Programming

Lesson20 – Complexity
الدرس 20 – التعقيد
الوقت والمكان

Saeed Isa

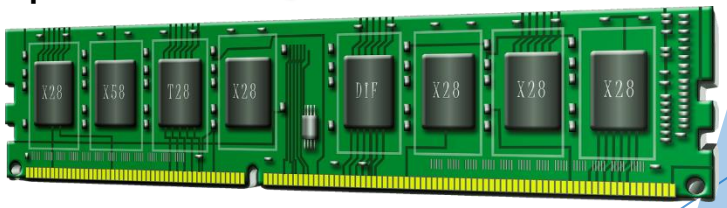
Complexity

- Function to describe and measure algorithm/function/program complexity
  - How much efficient is your algorithm ?

- Ever program consumes:

Time

Space



Time complexity

- Assume “print(<>)” takes 1ms
- Total time?
  - ➔ ~1ms
- If lst is 1000 ➔ 1ms
- If lst is n ➔ 1ms

- If lst definition takes 2ms
  - ➔ 1 ms + 2 ms = 3ms

```
lst = [3, 2, 64, 24, 42, 77, 39, 50, 82, 4]
print(lst[0])
```

Time complexity cont.

- Assume “print(i)” takes 1ms
- Total time ?
  - ➔ ~10ms
- If lst is 1000 item ➔ ~1000ms
- If lst is n items ➔ n ms

- If lst definition takes 2ms
  - ➔ n ms + 2 ms

```
lst = [3, 2, 64, 24, 42, 77, 39, 50, 82, 4]
for i in lst:
    print(i)
```

Time complexity cont.

```
lst = [3, 2, 64, 24, 42, 77, 39, 50, 82, 4]
for i in lst:
    for j in lst:
        print(i*j)
```

- Assume “print(i*j)” takes 1ms
- Total time:
  - ➔ 10 * 10 ms ➔ 100ms
- If lst is 100 ➔ 100 * 100 ➔ 10000ms
- If lst is n ➔ n * n ms ➔ n^2 ms

- If lst definition takes 2 ms and multiplication 2 ms
  - ➔ $(4n^2 + 2)$ms

Time complexity cont.

- Total time:
  - ➔ n^3
- ➔ $27n^3$ + 2ms

```
lst = [3, 2, 64, 24, 42, 77, 39, 50, 82, 4]
for i in lst:
    for j in lst:
        for s in lst:
            print(i*j*s)
```

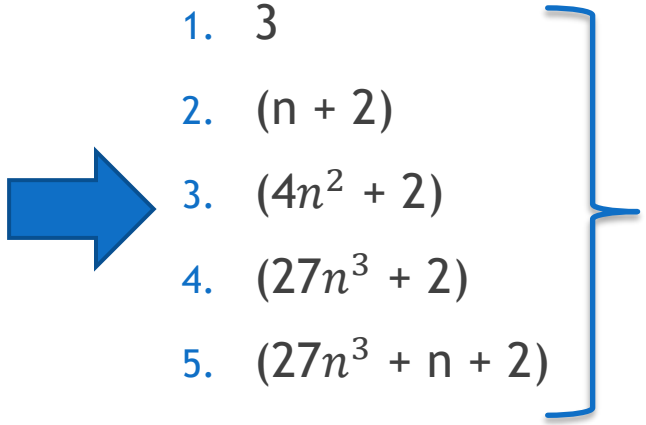
- ➔ $27n^3$ + n + 2ms

```
lst = [3, 2, 64, 24, 42, 77, 39, 50, 82, 4]
for i in lst:
    for j in lst:
        for s in lst:
            print(i*j*s)

for i in lst:
    print(i)
```

Time complexity cont.

1. 3ms
2. (n + 2)ms
3. $(4n^2 + 2)$ms
4. $(27n^3 + 2)$ms
5. $(27n^3 + n + 2)$ms



1. 3
2. (n + 2)
3. $(4n^2 + 2)$
4. $(27n^3 + 2)$
5. $(27n^3 + n + 2)$

1. O(1)
2. O(n)
3. $O(n^2)$
4. $O(n^3)$
5. $O(n^3)$

Time complexity conclusion

- The time complexity of an algorithm is the amount of time taken by the algorithm to complete its process as a function of its input length, n. (educative.io)
- Complexity expressed by: ***asymptotic*** notations *O(n)*

➔ $O(1) < O(\log n) < O(n) < O(n * \log n) < O(n^2) < O(n^3) < O(2^n) < O(n^n)$

➔ Calculate how much operations relatively to input length

Space complexity

- How much memory does the algo consume ?
- the amount of space (or memory) taken by the algorithm to run as a function of its input length, n.

- If list length is n ➔ space complexity O(n)

```
lst = [3, 2, 64, 24, 42, 77, 39, 50, 82, 4]
for i in lst:
    for j in lst:
        print(i*j)
```

Space complexity cont.

- Total space ?

  ➔ $O(n^2)$

To be specific:
➔ Total space ➔ n + n^2
➔ ***asymptotic*** notations ➔ $O(n^2)$

```
lst = [3, 2, 64, 24, 42, 77, 39, 50, 82, 4]
lst2 = []
for i in lst:
    for j in lst:
        lst2.append(i*j)
```

Binary search example

- Searching num in sorted numbers list

lst = [2, 12, 29, 100, 432, …, 1050, 5002, 6412, 1000]

- n numbers
  - n/2 numbers
    - n/4 numbers
      - n/8 numbers
        - …

$\log_2 n$

```
def binary_search(data, value):
    n = len(data)
    left = 0
    right = n - 1
    while left <= right:
        middle = (left + right) // 2
        if value < data[middle]:
            right = middle - 1
        elif value > data[middle]:
            left = middle + 1
        else:
            return middle
    raise ValueError('Value is not in the list')
```

Thank you ☺!

Stay tuned for more!