Programming fundamentals with Python

Session 10 - Search algorithms

Pepe García jgarciah@faculty.ie.edu

2020-04-20

Asymptotic notation refresher

- Big Theta (Θ) - Only one case in term of runtime (worst case $==$ best case)

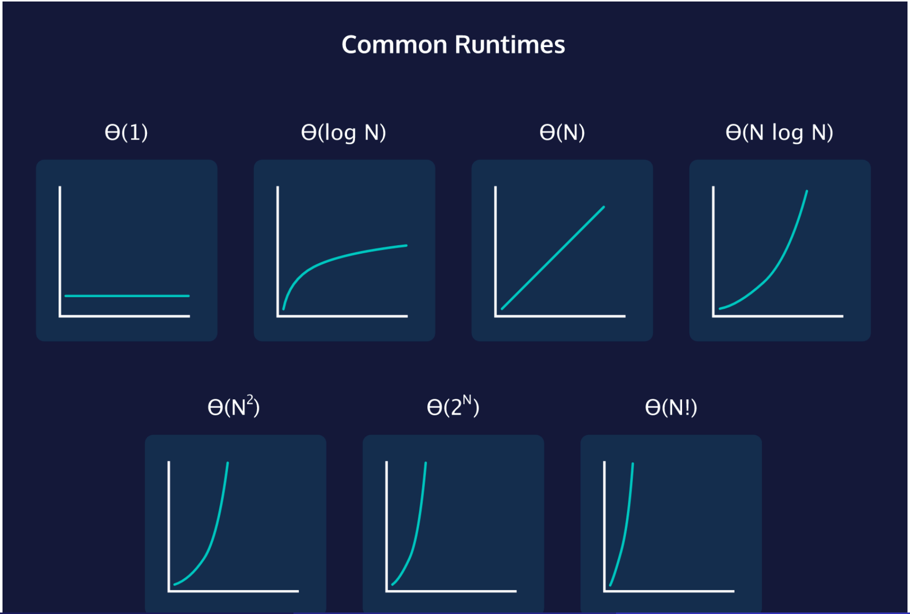
Asymptotic notation refresher

- Big Theta (Θ) - Only one case in term of runtime (worst case $==$ best case)
- Big Omega (Ω) - For describing **best case** runtime

Asymptotic notation refresher

- Big Theta (Θ) - Only one case in term of runtime (worst case $==$ best case)
- Big Omega (Ω) - For describing **best case** runtime
- Big O (O) - For describing **worst case** runtime. **This is the one we'll use most of the time**.

Different runtimes



Search algorithms

Search algorithms are used to check the existence of an element in a sequence. There are different things that may affect how to search in that sequence.

Search algorithms

Our search algorithms will be implemented as functions in Python that receive **two parameters, the element we're searching for, and the list**, and will **return a boolean** representing if the element exists in the sequence.

Types of search algorithms

We will learn about the two main search algorithms used. **Linear search** and **Binary search**.

Linear search

We will use linear search whenever we're not sure if the list is sorted or not. For implementing linear search we will:

- Iterate over all elements in the list

Linear search

We will use linear search whenever we're not sure if the list is sorted or not. For implementing linear search we will:

- Iterate over all elements in the list
- if we find the element, **return True**.

Linear search

We will use linear search whenever we're not sure if the list is sorted or not. For implementing linear search we will:

- Iterate over all elements in the list
- if we find the element, **return True**.
- if we don't find the element, **return False**.

Linear search

We will use linear search whenever we're not sure if the list is sorted or not. For implementing linear search we will:

- Iterate over all elements in the list
- if we find the element, **return True**.
- if we don't find the element, **return False**.

Linear search

We will use linear search whenever we're not sure if the list is sorted or not. For implementing linear search we will:

- Iterate over all elements in the list
- if we find the element, **return True**.
- if we don't find the element, **return False**.

Linear search

Let's implement linear search ourselves!

Linear search - Discussion

- What's the worst case runtime of linear search? (**Big O**)

Linear search - Discussion

- What's the worst case runtime of linear search? (**Big O**)
- And the best case? (**Big Omega**)

Linear search - Discussion

- What's the worst case runtime of linear search? (**Big O**)
- And the best case? (**Big Omega**)
- Can you say something good or bad about linear search?

Linear search - Discussion

- What's the worst case runtime of linear search? (**Big O**)
- And the best case? (**Big Omega**)
- Can you say something good or bad about linear search?

Linear search - Discussion

- What's the worst case runtime of linear search? (**Big O**)
- And the best case? (**Big Omega**)
- Can you say something good or bad about linear search?

The good thing about linear search is that it will work for any sequence, regardless if it's sorted.

Something bad, it may be inefficient in some cases.

Binary search

Binary search is the algorithm we'll apply to search for an element in a **sorted** sequence. A sequence being sorted implies that elements inside it are greater than or equal to previous elements and lesser than or equal to following elements.

`[]` is sorted

`[1]` is sorted

`[1,1,1,1]` is sorted

`[1,2,3,3,3,3,4,7,9,12,31]` is sorted

`[2,1,3,3,3,3,4,7,9,12,31]` is **not** sorted

Binary search

Imagine we need to go through a dictionary (the actual book, not a Python dictionary) to search for a word. What's the algorithm you automatically apply to the search?

Binary search

Imagine we need to go through a dictionary (the actual book, not a Python dictionary) to search for a word. What's the algorithm you automatically apply to the search?

- Open the dictionary in a page in the middle.

Binary search

Imagine we need to go through a dictionary (the actual book, not a Python dictionary) to search for a word. What's the algorithm you automatically apply to the search?

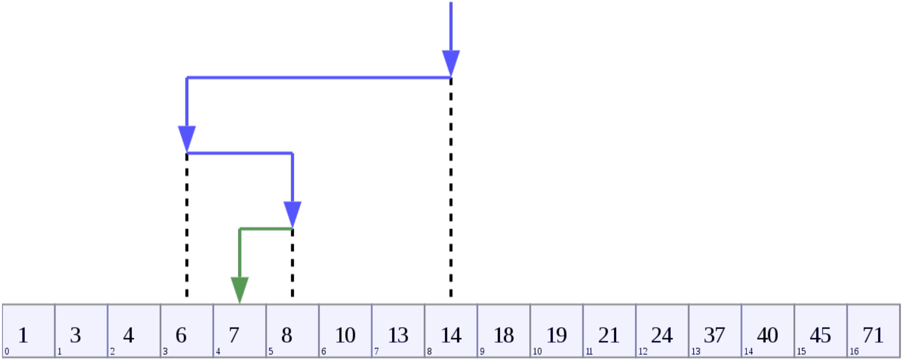
- Open the dictionary in a page in the middle.
- If the word is *lesser than* the current words, jump to the middle of the left half.

Binary search

Imagine we need to go through a dictionary (the actual book, not a Python dictionary) to search for a word. What's the algorithm you automatically apply to the search?

- Open the dictionary in a page in the middle.
- If the word is *lesser than* the current words, jump to the middle of the left half.
- If it's *greater than* the current word, jump to the middle of the right half.

Binary search



1	3	4	6	7	8	10	13	14	18	19	21	24	37	40	45	71
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Binary search

Binary search

Let's implement binary search ourselves!

Binary search - discussion

- What's the worst case runtime of it? (**Big O**)

Binary search - discussion

- What's the worst case runtime of it? (**Big O**)
- And the best case? (**Big Omega**)

Binary search - discussion

- What's the worst case runtime of it? (**Big O**)
- And the best case? (**Big Omega**)
- Can you say something good or bad about it?

Binary search - discussion

- What's the worst case runtime of it? (**Big O**)
- And the best case? (**Big Omega**)
- Can you say something good or bad about it?

Binary search - discussion

- What's the worst case runtime of it? (**Big O**)
- And the best case? (**Big Omega**)
- Can you say something good or bad about it?

The good thing about binary search is that it's more performant than linear search.

Something bad, it doesn't work for all sequences, they must be sorted.