Data Structures & Algos

Part - 1

Let’s impose some new constraints

1. Before asking a question, explain your opinion first and then ask where your thought process might have deviated from the answer.
2. When you write code from now on, always think about the performance implications of what you’re writing while writing it.
3. If code I’m writing or something I’m saying seems off to you exercise challenging it.
4. Practice what you’ve learned in your own time on top of our set work or it’ll disappear. Even 10 mins a day, slowly working your way up is a great starting point.
5. If you self impose these constraints and find something that works for you, you’ll be successful in your career.

Complexity analysis recap

Let’s remind ourselves…

1. What’s time complexity?

2. What’s space complexity?

3. What’s complexity analysis for in general?

Big O Notation

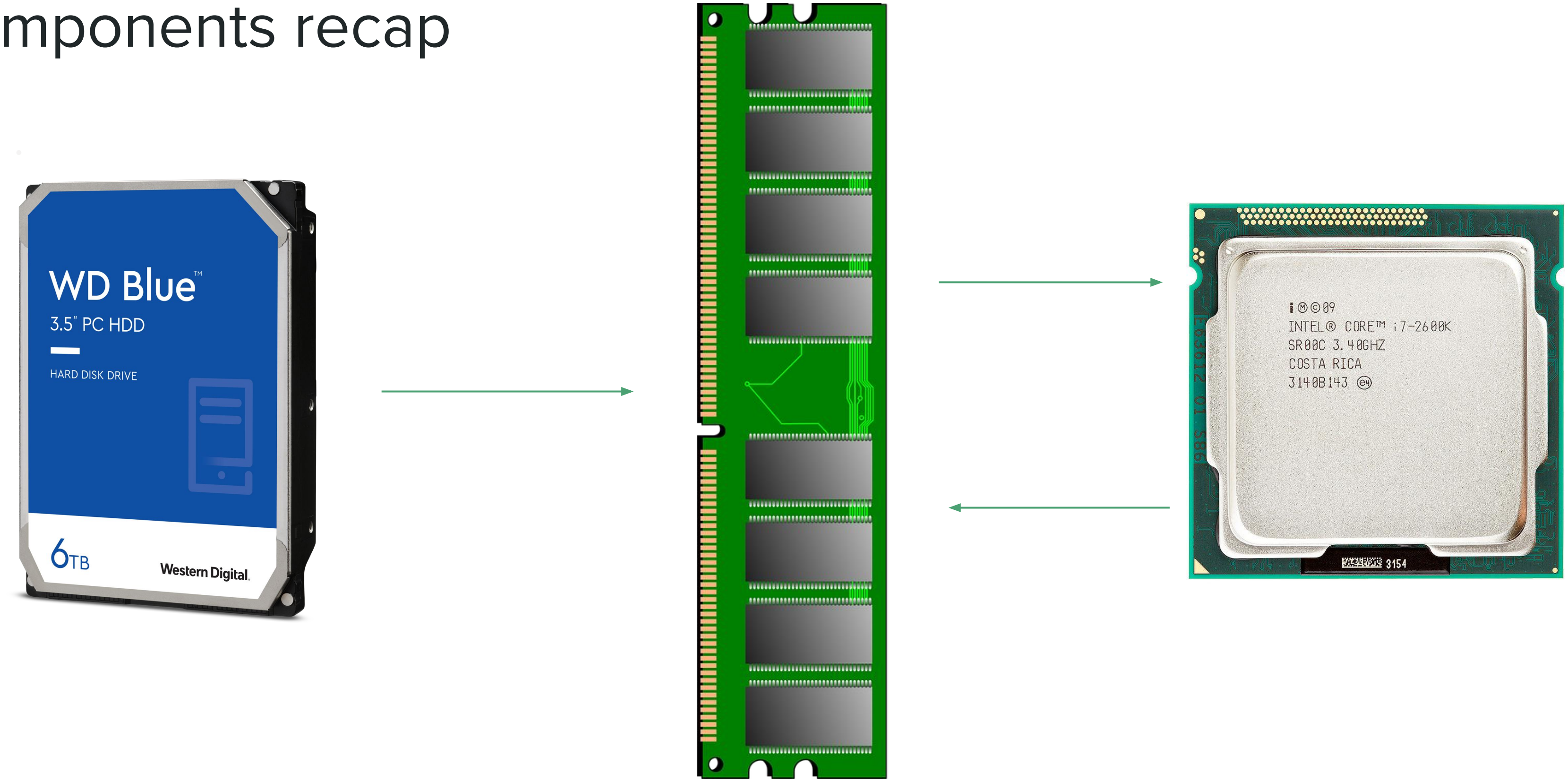
Why do we use BigO and not measure efficiency in seconds?

```
func doSomething() {
    time.sleep(5); // Sleeps for 5 seconds
}

func printEverything(arrayOfEverything) {
    for (eachObject in arrayOfEverything) {
        print(eachObject);
    }
}
```

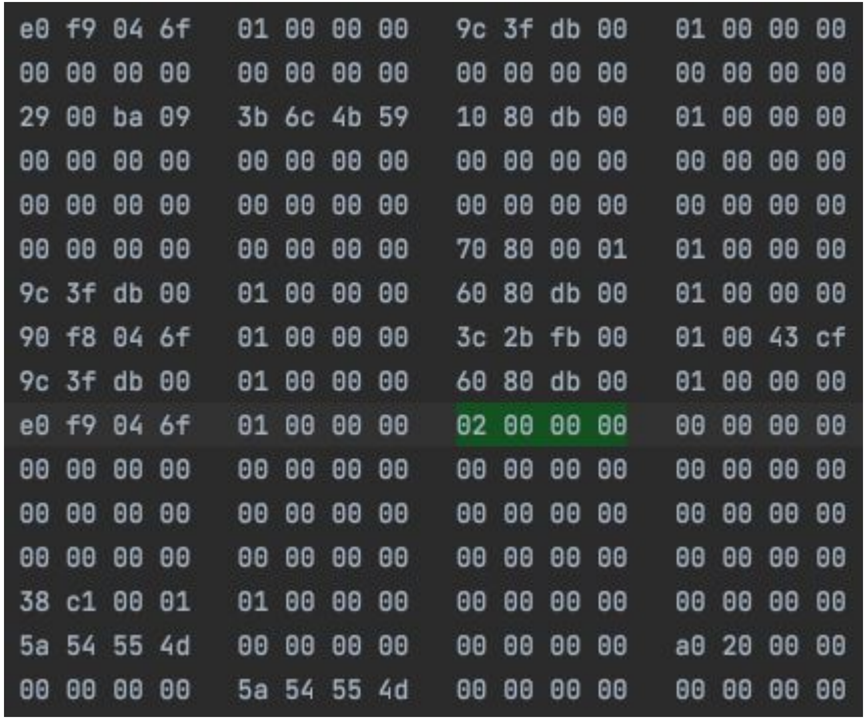
Big O Notation

```
Int[] arr = initialiseArray(N = 10000);

func constantTimeFunction(int[] arr) {
    ...
}

func linearTimeFunction(int[] arr) {
    ...
}

func exponentialTimeFunction(int[] arr) {
    ...
}
```

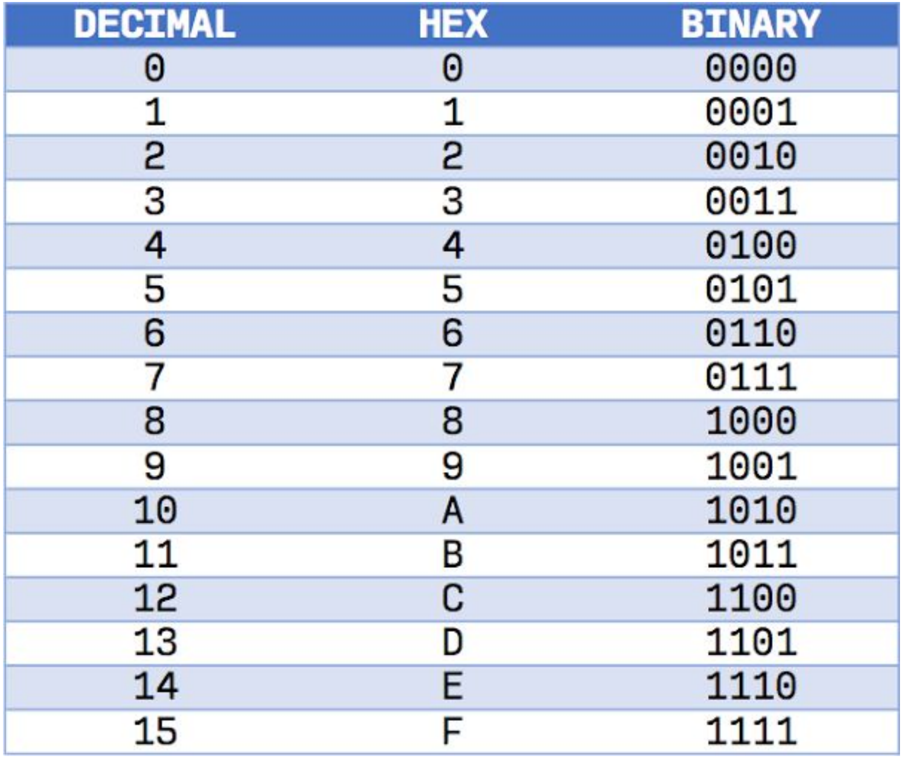
Components recap



Don’t get caught out though, let’s look under the hood



```
e0 f9 04 6f   01 00 00 00   9c 3f db 00   01 00 00 00
00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
29 00 ba 09   3b 6c 4b 59   10 80 db 00   01 00 00 00
00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
00 00 00 00   00 00 00 00   70 80 00 01   01 00 00 00
9c 3f db 00   01 00 00 00   60 80 db 00   01 00 00 00
90 f8 04 6f   01 00 00 00   3c 2b fb 00   01 00 43 cf
9c 3f db 00   01 00 00 00   60 80 db 00   01 00 00 00
e0 f9 04 6f   01 00 00 00   02 00 00 00   00 00 00 00
00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
38 c1 00 01   01 00 00 00   00 00 00 00   00 00 00 00
5a 54 55 4d   00 00 00 00   00 00 00 00   a0 20 00 00
00 00 00 00   5a 54 55 4d   00 00 00 00   00 00 00 00
```



DECIMAL	HEX	BINARY
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

What inside the RAM stick looks like...

Address	Variable	Hex
0x0000...4f800	Not initialised	00 00 00 00 00 00 00 00...
0x0000...4f810	int one = 1	01 00 00 00 00 00 00 00...
0x0000...4f820	int nine = 9	09 00 00 00 00 00 00 00...
0x0000...4f830	arr[0] = 2	02 00 00 00 00 00 00 00...
0x0000...4f840	arr[1] = 3	03 00 00 00 00 00 00 00...
0x0000...4f850	arr[2] = 4	04 00 00 00 00 00 00 00...
0x0000...4f860	ptr to 3rd slot: what var is this?	0x0000...4f820

Big O Notation

O(6)

O(N)

O(N^2)

O(N^3)

O(N + M)

O(9N)

Space-time complexity

Traverse an array: O(?) - time, O(?) Space

Copy an array: O(?) - time, O(?) Space

Get item at specific index of an array: O(?) - time, O(?) Space

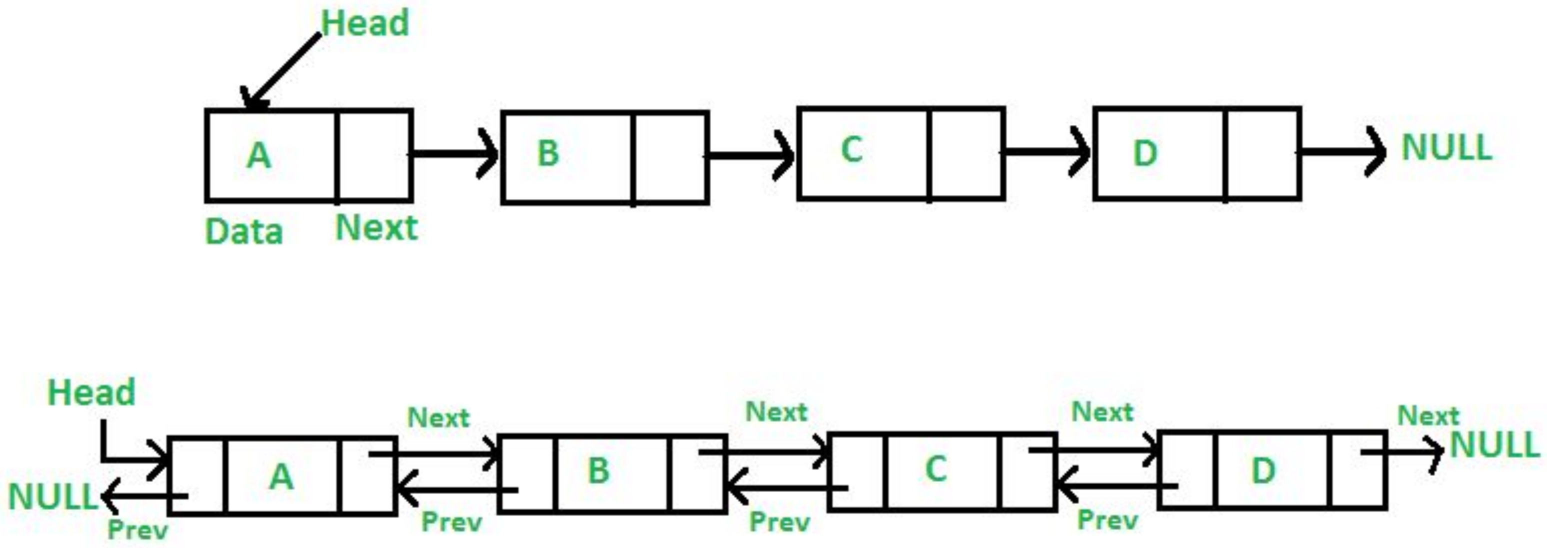
Array Recap: int[] arr = {2, 3, 4};

Address	Variable	Hex
0x0000...4f800	Not initialised	00 00 00 00 00 00 00 00...
0x0000...4f810	Not initialised	00 00 00 00 00 00 00 00...
0x0000...4f820	Not initialised	00 00 00 00 00 00 00 00...
0x0000...4f830	arr[0] = 2	02 00 00 00 00 00 00 00...
0x0000...4f840	arr[1] = 3	03 00 00 00 00 00 00 00...
0x0000...4f850	arr[2] = 4	04 00 00 00 00 00 00 00...
0x0000...4f860	Not initialised	00 00 00 00 00 00 00 00...

Linked lists: List<Double> temp1 = new LinkedList<Double>(Arrays.asList(1.0, 2.0));

Address	Variable	Hex
0x0000...4f800	temp1.node1.value = 2.0	02 00 00 ... 00 00 00 00
0x0000...4f810	Null address	01 34 00 ... 00 00 83 FF
0x0000...4f820	not initialised	00 00 00 ... 00 00 00 00
0x0000...4f830	temp1.node1.value = 1.0	01 00 00 ... 00 00 00 00
0x0000...4f840	temp1.node1.ptr = 0x..4f800	00 00 00 ... 00 04 f8 00
0x0000...4f850	not initialised	00 00 00 ... 00 00 00 00
0x0000...4f860	not initialised	00 00 00 ... 00 00 00 00

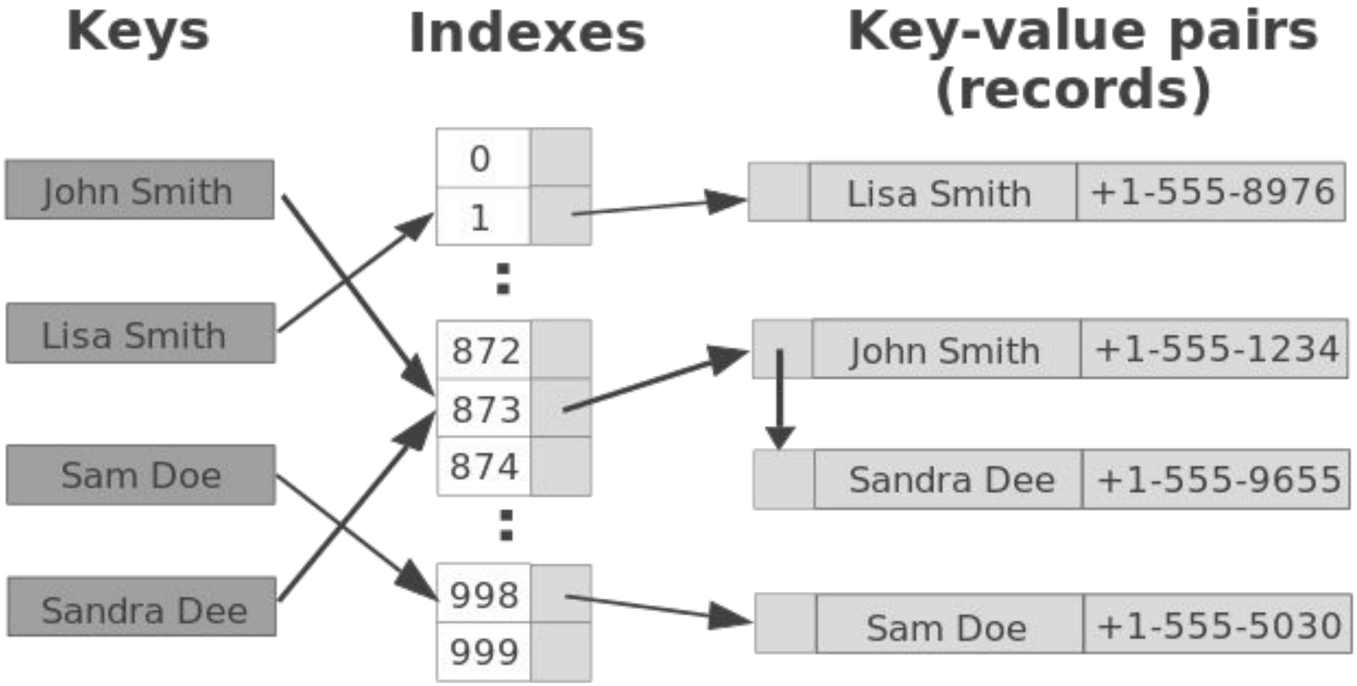
Doubly linked lists



That’s Pretty much all you need to know

On to the next quick challenge

Hash tables


Keys
Indexes
Key-value pairs
(records)
John Smith
Lisa Smith
Sam Doe
Sandra Dee
0
1
:
872
873
874
:
998
999
Lisa Smith
+1-555-8976
John Smith
+1-555-1234
Sandra Dee
+1-555-9655
Sam Doe
+1-555-5030