2025

4Geeks Academy: data science cohort 12

DAY 14: ALGORITHM OPTIMIZATION

TODO

ALGORITHM OPTIMIZATION

Time and space complexity considerations for writing data science code in Python

ALGORITHM OPTIMIZATION PROJECT

Work on Algorithm Optimization Project in Python (Algorithm optimization module), plan to finish over the weekend

HYPOTHESIS TESTING PROJECT

Submit Hypothesis Testing Exercises Project in Python (Hypothesis testing module), if you haven’t done so already

TOPICS

01 ALGORITHMS

02 TIME & SPACE COMPLEXITY

03 TIPS

ALGORITHMS

ALGORITHM

- Specific steps to accomplish some goal
- Often more than one possible algorithm for a goal

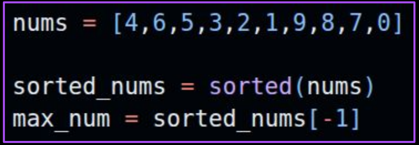
```
1. Set the first number as the
   maximum
2. For each number in the list
   a. If the number is greater
      than the maximum, set it
      as the maximum
```

VS

```
1. Sort the list from smallest
   to largest
2. Take the last number in the
   sorted list
```

IMPLEMENTATION

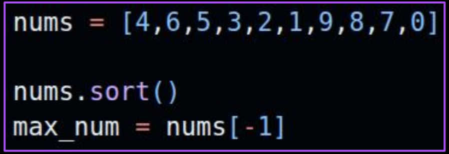
- Code that does the steps of an algorithm
- Can be more than one implement for a given algorithm



```
nums = [4,6,5,3,2,1,9,8,7,0]

sorted_nums = sorted(nums)
max_num = sorted_nums[-1]
```

VS



```
nums = [4,6,5,3,2,1,9,8,7,0]

nums.sort()
max_num = nums[-1]
```

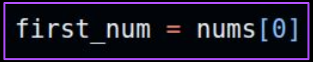
ALGORITHMS

TIME COMPLEXITY

How does an algorithm’s run time scale as the input size increases?

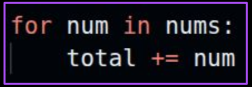
SPACE COMPLEXITY

How does an algorithm’s memory use scale as the input size increases?



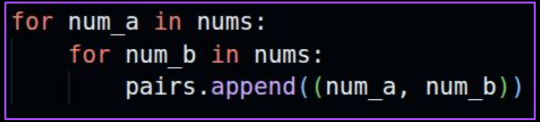
```python
first_num = nums[0]
```

$O(1)$ time complexity: doesn't matter how long list is, ‘constant time’



```python
for num in nums:
    total += num
```

$O(n)$ time complexity: as list length increases, run time increases by same factor, ‘linear time’



```python
for num_a in nums:
    for num_b in nums:
        pairs.append((num_a, num_b))
```

$O(n^2)$ time complexity: as list length increases, run time grows by the square, ‘quadratic time’

TIPS

USE BUILTIN FUNCTIONS

If Python, Numpy, Pandas etc., already have a function, use it (ex: don’t ‘hard-code’ your own sort!)

DON’T LOOP

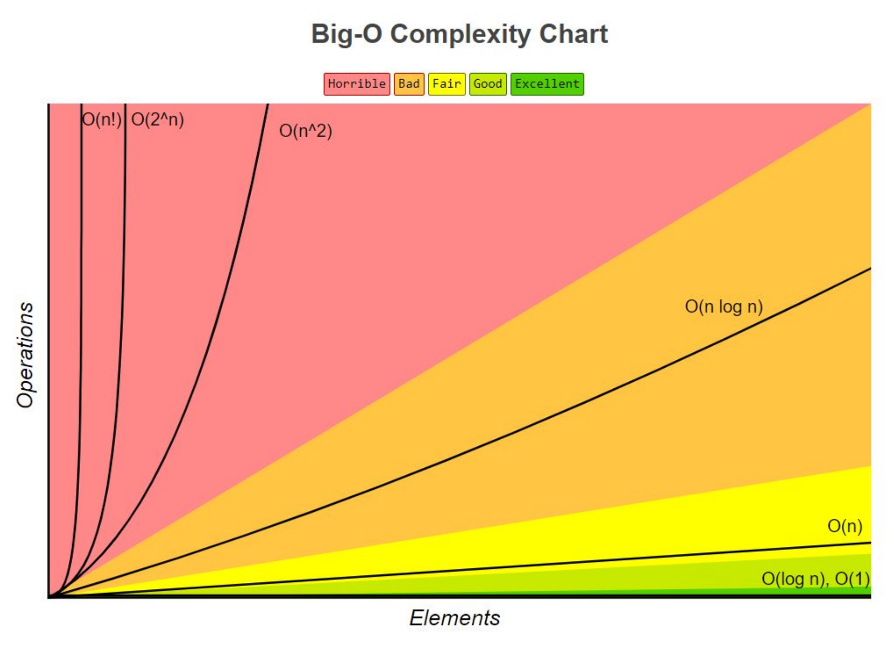
Use vectorized operations & generators where possible (ex: Pandas apply)

MUTATE

Don’t create extra objects or copies, especially of large datasets

CONSIDER CONTEXT

”Bad’ implementations can be fine if inputs are small and/or run infrequently - sometimes clearly written loops make code easier to understand


Big-O Complexity Chart
Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements