CS2040S Tutorial 1

Julius (juliuss@u.nus.edu)

January 30, 2020

About me

- Year 3 in CS (Interest in Distributed Systems)

About me

- Year 3 in CS (Interest in Distributed Systems)
- Did CVWO for the past 2 summers

About me

- Year 3 in CS (Interest in Distributed Systems)
- Did CVWO for the past 2 summers
- Uses Emacs

Now your turn!

- Name
- Interests (Preferably not CS related)
- What was the nicest thing you ate over CNY?

What to expect

- CS2040S is intense

What to expect

- CS2040S is intense
  - Some of the material is new to me!

What to expect

- CS2040S is intense
  - Some of the material is new to me!
- Prepare for class

What to expect

- CS2040S is intense
  - Some of the material is new to me!
- Prepare for class
- Contribute to class

What to expect

- CS2040S is intense
  - Some of the material is new to me!
- Prepare for class
- Contribute to class
- I try to respond to emails in like ~24 hours

What to expect

- CS2040S is intense
  - Some of the material is new to me!
- Prepare for class
- Contribute to class
- I try to respond to emails in like ~24 hours
  - Or you can ping our Telegram group - others probably have the same query!

What to expect

- CS2040S is intense
  - Some of the material is new to me!
- Prepare for class
- Contribute to class
- I try to respond to emails in like ~24 hours
  - Or you can ping our Telegram group - others probably have the same query!
- I don't want to touch code!

Classes and Objects

- What is a class?
- What is an object?

Classes and Objects

```
class Student {
  private String name;

  public Student(String name) {
    this.name = name;
  }
}

Student hans = new Student("Hans");
```

Classes and Objects

```
class Student {
  private static String name;

  public Student(String name) {
    this.name = name;
  }
}

Student hans = new Student("Hans");
```

Classes and Objects

```
class Student {
  private String name;

  public Student(String name) {
    this.name = name;
  }

  private void sayName() {
    System.out.printf("Hi, my name is %s\n", this.name);
  }
}

Student hans = new Student("Hans");
hans.sayName();
```

Java vs Source

Object-orientedness

Everything in Java is an object (well, mostly).

Java vs Source

Object-orientedness

Everything in Java is an object (well, mostly).

Immutability

Because of the object-oriented nature of Java, immutability often isn't guaranteed.

Java vs Source

Object-orientedness

Everything in Java is an object (well, mostly).

Immutability

Because of the object-oriented nature of Java, immutability often isn't guaranteed.

Higher Order Functions

It's a pain in Java (Covered in CS2030).

Pass by value vs pass-by-reference

```
class X { String n = "hi"; }

void foo(X x) {
  x.n = "bye";
}

X x = new X();
foo(x);
System.out.println(x.n);
```

Pass by value vs pass-by-reference

```
class X { String n = "hi"; }

void goo(X x) {
  x = new X();
  x.n = "bye";
}

X xx = new X();
goo(xx);
System.out.println(xx.n);
```

Pass by value vs pass-by-reference

```
void hoo(int x) {
  x += 3;
}

int n = 3;
hoo(n);
System.out.println(n);
```

Formulas

- $f_1(n) = 7.2 + 34n^3 + 3254n$

Formulas

- $f_1(n) = 7.2 + 34n^3 + 3254n$
- $f_2(n) = n^2 \log n + 25n \log^2 n$

Formulas

- $f_1(n) = 7.2 + 34n^3 + 3254n$
- $f_2(n) = n^2 \log n + 25n \log^2 n$
- $f_3(n) = 2^{4 \log n} + 5n^5$

Formulas

- $f_1(n) = 7.2 + 34n^3 + 3254n$
- $f_2(n) = n^2 \log n + 25n \log^2 n$
- $f_3(n) = 2^{4 \log n} + 5n^5$
- $f_4(n) = 2^{2n^2+4n+7}$

Formulas

Let f and g be functions of n where $f(n) = O(n)$ and $g(n) = O(\log n)$.

- $h_1(n) = f(n) + g(n)$

Formulas

Let f and g be functions of n where $f(n) = O(n)$ and $g(n) = O(\log n)$.

- $h_1(n) = f(n) + g(n)$
- $h_2(n) = f(n) \times g(n)$

Formulas

Let f and g be functions of n where $f(n) = O(n)$ and $g(n) = O(\log n)$.

- $h_1(n) = f(n) + g(n)$
- $h_2(n) = f(n) \times g(n)$
- $h_3(n) = \max(f(n), g(n))$

Formulas

Let f and g be functions of n where $f(n) = O(n)$ and $g(n) = O(\log n)$.

- $h_1(n) = f(n) + g(n)$
- $h_2(n) = f(n) \times g(n)$
- $h_3(n) = \max(f(n), g(n))$
- $h_4(n) = f(g(n))$

Formulas

Let f and g be functions of n where $f(n) = O(n)$ and $g(n) = O(\log n)$.

- $h_1(n) = f(n) + g(n)$
- $h_2(n) = f(n) \times g(n)$
- $h_3(n) = \max(f(n), g(n))$
- $h_4(n) = f(g(n))$
- $h_5(n) = f(n)^{g(n)}$

Example 1

```
public static int niceFunction(int n)
{
        for (int i = 0; i < n; i++)
        {
                System.out.println("I am nice!");
        }
        return 42;
}

public static int meanFunction(int n)
{
        if(n == 0) return 0;
        return (2 * meanFunction(n/2) + niceFunction(n));
}

public static int evilFunction(int n)
{
        for(int i = 2; i < n; i *= i){
                System.out.println("To be or not to be");
        }
        return 666;
}
```

Example 2

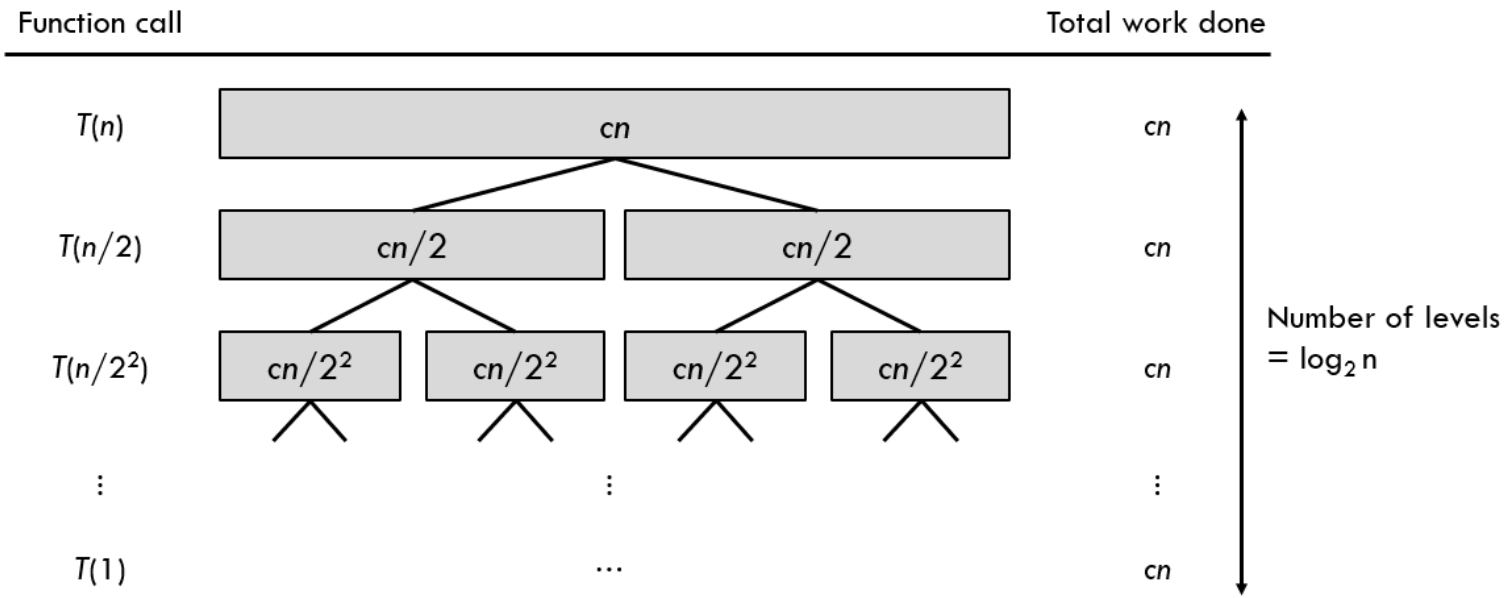
```
public int strangerFunction(int n) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
      System.out.println("Execute order?");
    }
  }
  return 66;
}
```

Example 2

```
public int suspiciousFunction(int n) {
  if (n == 0) return 2040;

  int a = suspiciousFunction(n / 2);
  int b = suspiciousFunction(n / 2);
  return a + b + niceFunction(n);
}
```

Example 2


Function call
Total work done
T(n)
cn
cn
T(n/2)
cn/2
cn/2
cn
T(n/2²)
cn/2²
cn/2²
cn/2²
cn/2²
cn
T(1)
cn
Number of levels
= log₂ n

Example 3

```
public int badFunction(int n) {
  if (n <= 0) return 2040;
  if (n == 1) return 2040;
  return badFunction(n - 1) + badFunction(n - 2) + 0;
}
```

Example 4

```java
public int metalGearFunction(int n) {
  for (int i = 0; i < n; i++) {
    for (int j = 1; j < i; j *= 2) {
      System.out.println("!");
    }
  }
  return 0;
}
```

When can we use binary search?

- Increasing/decreasing sequence of numbers

When can we use binary search?

- Increasing/decreasing sequence of numbers
- Can we think of a more general criteria?

When can we use binary search?

- Increasing/decreasing sequence of numbers
- Can we think of a more general criteria?
- Think about the peakfinding algorithm

A more general criteria

- Given a sequence of elements $E = [e_1, e_2, ..., e_j]$
- Want to find element e^* in E
- $\exists f$, such that $\forall i, f(e_i)$ can tell us:
  - if e_i is e^*, or
  - e^* is in $[e_1, e_2, ...e_{i-1}]$, or
  - e^* is in $[e_{i+1}, e_{i+2}, ...e_j]$

Example: Normal Binary Search

- Find 5 in [1,2,4,5,6,7]
- $E = [1, 2, 4, 5, 6, 7]$
- $f(x) =$
  - found if $x = 5$,
  - search in $[1, 2, ..., x - 1]$ if $x > 5$,
  - search in $[x + 1, x + 2...]$ if $x < 5$

Example: Peakfinding

- Find peak in [1,2,4,5,2,3].

Example: Peakfinding

- Find peak in [1,2,4,5,2,3].
- $f(x) = ?$

Example: Peakfinding

- Find peak in [1,2,4,5,2,3].
- $f(x) = ?$
  - $f(x) = \frac{d}{dx}(x)$

Example: Peakfinding

- Find peak in [1,2,4,5,2,3].
- $f(x) = ?$
  - $f(x) = \frac{d}{dx}(x)$
  - found if $f(x) = 0$,

Example: Peakfinding

- Find peak in [1,2,4,5,2,3].
- $f(x) = ?$
  - $f(x) = \frac{d}{dx}(x)$
  - found if $f(x) = 0$,
  - search in $[1, 2, ..., x - 1]$ if $f(x) < 0$,

Example: Peakfinding

- Find peak in [1,2,4,5,2,3].
- $f(x) = ?$
  - $f(x) = \frac{d}{dx}(x)$
  - found if $f(x) = 0$,
  - search in $[1, 2, ..., x - 1]$ if $f(x) < 0$,
  - search in $[x + 1, x + 2...]$ if $f(x) > 0$

Example: Peakfinding

- Find peak in [1,2,4,5,2,3].
- $f(x) = ?$
  - $f(x) = \frac{d}{dx}(x)$
  - found if $f(x) = 0$,
  - search in $[1, 2, ..., x - 1]$ if $f(x) < 0$,
  - search in $[x + 1, x + 2...]$ if $f(x) > 0$
- $E = ?$

Example: Peakfinding

- Find peak in [1,2,4,5,2,3].
- $f(x) = ?$
  - $f(x) = \frac{d}{dx}(x)$
  - found if $f(x) = 0$,
  - search in $[1, 2, ..., x-1]$ if $f(x) < 0$,
  - search in $[x+1, x+2...]$ if $f(x) > 0$
- $E = ?$
- $E = [[-\infty, 1, 2], [1, 2, 4], [2, 4, 5], [4, 5, 2], [5, 2, 3], [2, 3, \infty]]$

Peakfinding Implementation

- `map(E)` takes $O(n)$

Peakfinding Implementation

- `map(E)` takes $O(n)$
- Lazy evaluation

Peakfinding Implementation

- `map(E)` takes $O(n)$
- Lazy evaluation
- $f(x)$ is an operation where x is an index perhaps?

Binary search skeleton

```
def binary_search(arr, f):
  mid = len(arr) / 2
  if f(mid) == FOUND:
    return mid
  elif f(mid) == LEFT:
    # don't actually do this in python
    return binary_search(arr[:mid-1], f)
  else:
    return binary_search(arr[mid+1:], f)
```

- By reducing the algorithm to its simplest elements, we can ease our implementation

Let's try out today's problem

(Refer to Tutorial Sheet)

Invariants & Inductive properties

At *any* point of the execution you're at

- What can we say about our array?
- What can we say about indices?

Summary

Java

- OOP properties

Binary search

- Removing edge cases
- Abstracting reusable elements

To think about

- Your brother comes to you with a corrupted hard disk, such that any bit until position x is safe, but not the rest.

To think about

- Your brother comes to you with a corrupted hard disk, such that any bit until position x is safe, but not the rest.
- The good news: you have a hard disk debugger that tells you at a position p:

To think about

- Your brother comes to you with a corrupted hard disk, such that any bit until position x is safe, but not the rest.
- The good news: you have a hard disk debugger that tells you at a position p:
  - `TRUE` if the bit at p is safe,

To think about

- Your brother comes to you with a corrupted hard disk, such that any bit until position x is safe, but not the rest.
- The good news: you have a hard disk debugger that tells you at a position p:
  - `TRUE` if the bit at p is safe,
  - `FALSE` if the bit at p is either corrupted or not in the hard disk.

To think about

- Your brother comes to you with a corrupted hard disk, such that any bit until position x is safe, but not the rest.
- The good news: you have a hard disk debugger that tells you at a position p:
  - `TRUE` if the bit at p is safe,
  - `FALSE` if the bit at p is either corrupted or not in the hard disk.
- The bad news: your brother forgot how big his hard disk is

To think about

- Your brother comes to you with a corrupted hard disk, such that any bit until position x is safe, but not the rest.
- The good news: you have a hard disk debugger that tells you at a position p:
  - `TRUE` if the bit at p is safe,
  - `FALSE` if the bit at p is either corrupted or not in the hard disk.
- The bad news: your brother forgot how big his hard disk is
- How can we find x?