Asymptotics 2 of 2!

Defn $T(n)$ is $o(g(n))$ if it is $O(g(n))$ but not $\Theta(g(n))$.

OR if $\lim_{n \to \infty} \frac{T(n)}{g(n)} = 0$.

Largest input we could process in 1 second (10^8 ops/s).

∞

Constant time: $\Theta(1)$.

ie does not depend on n.

- Array access.
- Arithmetic
- Access first element of a list
- Push/pop from a stack
- Create a variable
- Hash table operations (set/dictionaries in Python, HashSet, HashMap in Java)

Logarithmic time: $\Theta(\lg n)$ (unimaginably large)

- binary search
- height of balanced binary tree w/ n nodes
- # of bits needed to store n.

Note: we write lg as abbreviation for $\log_2$.

Recall $\log_a n = \frac{\log_b n}{\log_b a} = \frac{1}{\log_b a} \cdot \log_b n$.

Hence, the base of the log does not matter for big-O, big-Theta, etc.

Thm. $\log n$ is $o(n^x)$ for all $x > 0$.

Proof: $\lim_{n \to \infty} \frac{\ln n}{n^x} = \lim_{n \to \infty} \frac{1/n}{x n^{x-1}} = \lim_{n \to \infty} \frac{1}{x n^x} = 0$.

$\Theta(\sqrt{n})$

up to 10^{16}

Linear time : $\Theta(n)$ 10^8.

- Linear search
- Sum/average/maximum etc. of a list.
- Inserting @ start of array/list.
- Appending to end of a string.

Linearithmic time: $\Theta(n \lg n)$ ~ 4.5 million.

- Mergesort, quicksort.

Quadratic time : $\Theta(n^2)$ ~ 10 thousand.

- Insertion sort, bubble sort.
- Nested loops (often)
- $1+2+3+4+\cdots+n = \frac{n(n+1)}{2}$ is $\Theta(n^2)$.
- # of pairs out of n things $\binom{n}{2}$

Polynomial time: $\Theta(n^k)$

- k nested loops.
- # of subsets of size k $\binom{n}{k}$

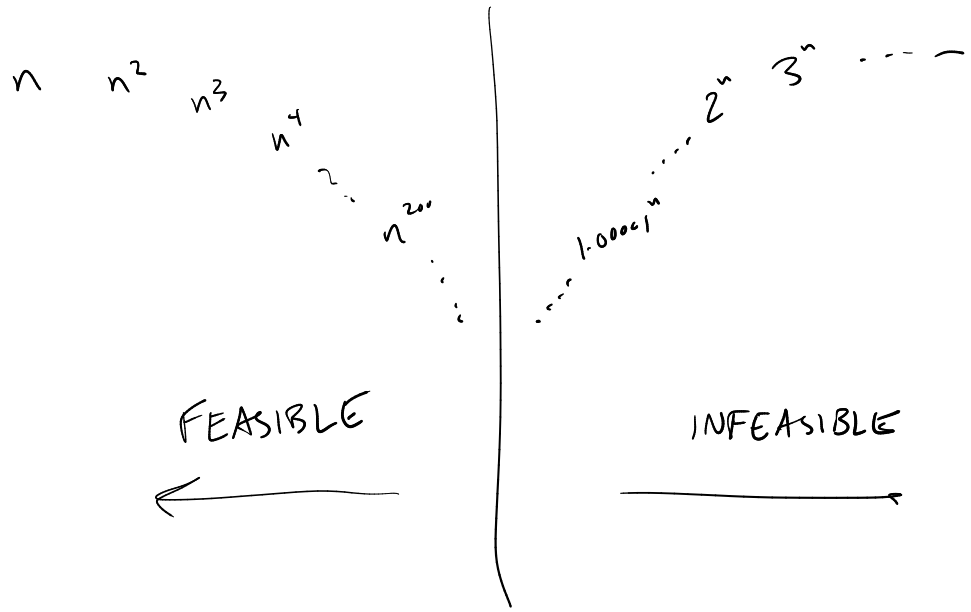
Each n^j is $o(n^k)$ when $j<k$.

Exponential time: $\Theta(2^n)$ 26.

# of subsets of n.
# of bitstrings of length n.

$1+2+4+8+\cdots+2^n = 2^{n+1}-1$ is $\Theta(2^n)$.

Theorem: n^k is $o(r^n)$ for all $k \geq 0$ and all $r > 1$.


n
n^2
n^3
n^4
n^{20}
2^n
3^n
1.0001^n
FEASIBLE
INFEASIBLE