

Homework 1

The text of the questions to this homework can be found [here](#).

1. The functions can be ranked as follows:

$$O(1) < O(\lg n) = O(k \lg n) < O(n) = O(2n) = O(kn) = O(100000n) \\ < O(n \lg n) < O(n^2) < O(n^{100000}) < O(n!)$$

- 2.

This statement is true, but not very useful and hence meaningless. Big O notation gives us the upper bound of a given function, but it doesn't tell us the actual growth of a given function. Saying that the running time of algorithm A is at least as big as an upper bound is not very helpful. For example, let $f(n) = 0$, then $f(n) = O(n^2)$. Since the running time of algorithms is always non-negative, then $T(n)$ grows at least as fast as $f(n)$, which is true for every function. Thus, this statement has told us nothing of value about algorithm A , and is therefore meaningless.

3. The first one is true, but the second one is false.

Proof of 1. Let $c = 2$ and $n_0 = 1$, then

$$0 \leq 2^{n+1} = 2 \cdot 2^n \leq 2 \cdot 2^n = c \cdot 2^n$$

for all $n \geq n_0 = 1$. So the definition of big O is satisfied. \square

Proof of 2. Assume that $2^{2n} = O(2^n)$. Then there exists constants c and n_0 such that

$$0 \leq 2^{2n} = 2^n \cdot 2^n \leq c \cdot 2^n$$

for all $n \geq n_0$. Thus, $c \geq 2^n$, for an arbitrarily large value of n , a contradiction since c is a constant. Thus, $2^{2n} \neq O(2^n)$. \square

4.

One way we can compare these functions is by plotting them. We use the Python package `matplotlib` (example code [here](#)) to plot the functions as in 1. Since it is easy for the plots to become cluttered, we compare two or three functions at a time.

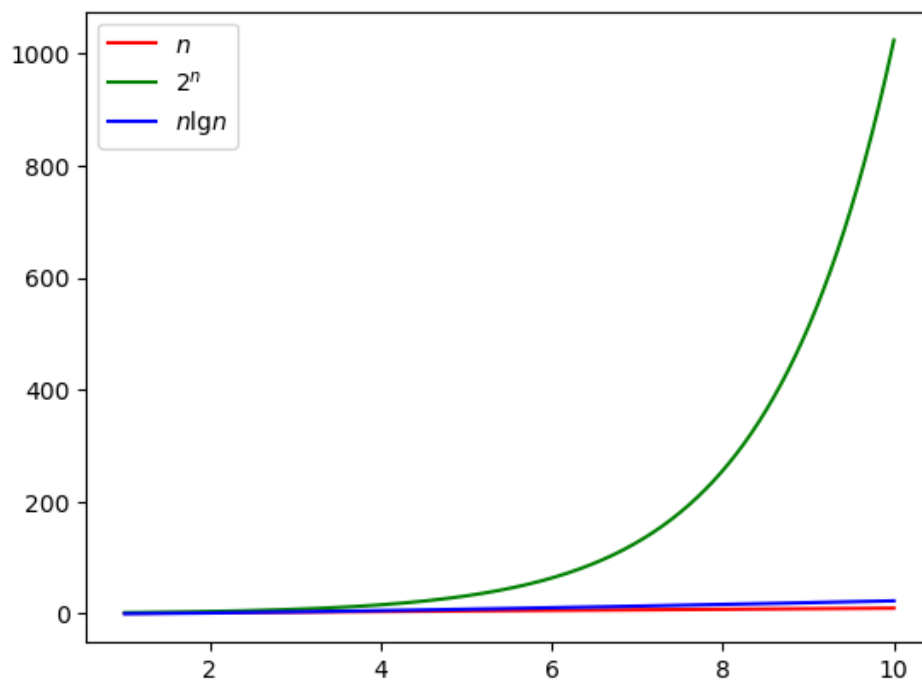


Figure 1: Comparing functions

Continuing on in this manner, we get the following list ordered fastest growing to slowest:

1. $2^{2^{n+1}}$
2. 2^{2^n}
3. $(n+1)!$
4. $n!$
5. e^n
6. $n \cdot 2^n$

- 7. 2^n
- 8. $n^{\lg \lg n}$
- 9. $(\lg n)!$
- 10. $4^{\lg n} = n^2$
- 11. $\lg(n!) = n \lg n$
- 12. $(\sqrt{2})^{\lg n}$
- 13. n
- 14. $\lg^2 n$
- 15. $\ln n$
- 16. $\sqrt{\lg n}$
- 17. $n^{1/\lg n} = 1$