COL100 Lecture 19

Note Title 08-10-2018

Review:

→ Big-O notation for measuring efficiency

Complexity class

(name) class	Big-O	if you double N	Example runtime
constant	$O(1)$	unchanged	10 ms
logarithmic	$O(\log_2 N)$	increases very slightly	175 ms
linear	$O(N)$	doubles	2.1 sec
log-linear	$O(N * \log_2 N)$	slightly ~~double~~ more than double	9 sec
quadratic	$O(N^2)$	multiplies by 4	2 minutes
quad-linear	$O(N^2 \cdot \log N)$	slightly more than 4x	8 minutes

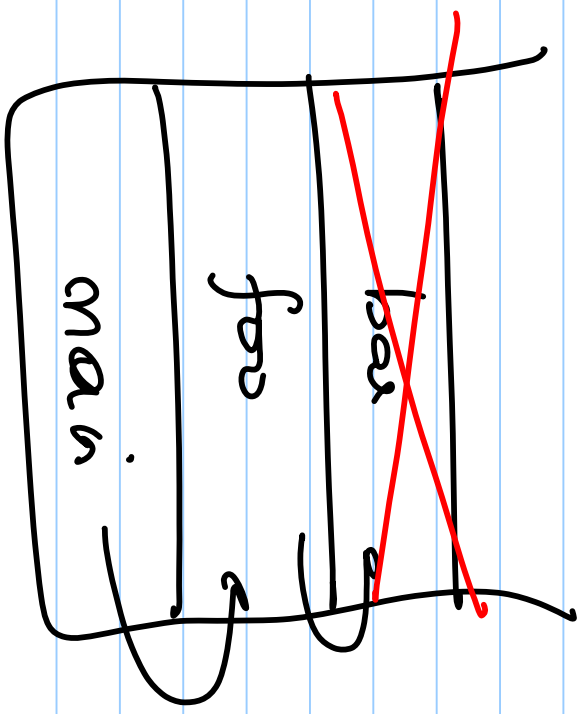
		if N doubles	
cubic	$O(N^3)$	8 x	1 hour 30 minutes
. . .			
. - .			
exponential	$O(2^N)$	multiplies drastically	10^{61} years
factorial	$O(N!)$	multiplies drastically	10^{1000} years

Stacks : another ADT

Vectors

"Last-in first-out" (LIFO)

allows users to add, access, and remove the last element


main
def
test

```
#include "stack.h"

Stack<int> s;
```

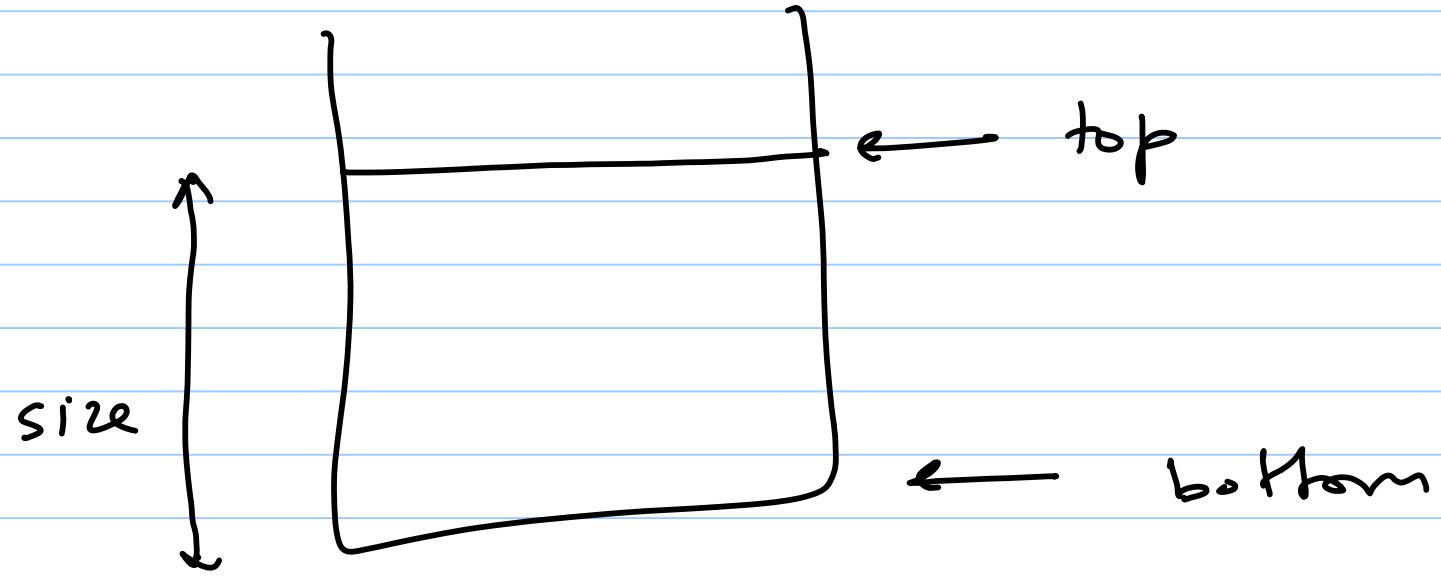
Member functions

1. s.push(value) : adds an element on top of stack, increasing its size by one

   → O(1) super-fast

2. s.pop() | removes top value and returns it; error if stack is empty

3. s.peek() | returns top value (without removing it). error if empty


top
size
bottom

4.	s.size()	returns number of elements in s	O(1)
5.	s.isEmpty()	boolean value TRUE or FALSE if stack has no elements	O(1)

Cannot access elements by index!

```
Stack<int> S;
for (int i = 0; i < s.size(); i++)
{
    S[i] = ...
}
```

✗ will not compile

```
// process an entire stack
// common pattern
while ( ! s.isEmpty() )
{
        int i = s.pop();
        // do something with i
}
```

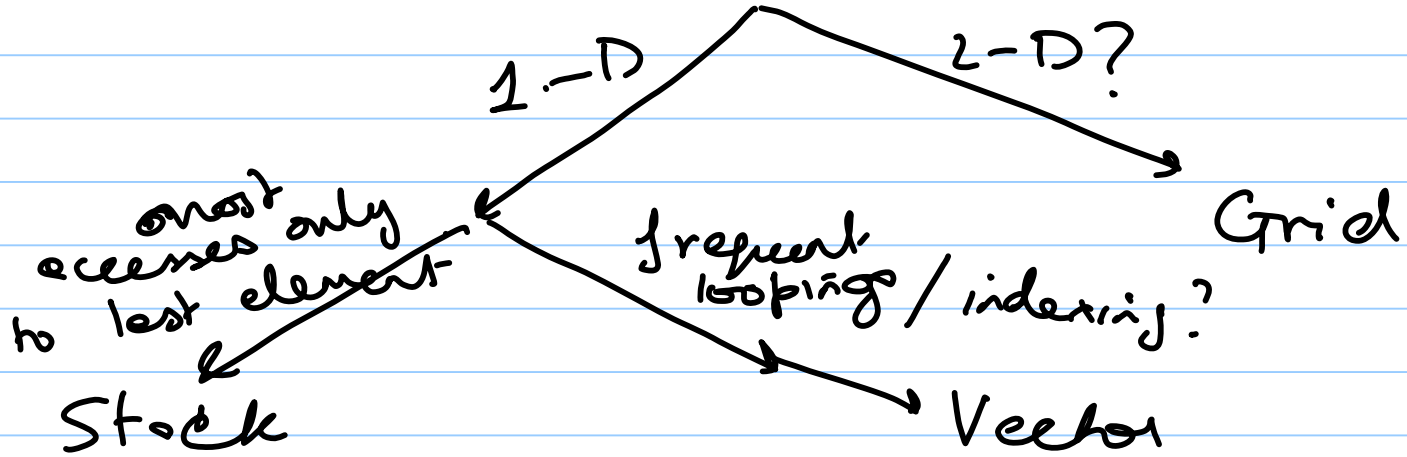
Exercise: Sentence reversal

Write a program with words of a sentence in reverse order

Input: "I am xyz"
Output: "xyz am I"

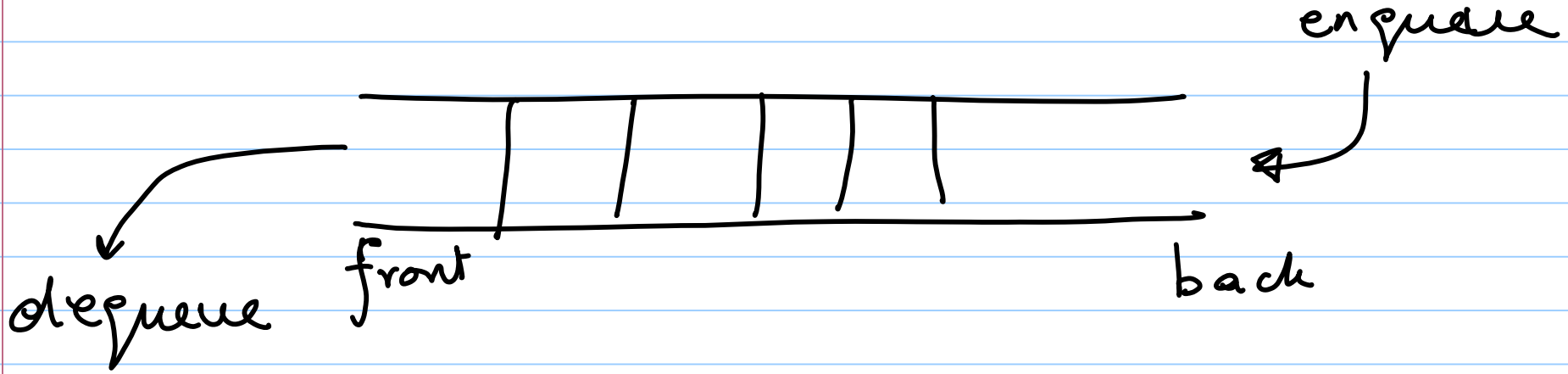
Input: "Hello"
Output: "Hello"

```
string printSentenceReverse(string const &str)
{
    string ret;
    istringstream iss(str);
    stack<string> s;
    string word;
    while (iss >> word)
    {
        s.push(word);
    }
    while (!s.isEmpty())
    {
        ret = ret + s.pop();
        ret = ret + " ";
    }
    return ret;
}
```


1-D
2-D?
Grid
mostly accesses to last element
frequent looping / indexing?
Stack
Vector

Queues : yet another ADT

"First In First Out" (FIFO)



```
#include "queue.h"

Queue<string> q;
Queue<int> qi;
```

Member functions

q.enqueue(value)	places given value at back of queue increasing its size by one	O(1)
q.dequeue()	removes front value and returns it. error if queue is empty	O(1)
q.peek()	returns front value without removing; error if empty	O(1)
q.size()	number of elements in q	O(1)
q.isEmpty()	returns true if q is empty	O(1)

```
Queue<int> q;                    // {} front=back
q.enqueue(42);                   // f{42}b
q.enqueue(-3);                   // f{42,-3}b
q.enqueue(17);                   // f{42,-3,17}b
cout << q.dequeue() << endl;     // f{-3,17}b
                                 // prints 42
cout << q.peek() << endl;        // f{-3,17}b
                                 // prints -3
cout << q.dequeue();             // f{17}b
                                 // prints -3
```

```
Queue <int>  q;
for (int i = 1; i <= 6; i++)
{
    q.enqueue(i);
}                          // q{1,2,3,4,5,6}

for (int i = 0; i < q.size(); i++)
{
    cout << q.dequeue() << " ";
}
```