# Discussion 6: Testing & Algorithmic Complexity

## Testing


1. We try to test our code, but we get an error. What does it mean and how can we fix it?


DO NOT WORRY ABOUT THIS QUESTION. IT IS OUTDATED; THE TESTING BLOCK HAS CHANGED.

## Algorithmic Complexity: Definitions

1. What is runtime? How do we measure it?

Runtime is a measure of the amount of time a procedure takes to execute. However, since timing computer programs using sub-seconds is impractical, we instead measure runtime as the number of steps a procedure takes to execute, as a function of the input size.

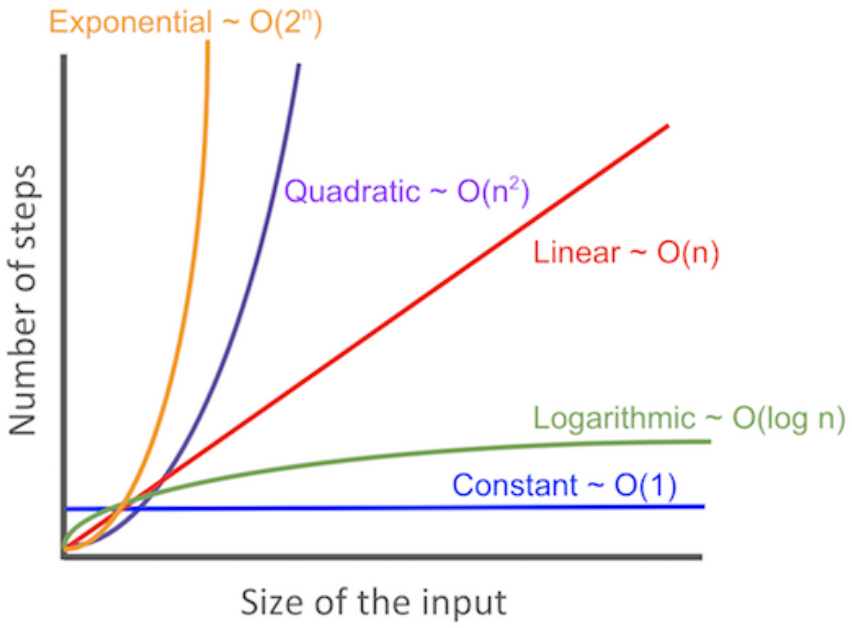
2. If a function runs in O(n) time, that means it runs…
   O in linear time at worst  O in linear time on average  O in linear time at best

## Understanding Runtimes

1. Fill in the following chart:

| **Runtime** | **Notation** | **As input size increases by…** | **The number of steps change by…** |
|---|---|---|---|
| Constant | O(1) | x2 | +0 |
| Logarithmic (base 2) | O(logn) | x2 | +1 |
| Linear | O(n) | x2 | x2 |
| Quadratic | $O(n^2)$ | x2 | x4 |
| Exponential (base 2) | $O(2^n)$ | +1 | x2 |

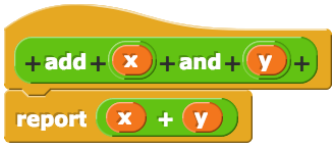
2. In the following diagram, which is the best runtime? The worst?



## Runtime Practice

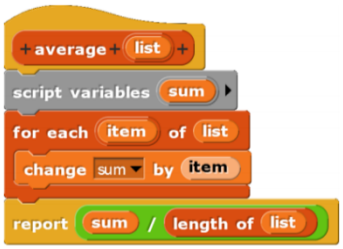
1. Find the runtime of each of the following blocks or processes.

a.



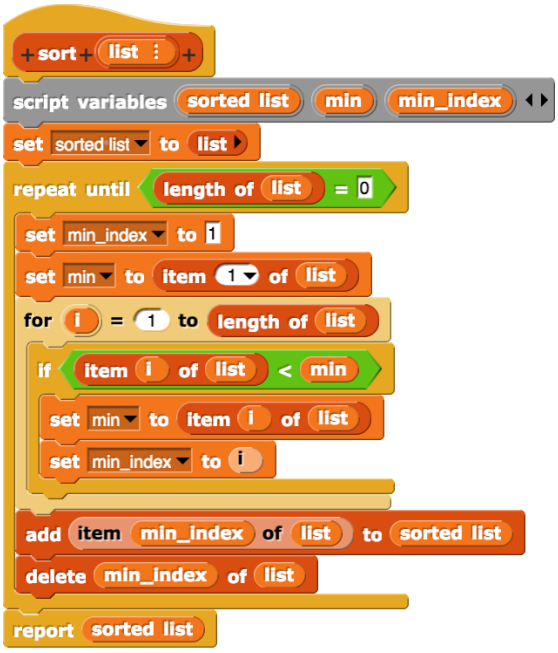

Constant

b.




Linear

c.




Quadratic

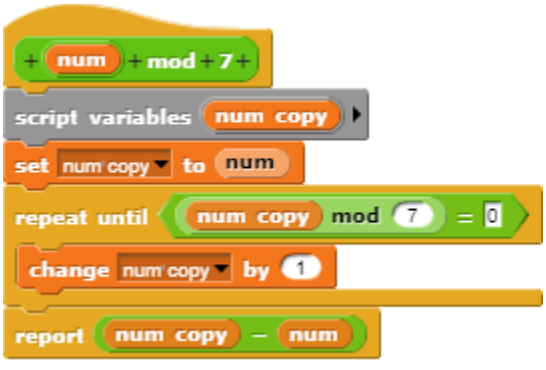
d. This process takes in a value and a list and searches through every item in the list one by one to see if it can find that value.

Linear

e. This process takes in a value and a sorted list and searches for the value in the sorted list. Every iteration of the algorithm, it figures out which half of the list the value would be in, and then only searches in that half of the list.

Logarithmic

f.



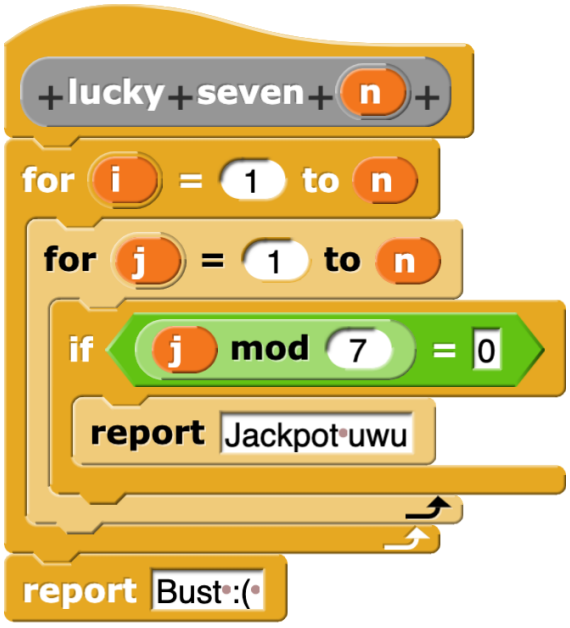

g. You know a secret, and you want to share it with the world. In *state 0*, you are the only person who knows the secret. Then in *state 1*, you share the secret with two friends, so three total people know the secret. Then in *state 2*, both of your friends tell two of their friends, so seven total people know the secret. This pattern (of people sharing the

secret with two friends) continues indefinitely. As a function of the *state*, what is the order of growth of the number of people who know the secret?

Constant

Exponential

## More Runtime Practice




**What is the runtime of this block when n is less than 7?**

**Constant**

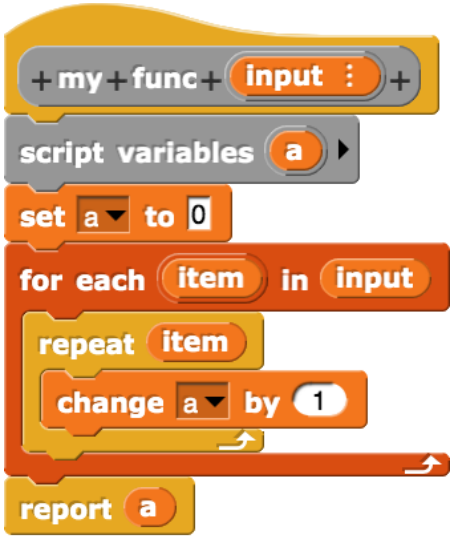
**What is the runtime of the block when n is greater than 7?**

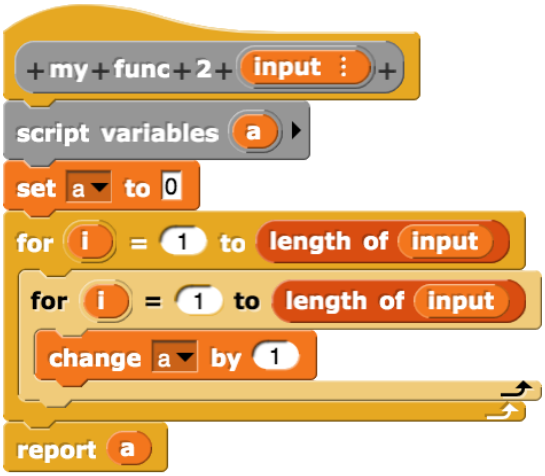
**Constant**

**Why?**

**Generally, we ask runtime questions in a theoretical context. However, here, we are given the input size *n* ahead of time, and we know it is a constant number. There is no way to consider what happens to the runtime as *n* goes to infinity, because we are already**

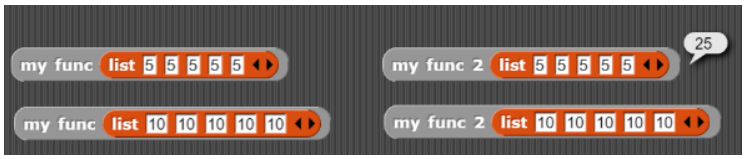
told it is upper bounded by 6, which is a constant. Thus, the runtime is also constant when *n* is less than 7.




**What do the following calls report? The first one is done for you.**







## Challenge Problems

1. What does the following expression do? Assuming that all helper (non-HOF) blocks operate in constant time, what is its runtime?


This block reports whether all items in the input list are even (evenly divisible by 2). It reports True if all items are even, and False otherwise. Map and combine both execute in linear time, and here operate sequentially (map runs first, then, after finishing, combine runs using the output of map as its input list). So, assuming all helper blocks take constant time, the overall runtime here is O(n+n): n for map and n for keep, assuming that the input list contains n items. This simplifies to O(2n), but since we ignore linear factors in big O notation, we conclude that the block's runtime is O(n).

2. Assume that the word → list block executes in linear time as a function of the length of the input word. If myList is a list of n words, each of length n, what is the runtime of the following expression?


Assuming myList contains n items, map will execute the word → list function n times—once for each item. How many steps does the word → list function take? Well, since it executes in linear time and we're assuming that every item in myList is a word of length n, word → list will take n steps to run on each individual word. The map is calling a function that takes n steps a total of n times. Thus, the overall runtime here is O(n*n), or $O(n^2)$.