Time Complexity of
Prime Checking Algorithm

Rohan Thomas

1 Problem

Given any number $n \in \mathbb{N}$ such that $n > 1$, check if n is a **prime** or not

2 Algorithm

The intuition is to start from $k = 2 \in \mathbb{N}$ and iterate k by checking if k divides n. but before we formally state the algorithm we should answer the question, **till where should we check ?**

That is to find the **worst case scenario** of prime checking algorithm

Theorem: For any composite number $n > 2 \in \mathbf{N}$, there exist a prime divisor less than or equal to $\sqrt{n}$.

Proof: Since n is a composite number, there exists $a, b < n \in \mathbf{N}$ such that $n = ab$ suppose a,b $> \sqrt{n} \implies n = ab > n$ which is a contradiction.
$\implies$ is either a or b is less than or equal to $\sqrt{n}$

Assume WLOG, $a \leq \sqrt{n}$, by fundamental theorem of arithmetic, there exist a prime $p \leq a$ such that $p \mid a$. but $n = ab \implies a \mid n$. by transitivity of divisibility relation $\implies p \mid n$ $\square$

Notice, we now we can formally state the algorithm,

Algorithm 1 Checking if a number is prime

$n, flag \leftarrow 0$
for $i = 2$ to $\sqrt{n}$ **do**
  if $n \mod i = 0$ **then**
    $flag \leftarrow 1$
  end if
end for
if $flag = 1$ **then**
  not a prime
else
  prime
end if

3 Graphing the algorithm

We can plot the algorithm in number of input n versus number of iterations. notice the graph is bounded by the function $\sqrt{n}$