CS2100

Queues

Recap

- HW due Friday - work w/ a partner
- Lab tomorrow
- Next Monday: review session

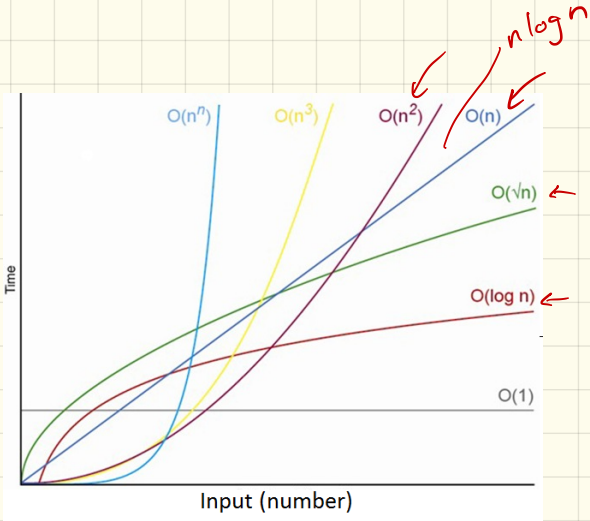
Tuesday: Midterm I

↳ no accomodation requested, let me know today if we need to arrange anything

Last time: Big O

Asymptotic analysis
↳ formal abstract way to compare running times

- in terms of input size



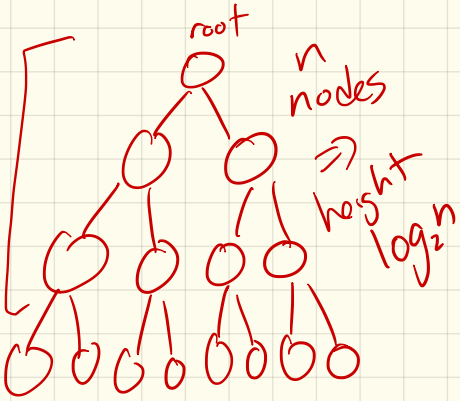
From here on out, we'll use this analysis for any function or data structure we code.

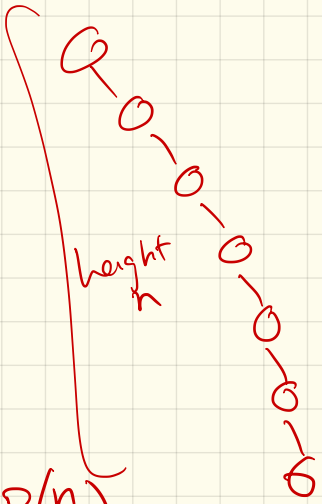
Some may be obvious:

- searching in a list
- nested for loops
- single for loop

Some harder:

- trees





Bin. Search:

$B(n) = 1 + B(\frac{n}{2})$

$= O(\log_2 n)$

Runtime of stack operations

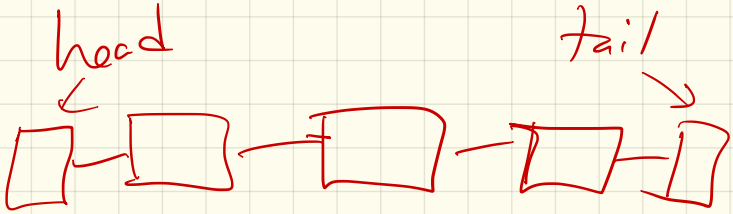
	Linked	Array
- size	O(1)	O(1)
- empty	O(1)	O(1)
- push	O(1)	O(1)
- pop	O(1)	O(1)
- top	O(1)	O(1)

→ used AddFront
w/ 3-4 pointer updates
stored data, new allocation
of 1 node

Today: Queues

British for what?

Line



add to back
remove from front

FIFO

Behavior: (STL-style)

```
#include <queue>
using namespace std;
int main() {
    queue<float> myQ;
    myQ.push(10.2);
    myQ.push(16.5);
    myQ.push(2.6);
    cout << myQ.top() << endl;
    myQ.pop();
    cout << myQ.top() << endl;
}
```

Output

10.2

16.5

front

front ~~10.2~~ - 16.5 - back 2.6

Setup + structure

This is also a simple data structure:

- limited functionality:

  5 functions

- but fast

  O(1) per function

Operations:

push
pop
front (or top)

size
empty

(see Cplusplus.com)

Implementation

2 options:

Linked: code from "scratch"

Array → not using SinkedList

Now - code!