Data Structures

BCI CS club

Let’s Implement Multiplication

Python

```python
a = 196
b = 4372

total = 0
for i in range(a):
    total += b
```

C++

```cpp
int main(){
    int a = 196;
    int b = 4372;

    int total = 0;
    for (int i = 0; i < a; i++) {
        total += b;
    }
}
```

Why Is This Bad?

What Does “Bad” Even Mean?

takes up a lot of time

takes up a lot of memory

let’s focus on time

So What Does “A Lot Of Time” Mean

Turns out it isn’t really that useful to just use standard time measurements

- time varies with conditions

Surely there is a better way!

Scaling

- instead of describing how a program performs with a given input
    - how fast is insertion sort with a list of 1 000 000 elements
- we ask how insertion sorts' speed increases as you increase the number of elements
    - if we say the length of the list to sort is N, then insertion sort takes

      10.5 x (n ^ 2) + 13.17 imperial nanoseconds to run

Why Imperial Nanoseconds?

It’s just a convention, get used to it

Im Joking, Please Don’t Leave

Units Aren’t Useful

If you think about it, using units makes about as much sense as just comparing times

so that changes the performance of insertion sort to 10.5 (n ^ 2) + 13.17 time units

Why Do We Have These Random Numbers?

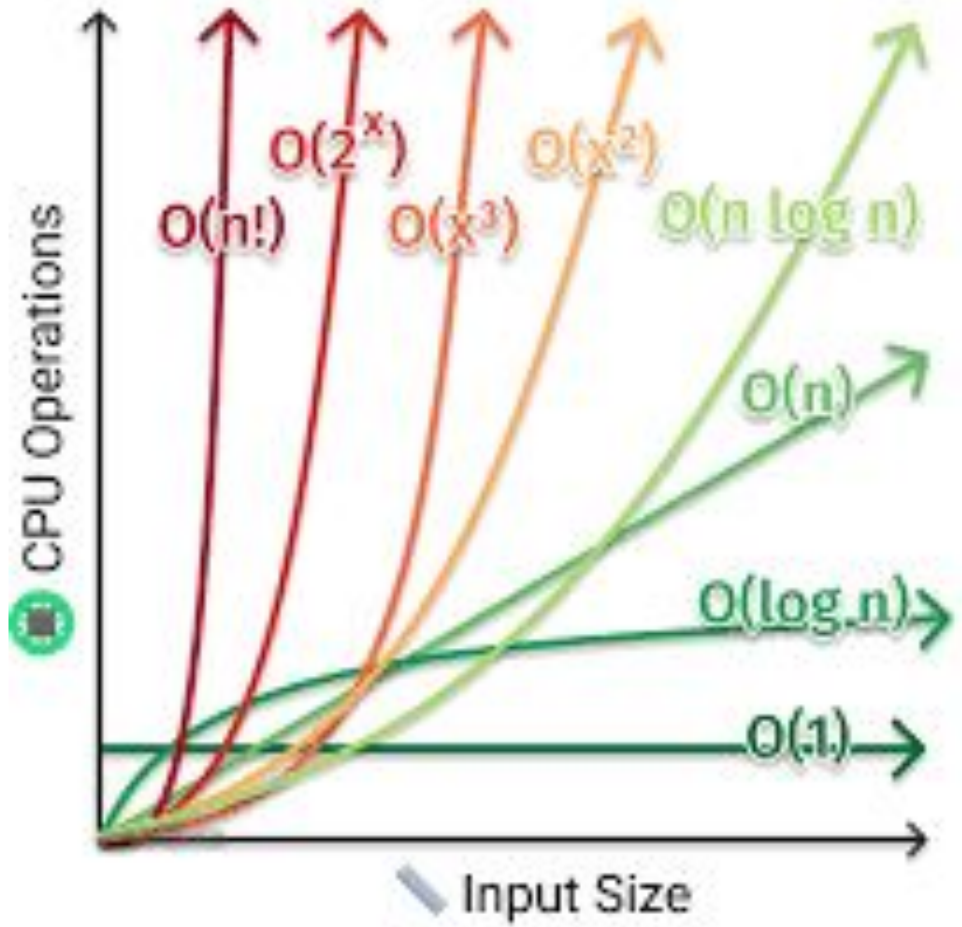
$10.5 (n ^ 2) + 13.17$

They are even more meaningless than units

This leaves us with

$n ^ 2$

⏱️ Time Complexity



Big O Notation

Three ways to talk about run time

- best case: $\Omega(n)$
- average case: $\Theta(n)$
- worst case: $O(n)$

Which Ones are Useful?

Calculating the solutions to a Rubik’s Cube

Throughput

Running a life support system

Latency

In CS club, we are not controlling life support systems

What About Best Case?

We Don't Talk About Best Case

Practice 1: n is our parameter

```python
x = 0
for i in range(n):
    for j in range(n):
        x += 1
```

```c
int x = 0;
for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
        x++;
    }
}
```

Practice 2: n is our parameter

```python
for i in range(n):
    for j in range(i, n):
        print(i, j)
```

```cpp
for(int i = 0; i < n; i++){
    for(int j = i; j < n; j++){
        cout << i << ' ' << j << '\n';
    }
}
```

Isn’t this slide deck about data structures?

Ways of Storing Data

Turns out that arrays aren’t that great to work with in practice

Their size is determined when they are first created

What if we don’t know how many elements we want to store?

Big Arrays?

What if we just make a big array

```
arr = [0]*100000
```

```
int arr[100000];
```

The problem with this is that we might end up only using 2 slots because, again, we don't know how many elements we need

Expandable Arrays

Every time we add an element to an array, we make a new array 1 larger and copy all the elements over:

```
newArr = [0] * (sz+1)
for i in range(sz):
    newArr[i] = arr[i]
arr = newArr

arr[sz] = newElem
sz += 1
```

```
int *newArr = new int[sz+1];
for(int i = 0; i < sz; i++) newArr[i] = arr[i];
delete[] arr;
arr = newArr;
arr[sz++] = newElem;
```

Time Complexity Time!

A Better Way

If we run out of space in an array, we double the size of the array

add 1 to [] -> [1]

add 2 to [1] -> [1, 2]

add 3 to [1, 2] -> [1, 2, 3, None]

add 4 to [1, 2, 3, None] -> [1, 2, 3, 4]

add 5 to [1, 2, 3, 4] -> [1, 2, 3, 4, 5, None, None, None]

add 6 to [1, 2, 3, 4, 5, None, None, None] -> [1, 2, 3, 4, 5, 6, None, None]

The ArrayList!

```python
if sz == capacity:
    capacity *= 2
    newArr = [0] * capacity
    for i in range(sz):
        newArr[i] = arr[i]
    arr = newArr

arr[sz] = newElem
sz += 1
```

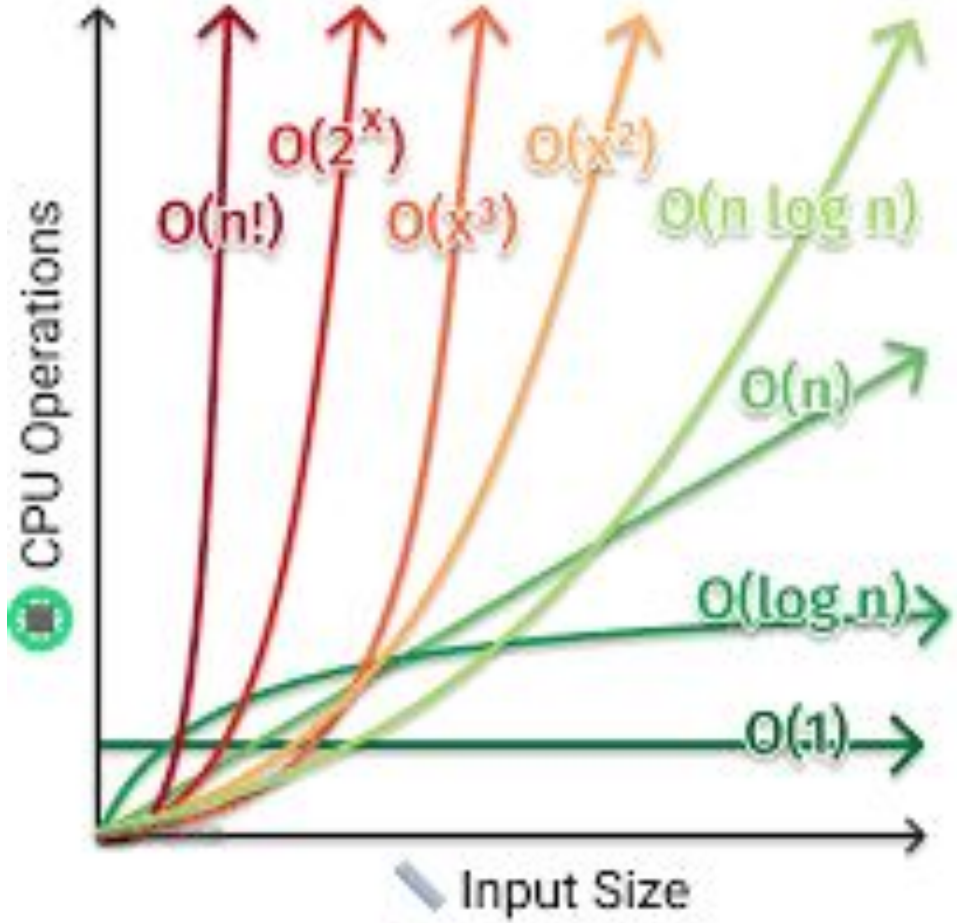
```cpp
if(sz == capacity){
    capacity *= 2;
    int *newArr = new int[capacity];
    for(int i = 0; i < sz; i++) newArr[i] = arr[i];
    delete[] arr;
    arr = newArr;
}
arr[sz++] = newElem;
```

Time Complexity Time!

Random Access

How long (relative to the size of the list) does it take to get the element at index i

⏱ Time Complexity



Adding an Element

What is the average time complexity of adding an element to an arraylist of length n?

fancy math:

$$\lim_{x \to \infty} \frac{\sum_{n=1}^{x} \begin{cases} n \text{ if is\_pow\_2}(n) \\ 1 \end{cases}}{x}$$

But what’s the worst case?

This is where we run into a problem:

Worst case is where we assume that everything that can go wrong will go wrong:

But what if the something that could go wrong is n being a specific value (like a power of 2)

Time complexity analysis is basically math, and being math there are multiple answers to this question

Amortized Worst Case

Here we basically ignore the outlier values

Note that we still assume that others things that could go wrong will go wrong

- Insertion sort is very slow when the list is sorted in reverse order, and the order of a list doesn’t have anything to do with the size, so we say insertion sort is very slow in the asymptotic worst case
- Most of the time, adding an element to an arraylist is O(1), so we just stick with O(1) time complexity, as nothing else can go wrong

Hard-Realtime Worst Case

This is where we are running a life support machine

If once every now and then it takes 1000x longer to calculate how much oxygen is needed, bad things will happen

So in hard-realtime worst case, we say adding an element to an arraylist is O(n)

What if I need a list on a life-support system?

we need a list that lets us add elements very quickly in the hard-realtime worst case

The LinkedList!

```
list = [1]

list2 = [2, list]

list3 = [3, list2]

print(list3) # [3, [2, [1, []]]]

print(list3[0]) # first element

print(list3[1][0]) # second element

print(list3[1][1][0]) # third element
```

Time Complexity Time!

Insertion

Lookup

Length

How is this any better?

It’s really not*

*Until we get to dynamic programming

Moving On!

Is the number 3 an element of my list?

to do this well, we need to give up random access :(

A Better Way

```
set = [False, False, False, True, False, False]

print(set[3]) # is the number 3 in my set


set[5] = True # add the number 5 to my set

print(set[5]) # True
```

Problems with this

what if i want to add the number 42?

what if i want to have negative numbers?

what if i want to have strings?

The Hash Table!

We have a “hash function” that takes in whatever value we want to store and spits out a positive number that is less than a given max

The hash table is an array of “hash buckets” (normally each bucket is a linked list)

When we want to add an element, we hash it with the max being the number of hash buckets, and put the element into the hash bucket at its hash index

When we want to know if an element is in our hash table, we hash it, and scan over the bucket to see if it is there

Code!

```
hash_table = [[], [], [], [], []] # a hash table with 5 empty hash buckets

# storing -13

hash = -13 % 5

hash_table[hash] = [-13, hash_table[hash]]

print(hash_table) # [[], [], [-13, []], [], []]
```

Code (continued)

```
hash_table = [[], [], [-13, []], [], []]

# does our hashtable have -13 in it?

hash = -13 % 5

bucket = hash_table[hash]

while bucket != [] and bucket[0] != -13:

        bucket = bucket[1]

if bucket != []:

        print(“found it!”)

else:

        print(“didn't find it!”)
```

Time Complexity Time!

Assume hash takes O(1)

Average case element addition

Average case element lookup

But this sucks!

We already decided to cut out meaningless numbers from our time complexities, so O(n / hash_buckets) looks a lot like O(n) in disguise.

Dynamic Hash Tables

Let’s do something similar to the array list: whenever we have as many elements as hash buckets, we double the size of the hash table

The problem is that our hash function will also change, so we need to rehash all the elements, which will take O(n)

This means that most of the time our hash table will take O(1), and every now and again it will take O(n)

Implementing this is left as an exercise to the viewer

Time Complexities!

Average Case

O(1)

Worst Case (Asymptotic)

O(1)

Worst Case (Hard-Realtime)

O(n)

A (seemingly unrelated) extension: Maps

A map is a data structure that holds a bunch of key value pairs.

If we wanted to store the price of fruit, we might have a map from strings (the name of the fruit) to numbers (the cost of the fruit)

Such a map should have fast key-value insertion and fast lookup

Sounds familiar

Maps are just hash tables in disguise!

```
map = [[], [], [], []] # empty map

# inserting the string “banana” with the cost 9.5

hash  = len(“banana”) % 4

map[hash] = [“banana”, 9.5, map[hash]]
```

```
…

# what is the value for the key “banana”

bucket = map[hash]

while bucket != [] and bucket[0] != “banana”:

    bucket = bucket[2]

if bucket[0] == “banana”:

    print(bucket[1])

else:

    print(“banana is not in our map”)
```

Time Complexity?

Hash Functions

The ones we’ve used have been terrible:

hash = len(“banana”) % 4

hash = -13 % 5

But what does terrible even mean?

time complexity?

distribution?

Both of our hash functions did pretty well on these

- modulo and len are both O(1)
- distribution is even

Predictability

We want our hash function to appear random.

it shouldn’t be easy to pick a bunch of keys for a hashmap that all hash to the same value.

Right now, the strings “abc” and “cba” and “zzz” and “123” all hash to the same value. It doesn’t take a genius to figure out what the hash function is.

Polynomial Rolling String Hash

(string[0] * p^0 + string[1] * p^1 + string[2] * p^2 … + string[n] * p^n) % m

where m and n are big prime numbers

Surprise!

You don’t really need ANY of this entire slide deck

(Except this next bit)

Why use your own data structures when you can ~~steal~~ borrow other peoples?

It turns out that implementing a hash table or an arraylist every time you need one is a pain

So python has done it for us!

Built in arraylists

Python arrays are actually arraylists

```
list = [1, 2, 3]

list.append(4)

print(list) # [1, 2, 3, 4]
```

HashSets

```
my_set = set()

my_set.add(37)

print(36 in my_set) # False

print(37 in my_set) # True
```

HashMaps

```
my_map = {“banana”: 9.5, “orange”: 17.33}

print(my_map[“banana”]) # 9.5
```

Linked Lists

Python doesn’t have a default linked list implementation, another reason to steer away from linked lists on the CCC.

Summary

All of these operations are O(1):

Arraylists

- Adding elements
- Random access

HashSets

- Adding elements
- Element presence

HashMaps

- Adding key-value pairs
- Key-value lookup

Bonus Content!

Don’t Repeat Yourself

Let’s say we want to use linked lists in python

Each time we want to get the length of a linked list, we would have to copy-paste the following piece of code:

```
tail = my_list

length = 0

while tail != []:

    tail = tail[1]

    length = length + 1
```

THIS SUCKS

A (bad) solution

Let’s write some code to solve this!

- ask the user for a file name
- read the file
- find all the times the word list_len appears in the file
- replace each instance with our code for the length of a linked list
- write the result back to the file

A better solution!

Turns out python has a way to reuse pieces of code that don’t involve potentially buggy pre-processors: functions

Functions!

```
def list_length(list):
    tail = list
    length = 0
    while tail != []:
        tail = tail[1]
        length = length + 1
    return length

print(list_length([1, [2, []]]))
```

Huh?

Turns out we’ve been using functions this whole time!

In python, whenever you see a name (like print), followed by parentheses, you are using a function call!

Anatomy of a Function

```
def list_length(list):
    tail = list
    length = 0
    while tail != []:
        tail = tail[1]
        length = length + 1
    return length

print(list_length([1, [2, []]]))
```

parameter

function name

function call

Function variables are private!

once a function exits, any variables it defined are no longer accessible.

```
def do_thing(n):
    i = n * 2
    print(i)
    return n


do_thing(13)
print(n) # error, undefined variable n
```

Functions have return values! (sometimes)

This allows you to assign the result of a function to a variable, ignore it, or even use it in another function call!

Some functions like print return None, a special value used to signal that the function has nothing meaningful to return.

If you want your function to return None, you can just not put a return statement and python will do it automatically.

Functions in the wild

Use functions as often as you can

But keep in mind they have a hidden cost

Practice Challenge!

Implement a rolling polynomial hash function

It should take in the string to hash, as well as p and m

Solution (in C++):

```cpp
typedef unsigned long long ull;
ull hash(string str, ull m, ull p){
    ll res = 0;
    for(char ch : str) res = (res*p + ch) % m;
    return res;
}
```

Questions?