**SEP 26**
**Tipple Tech Career Fair**
The Tipple Technology and Innovation Association is collaborating with the Association for Computing Machinery to bring together students from The University of Iowa’s Business Analytics and Information Systems (formerly MIS), Computer Science, and Informatics programs.

11:00am to 3:00pm
Currier Multi-Purpose Room - 100 Currier Hall
Tipple Tech | ACM
https://www.biz.uiowa.edu/tippietech/event/tippie-tech-career-fair-11

**SEP 27**
**Fall Job & Internship Fair**
The Fall Job and Internship Fair is a wonderful way to network with nationally-known organizations without having to leave campus! Whether you are in need of a full-time position or an internship, there will be plenty of organizations looking for candidates just like you!

Open to all majors! Over 150 employers will be available to talk about their internship and full-time opportunities! Bring copies of your resume, and dress professionally!

11:00am to 4:00pm
IMU - Main Lounge | 2nd Floor Ballroom
Pomerantz Career Center
https://careers.uiowa.edu/fall-job-internship-fair

**SEP 27**
**ACM Meeting - DuPont Pioneer Tech Talk**
Chad Bruggemann from DuPont will be presenting, as well as telling us about opportunities that exist at the company.

6:30pm to 7:30pm
118 MLH
Chad Bruggemann - ACM University of Iowa Chapter
https://acm.org.uiowa.edu/event/dupont-pioneer-tech-talk

**SEP 28**
**Fall 2017 Engineering Career Fair**
The Fall 2017 Engineering Career Fair will be geared towards engineering and computer science students graduating in December 2017 and May 2018, as well as students seeking spring, summer, and fall 2018 internships and co-ops.

The roster will include more than 100 manufacturing, consulting, health care, information technology (IT), government, and nonprofit organizations.


12:00pm to 5:00pm
IMU - Main Lounge
Searching an array

Let $R(N)$ be the running time to search for an integer in an unsorted array.

Can we find an $f(N)$ such that $R(N) \in O(f(N))$?
Searching an array

Let $R(N)$ be the worst-case running time to search for an integer in an unsorted array.

Algorithm: look at elements one-by-one until we find the integer or get to the end

Find an $f(N)$ such that $R(N) \in O(f(N))$

$$f(N) = N$$

or, in other words $R(N) \in O(N)$
Peer instruction

for (int i=0; i<array.length; i += 2) {
    if (array[i] > 10)
        break;
    else
        array[i] = i;
}

Let $R(N)$ be the worst-case running time of this code, where $N$ is the length of the array. Find an $f(n)$ such that $R(N) \in O(f(n))$.

a) 1
b) 2
c) 10
d) N
e) $N^2$

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
Searching an array

Let $R(N)$ be the worst-case running time to search for an integer in an unsorted array.

Find an $f(N)$ such that $R(N) \in O(f(N))$

$$f(N) = N$$

or, in other words $R(N) \in O(N)$

What if the array is sorted? Can we do better than $O(N)$?
Binary search for key 22
Binary search for key 22
Binary search for key 22
Binary search for key 22
Calculating $R(N)$

Let $c =$ time to compare two integers

$$R(N) = c + R(N/2)$$

How to solve the recurrence?
Calculating $R(N)$

let $c$=time to compare two integers

...but how many stages are there?

each stage does $c$ work
how many times can you halve* $N$ until you get to 1?

\[
\frac{N}{2^{\text{times}}} = 1
\]

\[
2^{\text{times}} = N
\]

times = \log_2 N

times needs to be integer so times = \lfloor \log_2 N \rfloor

*by “halve” we really mean \lfloor N/2 \rfloor
Calculating $R(N)$

let $c =$ time to compare two integers

there are $\lceil \log_2 N \rceil$ stages

$R(N) = c \times \lceil \log_2 N \rceil \in O(\log_2 N)$
Peer instruction

```c
int foo(int[] arr, int N) {
    int count; // Count the elements
    if (count==0) return 0;

    for (int i=0; i<N; i++) {
        arr[i]++;
    }

    return foo(arr, N/2);
}
```

Let $R(N)$ be the worst-case running time of a call to foo. Find an $f(n)$ such that $R(N) \in O(f(n))$.

a) $\log_N 2$
b) $\log_2 N$
c) $\frac{1}{2}$
d) $N^3$
e) $N^2$
f) $N \log_2 N$
g) $N^2$
Nested loops

Algorithm to find if an integer appears at least twice in an array.

```
int N = a.length;
for (int i=0; i < N; i++) {
    for (int k=i+1; k < N; k++) {
        if (a[i]==a[k]) return true;
    }
}
return false;
```

\[
N - 1 + N - 2 + ... + 2 + 1 = \frac{N(N-1)}{2}
\]

\[
\frac{1}{2}N^2 - \frac{1}{2}N
\]

\[R(N) \in O(N^2)\]

iterations are shaded

when i=0, the inner loop runs N-1 times
when i=1, the inner loop runs N-2 times...
when i=N-2 the inner loop runs 1 time
when i=N-1 the inner loop runs 0 times
Peer instruction

```java
for (int i=0; i<array.length; i++) {
    for (int j=1; j<array.length; j++) {
        for (int k=0; k<array.length; k+=j) {
            array[i] = array[j]+array[k];
        }
    }
}
```

Let \( R(N) \) be the worst-case running time of this code, where \( N \) is the length of the array. Find an \( f(n) \) such that \( R(N) \in O(f(n)) \).

a) \( N^2 \)  
b) \( N^3 \)  
c) \( N^4 \)  
d) \( N \ln N \)  
e) \( N^2 \ln N \)  
f) \( N^3 \ln N \)

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
We’ll look at two ways to compute the Nth Fibonacci number and analyze their running times

https://en.wikipedia.org/wiki/Fibonacci_number
Binary recursion to compute Fibonacci numbers

```java
public int fibBad(int n) {
    if (n==0) return 0;
    else if (n==1) return 1;
    else return fibBad(n-1) + fibBad(n-2);
}
```

\[ R(N) = \text{time to compute } \text{fibBad}(N-1) \]
\[ + \text{time to compute } \text{fibBad}(N-2) \]
\[ + \text{time to do everything else} \]

\[ R(N) = R(N-1) + R(N-2) + c \]

\[ R(N) \text{ will be exponential in } N \]
Linear recursion to compute Fibonacci numbers

```java
public static long[] fibGood(int n) {
    // computes fib(n) and fib(n-1) together
    if (n <= 1) {
        long[] ans = {n, 0};
        return ans;
    } else {
        long[] tmp = fibGood(n-1);
        long[] ans = {tmp[0]+tmp[1], tmp[0]};
        return ans;
    }
}
```

\[ R(N) = \text{time to compute } \text{fibGood}(N-1) \]
\[ + \text{ time to do everything else} \]

\[ R(N) = R(N-1) + c \quad R(N) \in O(N) \]
Peer instruction

```c
int weird(int[] arr, int N) {
    if (N== 0) return 0;
    if (N== 1) return arr[0];
    return weird(arr, N/3) + weird(arr, N-1);
}
```

Let $R(N)$ be the worst-case running time of a call to `weird`. Find an $f(n)$ such that $R(N) \in O(f(n))$.

a) $\log_3 N$
b) $\log_3 N + \log_1 N$

c) $\frac{1}{N^3}$
d) $N \ast \log_3 N$
e) $N$
f) $N^2 \ast \log_3 N$
g) $N^3$
h) $3^N$
Today’s big ideas

• big-Oh \( O(g(n)) \), big-omega \( \Omega(g(n)) \), and big-theta \( \Theta(g(n)) \) are used to describe the growth rate of functions

• can ask individually about worst, best, or average running time of an algorithm

• Try using this notation to describe the running time of some real algorithms!
Acknowledgements

• these slides borrow ideas from
  • http://homepage.divms.uiowa.edu/~ghosh/2116.html
  • http://datastructur.es/sp16/