Big O Notation

Linear time:

- O(n)
- As n increases time to compute increases at same rate
- Ex. Looping through an array once (for loop)

Constant time:

- O(1)
- As n increases time to compute stays the same

Quadratic time:

- O(n^2)
- As n increases time increases much more
- Ex. Two for loops, one nested in another

Logarithmic time:

- O(log(n))
- As n increases the smaller proportion of actual input the program has to go through so time is reduced compared to O(n).

Different steps get added:

- Take the complexity for each step in the algorithm and add them together.
- Ex. Two steps, first = O(n), second = O(a), overall complexity is O(n + a)

Drop constants:

- O(2n) = O(n)
- Interested in linear vs quadratic, constants are dropped.

Different inputs leads to different variables:

- Ex. Two arrays are input to algorithm (a and b).
- First array is looped through, and second array is looped through inside first loop (nested).
- O(a^2) is wrong.
- O(a*b) is right.

Drop non-dominant terms:

- Ex. Two steps in algorithm
- First is O(n)
- Second is O(n^2)
- Overall complexity is O(n + n^2) but n is linear and therefore non-dominant so is dropped leading to complexity of O(n^2).
-