Agenda

Coming up:

- Midterm 1 - Friday, Feb 21st, 5-7PM

This week:

- Assignment 4:
    - due this Sunday, 6PM (Feb. 16)

Today:

- Algorithm Complexity:
    - big-O notation
    - complexities of list implementations
- Stack data structure
- Course Survey

Visitor next lecture

Algorithm Complexity

Tuesday, September 17, 2019 2:33 PM

Various data structures exist, because some are better than others at various applications.

How do we actually quantify the performance?

Time our code? This gives us an absolute performance, but there are many factors that would influence the time:

- *system hardware resources*
- *state of operating system (concurrent processes)*
- *programming language*
- *compiler implementation*
- *how well the code is written*
- *ambient temperature :)*

Let's consider an example:

Say, it is 1995, Bill Gates just announced Windows 95. I have a Pentium II desktop and I am writing some C++ code to parse an array:

```
ifstream instream; //  1ms
int N = 10; //  1ms
string a[N]; //  1ms

int n = 0; //  1ms
while(inStream>>a[n]) //  10 ms
    n++;
```

Total time?

1ms x 4 + 10ms x N = 104 ms

What if N = 100 ?

4ms + 1000ms = 10,004 ms

What if N = 1e6 ?

10,000,004 ms

But, the individual time components might vary from one environment to another. What is the only **variable** we have affecting the execution time no matter the environment?

Complexity 2

Sunday, September 22, 2019 7:57 AM

continuing from previous page:

The one thing we know about this algo is that no matter what system it is run on, it will increase in time as N goes up in time.

This algorithm will need to perform N operations in order to complete.

We use something called the big-O notation to describe how an algorithm scales as N approaches infinity.

WIth big-O notation we drop all the constants and units.

O(N) - *not O(N+5) or O(4*N)*

The best possible complexity any algoirthm can have is O(1) (constant running time). Or not dependant on the input size.

Typically we use big-O to describe either **worst case** or average time performance.

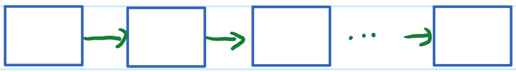
Space complexity can also be used to discuss an algorithm. (How the algorithm memory footprint scales with N.)

Complexity 3

Tuesday, September 17, 2019 1:43 PM

Let's compare the two kinds of lists we are familiar with:

array length n



# of nodes = n

What worst case big-O complexity does **inserting** into an array have?
O(N)
Array search? O(N)
Array access (assuming we know the index)? O(1)

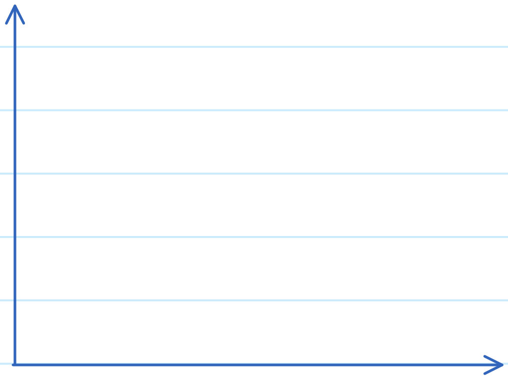
What about a linked list?
Insert? O(1)
Search? O(N)
Access? O(N)

Complexity is often used to compare sorting algorithms:

e.g. buble sort O(N^2) , heap sort O(N log(N))

With regards to data structures, we talk about complexity of common operations:

operations



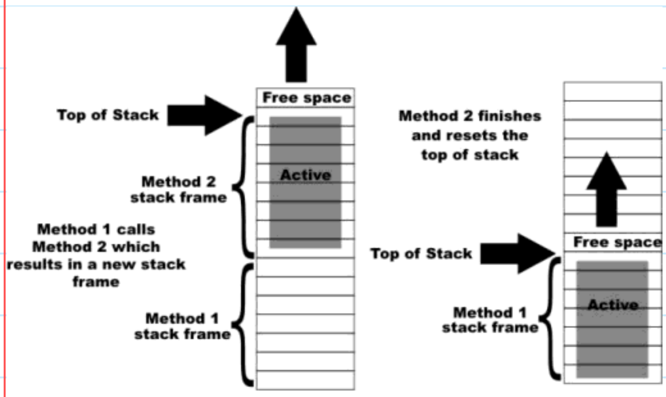
elements N

The Stack Data Structure

- Last In First Out data structure
- A "limited access" DS
    -
- Usage examples:
    -

example: Undo stack in editor

paste text
insert image
delete "there"
type "there"
type "hello"



credit: https://www.i-programmer.info

e.g. word processor "undo"

Stack ADT

private:

- top - keeps track of the top element
- maxSize - limit on total size of stack (optional - depends on implementation)
- count - current number of elements in stack

public:

- initialize() - constructor
- bool = isFull() - check whether stack is full
- bool = isEmpty - check if empty
- value = peek() - show the top item
- push(item) -
- pop() -
- disp() - traverse entire stack and print contents

Note that the ADT does not specify anything about the implementation.