computation branches (paths) of the tree. let n be the number of paths, starting from root, in the computation tree of the DFA. As a result n steps are executed.

Hence in step 2) we execute polynomially many steps.

We conclude then that $ALL_{DFA} \in P$

Exercice 2:

Let A be the adjacency matrix of G, i.e., $a_{ij} = 1 \Longleftrightarrow (i, j) \in E$. Now, let us observe the entries of the $B = A^2$ matrix: $b_{ij} = \sum_{k\in[n]} a_{ik}a_{kj}$ = number of common neighbors of i and j. Therefore, $b_{ij} > 0$ if
and only if there is a path of length exactly two between i and j. So when there is an edge between i and j, and also a path of length two between i and j through some other vertex k. This gives us the following algorithm:

```
Compute B = A^2
for i = 1 to n
   for j = 1 to n
       if bij > 0 and aij > 0
            Output "Triangle Found"
```

In order to analyze the running time of this algorithm, firstly, we need to know the running time of the matrix multiplication. The current best algorithms have a running time of $O(n^{2.373})$. Therefore, the running time of the previous algorithm is $O(n^{2.373} + n^2) = O(n^{2.373})$
$O(n^{2.373}) \subseteq O(n^4)$.

Exercice 3:

Show using the definitions from the lecture notes that:

a) $n^2 + 3n^{3/2} \in O(n^2)$

In order to prove that $n^2 + 3n^{3/2} \in O(n^2)$, we need to show that
$n^2 + 3n^{3/2} \leq k.n^2$

We know that: $n^{3/2} = n.\mathrm{sqrt}(n) \leq n.n$ (since $\mathrm{sqrt}(n) \leq n$)

Therefore: $3n^2 \geq 3n^{3/2} \geq 3$

Then: $n^2 \leq n^2 \Rightarrow n^2 + 3n^{3/2} \leq n^2 + 3n^2$
$\Rightarrow n^2 + 3n^{3/2} \leq 4n^2$

We conclude that $n^2 + 3n^{3/2} \in O(n^2)$

b) $c \cdot n^{1-e} \in o(n)$ for all $e > 0$ and constants c.

In order to prove that $c \cdot n^{1-e} \in o(n)$, we need to show that $c \cdot n^{1-e} \leq k.n$

Since $e \geq 0 \Rightarrow 1 - e \leq 1$

$\Rightarrow n^{1-e} \leq n$

$\Rightarrow c.n^{1-e} \leq c.n$

We can conclude that $c \cdot n^{1-e} \in o(n)$

Exercice 4:

a-)

get n from user

set IsPrime = False

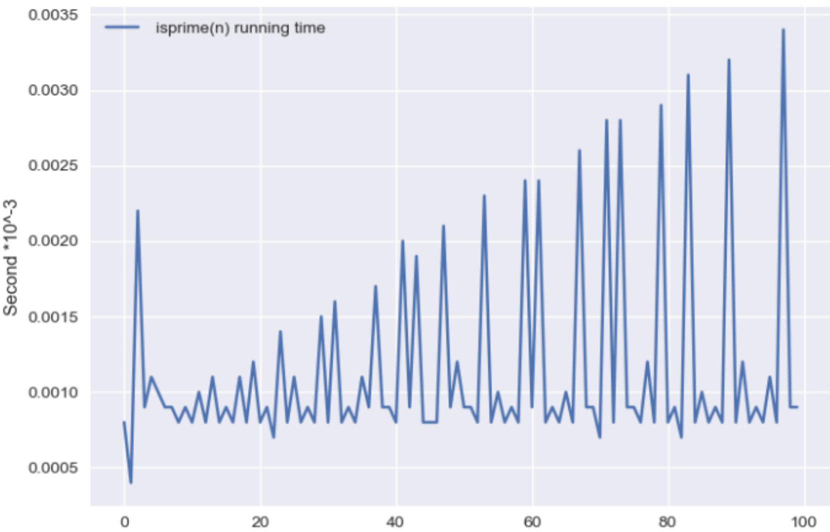
for m ranges from 2 to n-1

if Num divisible by PFactor then IsPrime = True

if IsPrime = True then display n is prime

else display n is not prime

We analyzed the running time of an implementation of our algorithm:



We conclude that it has a time complexity of O(n^k)

b-)

Suppose we have n 1's strung together (notated 1^n). n is the length of our input, obviously. We will divide all the integers from 11, 111,...,1^(n-1) into 1^n. Note that it takes log_2(x) (log base 2 of x) bits to represent x, approximately, in binary. Also note that we will be performing x-2 divisions (2, 3, 4, 5, ... , x-1 will be divided into x). So, for log_2(x) bits we use x-2 divisions. Suppose, instead, that we let n be the size of our input. So we have n = log_2(x).

Let f(x) be the number of divisions we have to do to x. f(x) = x-2 divisions, namely, 2, 3, ... , x-1. So if f(x) = x-2, and n = log_2(x), then x = 2^n. We conclude that f(n)= 2^n - 2

Thus running time is exponential, hence $L_{prime} \notin P$.