Ex. Selection Sort : $\Theta(n^2)$ runtime

Quick Sort : $\Theta(n \log n)$

which is better ?

in fact $n \ln(n) = o(n^2)$ ✓

$$\lim_{n\to\infty} \left(\frac{n \ln n}{n^2}\right) = \lim_{n\to\infty} \left(\frac{\ln n}{n}\right)$$

$$= \lim_{n\to\infty} \left(\frac{1/n}{1}\right) = 0$$

Ex $n \ln(n) = \omega(n)$ ✓

$$\lim_{n\to\infty} \left(\frac{n \ln(n)}{n}\right) = \lim_{n\to\infty} \left(\ln(n)\right) = \infty$$

what about BFS & DFS ?

- BFS

size of input instance

$n = |V(G)|$ and $m = |E(G)|$

Initialization $\Theta(3n+5) = \Theta(n)$

every $x \in V$ goes through queue at most once. $\therefore$ queue ops cost atmost $\Theta(n)$ inner for-loop executes, in total

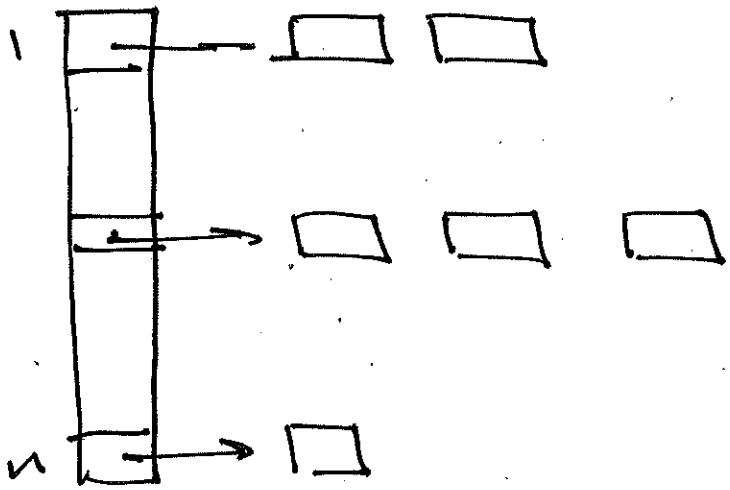
$$\sum_{x \in V} |adj(x)| = \sum_{x \in V} deg(x)$$

$$= \begin{cases} 2m & \text{undir.} \\ m & \text{dir.} \end{cases},$$

so that loop has total cost $\Theta(m)$.

$\therefore$ (total cost of BFS) $= \Theta(n+m)$

note: $n+m \approx$ a const. multiple of # of bytes needed to store the adj list representation



so $\Theta(n+m)$ is linear in size of input instance.

DFS $n = |V|$, $m = |E|$

Initialization: $\Theta(n)$

main loop (other than visit()): $\Theta(n)$

every vertex is visited exactly once. constant cost.

ops in visit() cost. $\Theta(n)$

for loop in visit() processes each adj list once, so its cost is

$$\sum_{x \in V} |adj(x)| = \sum_{x \in V} deg(x) = \begin{cases} 2m & \text{undir} \\ m & \text{dir.} \end{cases}$$

$$= \Theta(m)$$

(total cost) $= \Theta(n+m)$

again linear in # of bytes needed to store an input instance.

ADT's in C++

Pass: ext 3 days

A class in C++ is the Program implementation of an ADT.

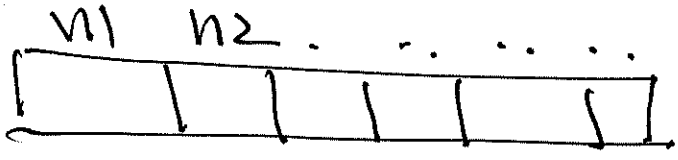
A struct in C++ is practically the same thing as a class.

major difference!

- struct members are default Public
- class " " " Private

most C++ programmers use structs for PODS (member vars only, apart from constructors.)

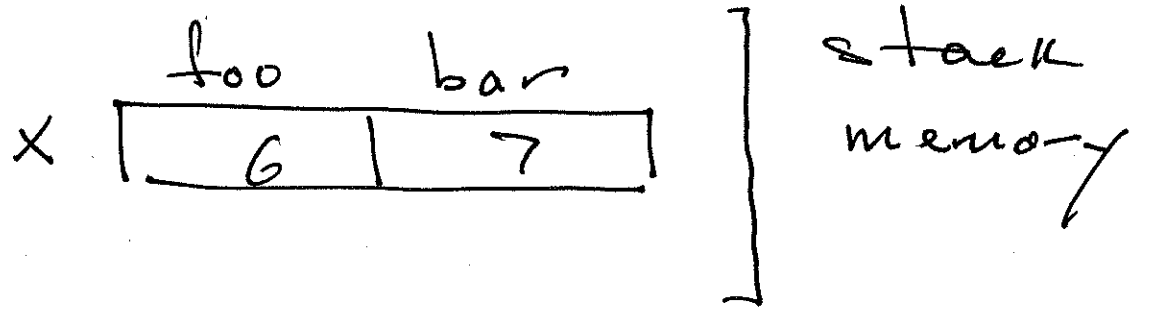
A struct in C is more like an array with field names in place of indices



In C :

```
struct Blah {
    int foo;
    int bar;
};
:
struct Blah x;
x.foo = 6
x.bar = 7
```

Keyword struct is necessary.

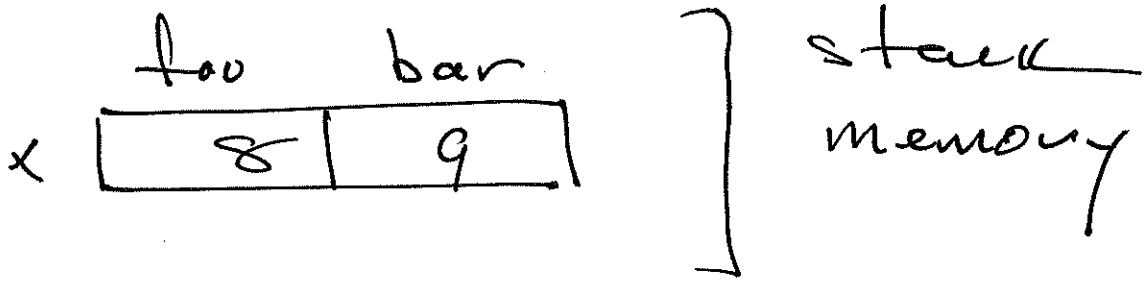


In C++

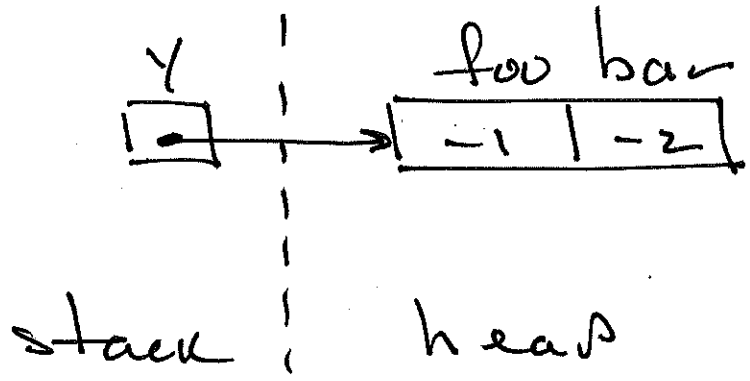
```
struct Blah {
    int foo;
    int bar;

    Blah (int a, int b){
        foo = a;
        bar = b;
    }
}
```

Calls constructor

```
Blah x(8, 9);
```



```
Blah* y = new Blah(-1, -2);
```



Basic skeleton for a class in C++:

```
class Blah {
private:
    // inner structs
    // member variables
    // helper functions
```

```
Public:
    // constructor (standard/copy)
    // destructor
    // member methods
    // overloaded operators
};
```

To build an ADT: 2 files

- Blah.h contains class defn and function prototypes.
- Blah.cpp contains fcn definitions

In Blah.h

```
class Blah{

private:
    // inner structs
    // member var
    // helper fcn. prototypes

public:
    // Prototypes for
    //     const. & dest.
    //     member methods
    //     overloaded operators
```

A class defines a namespace in C++. To access that namespace from outside the class defn, use the namespace resolution operator

::

In Blah.cpp

```
type Blah::fcn1(... Params ...){
    // code
}
```

Example: Queue.h
.cpp
QueueTest.cpp
Makefile.