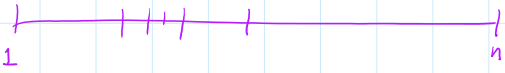
COMPUTATIONAL COMPLEXITY

Suppose we have an algorithm that operates on input of size n. We say the algorithm is $O(f(n))$ ("big-O of $f(n)$") if the runtime is not greater than $c \cdot f(n)$ for some constant c (and large n).

EXAMPLES:

$O(1)$ constant time e.g. adding two numbers

$O(\log n)$ log time e.g. binary search — searching for item in a sorted list



$O(n)$ linear time e.g. given list of size n

```
for i from 1 to n:
    Print list[i]
```

$O(n \log n)$ ← fast sorting algorithms (merge sort, quicksort)

$O(n^2)$ ← slow sorting algorithms (bubble sort, insertion sort)

```
for i from 1 to n:
    for j from 1 to i:
```

INCREMENTAL ALGORITHM FOR CONVEX HULLS

INPUT: set S of n points in the plane, given by coordinates

OUTPUT: list L containing vertices of $\text{conv}(S)$, in counterclockwise order

PSEUDOCODE:

1. Sort S by x-coordinate
2. Take first 3 points. Let H_3 be these points in counterclockwise order
3. For $k = 4$ to n:
   - Insert point k into H_{k-1} and remove all points now in the interior, forming H_k
4. Set L equal to H_n

Can we give detailed instructions on how to do this?