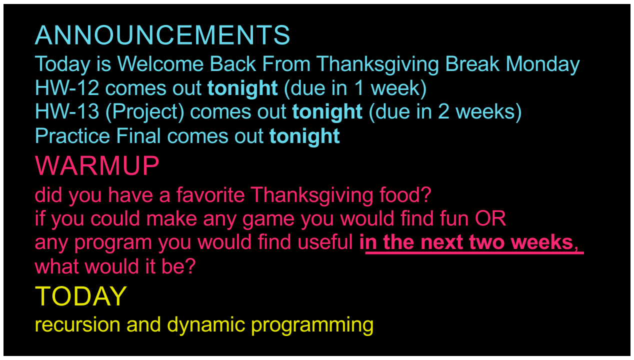


ANNOUNCEMENTS

Today is Welcome Back From Thanksgiving Break Monday
HW-12 comes out **tonight** (due in 1 week)
HW-13 (Project) comes out **tonight** (due in 2 weeks)
Practice Final comes out **tonight**

WARMUP

did you have a favorite Thanksgiving food?
if you could make any game you would find fun OR
any program you would find useful **in the next two weeks,**
what would it be?

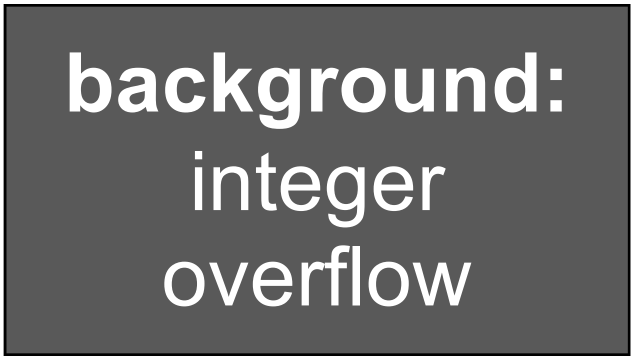
TODAY

recursion and dynamic programming

610

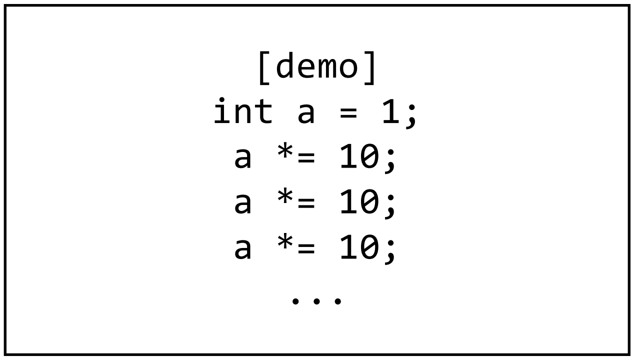
record LEC-02

611



background:
integer
overflow

612

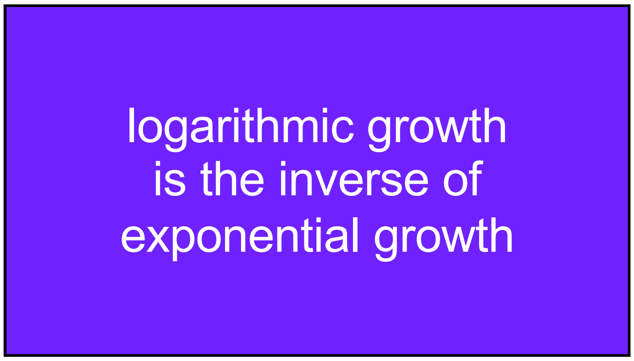


```
[demo]
int a = 1;
a *= 10;
a *= 10;
a *= 10;
...
```

613

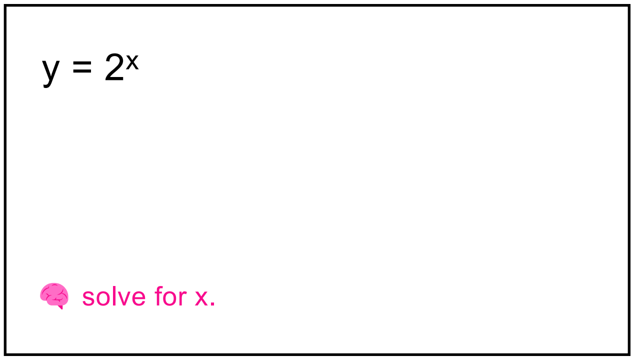
review:
$\log_2 n$ and 2^n

614



logarithmic growth
is the inverse of
exponential growth

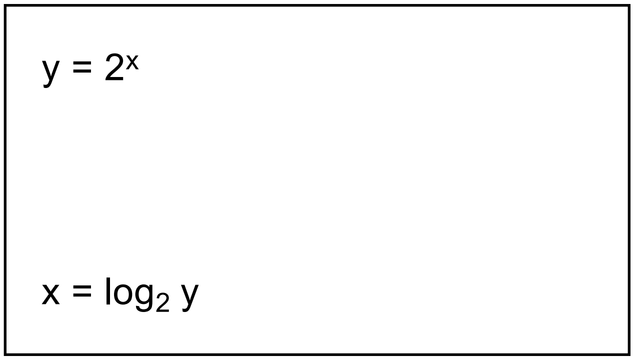
615



$y = 2^x$

solve for x.

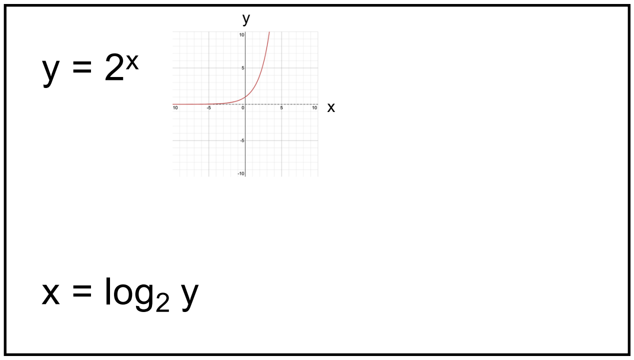
616



$y = 2^x$

$x = \log_2 y$

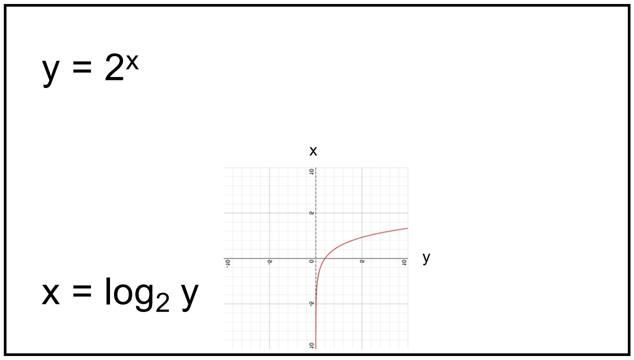
617



$y = 2^x$

$x = \log_2 y$

618



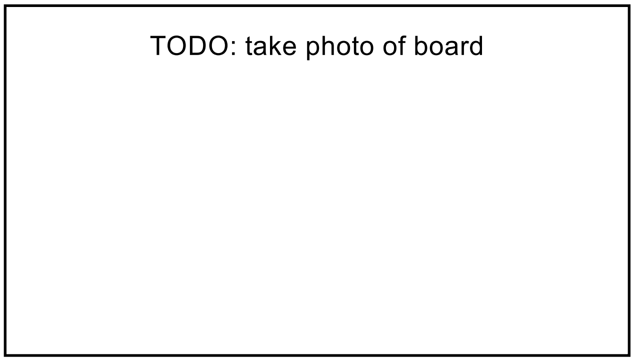
$y = 2^x$

$x = \log_2 y$

619

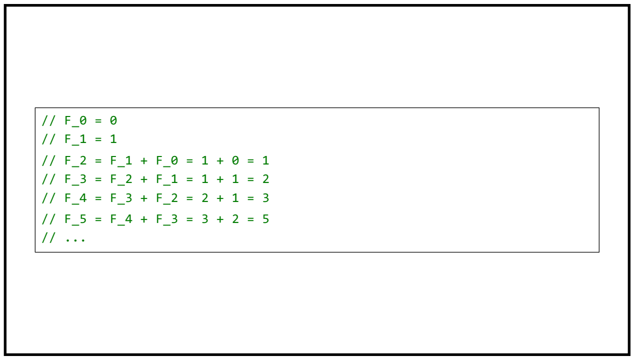
the Fibonacci sequence

620



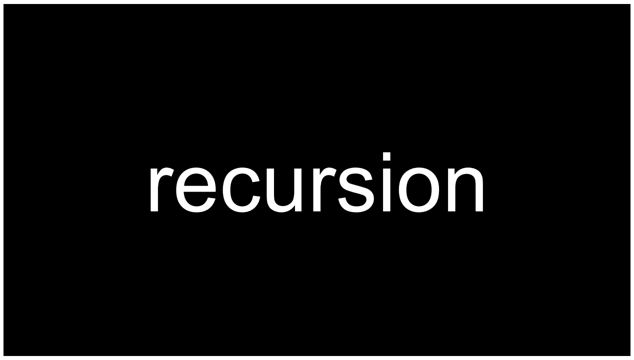
TODO: take photo of board

621



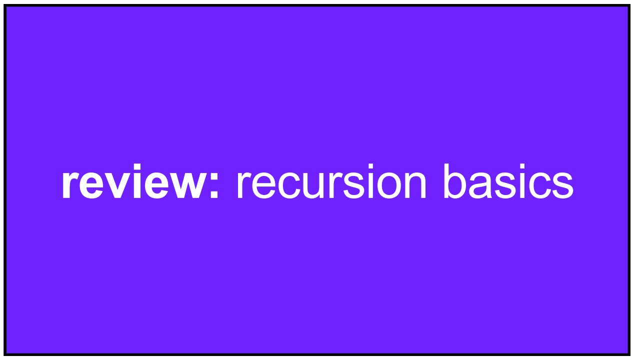
```
// F_0 = 0
// F_1 = 1
// F_2 = F_1 + F_0 = 1 + 0 = 1
// F_3 = F_2 + F_1 = 1 + 1 = 2
// F_4 = F_3 + F_2 = 2 + 1 = 3
// F_5 = F_4 + F_3 = 3 + 2 = 5
// ...
```

622



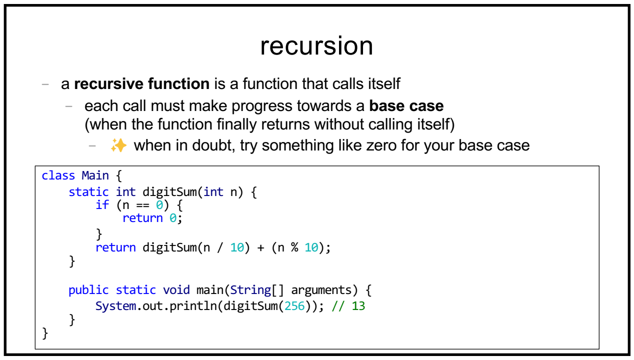
recursion

623



review: recursion basics

624

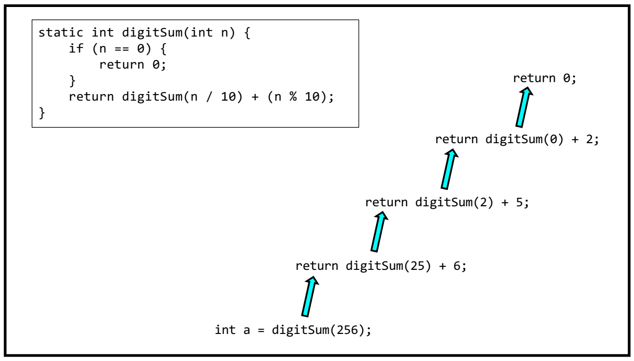


recursion

- a **recursive function** is a function that calls itself
  - each call must make progress towards a **base case** (when the function finally returns without calling itself)
    - ✨ when in doubt, try something like zero for your base case

```
class Main {
    static int digitSum(int n) {
        if (n == 0) {
            return 0;
        }
        return digitSum(n / 10) + (n % 10);
    }

    public static void main(String[] arguments) {
        System.out.println(digitSum(256)); // 13
    }
}
```

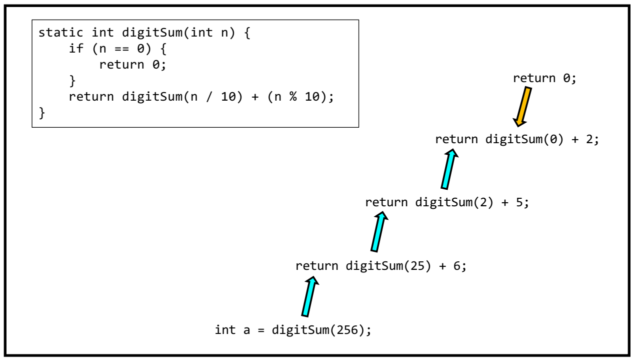
625



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 0;
return digitSum(0) + 2;
return digitSum(2) + 5;
return digitSum(25) + 6;
int a = digitSum(256);
```

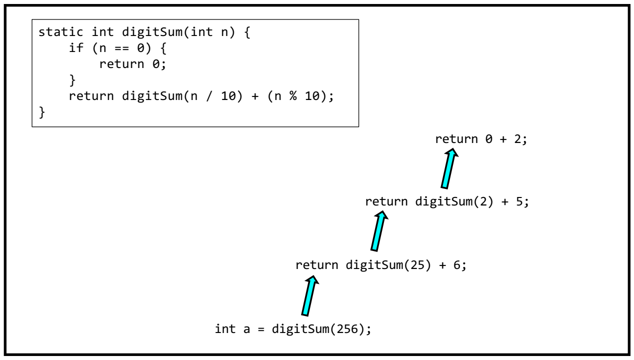
626



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 0;
return digitSum(0) + 2;
return digitSum(2) + 5;
return digitSum(25) + 6;
int a = digitSum(256);
```

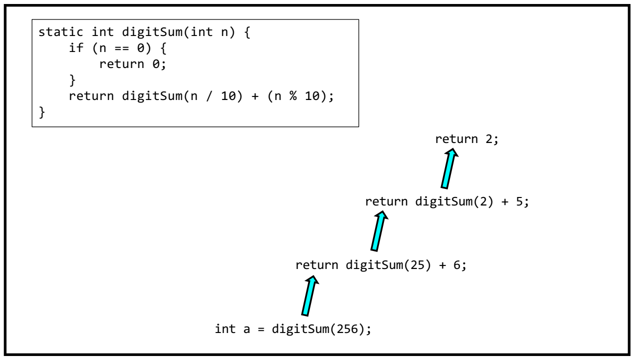
627



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 0 + 2;
return digitSum(2) + 5;
return digitSum(25) + 6;
int a = digitSum(256);
```

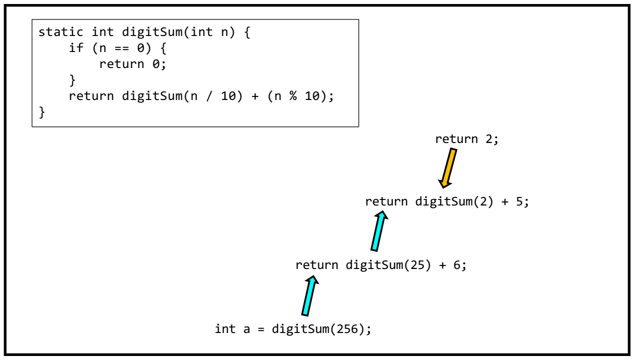
628



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 2;
return digitSum(2) + 5;
return digitSum(25) + 6;
int a = digitSum(256);
```

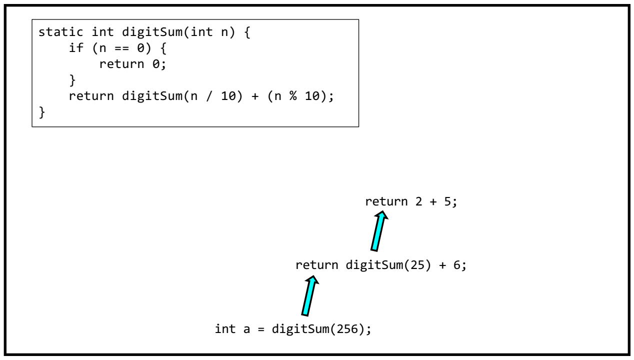
629



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 2;
return digitSum(2) + 5;
return digitSum(25) + 6;
int a = digitSum(256);
```

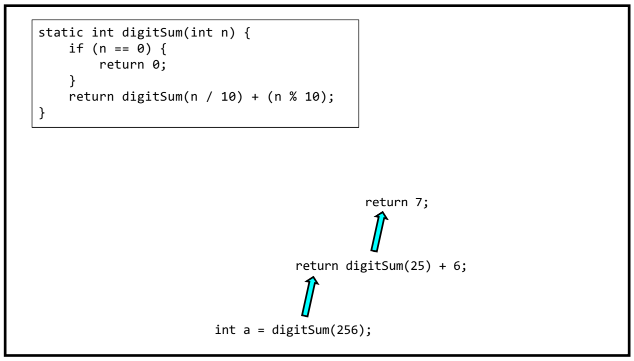
630



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 2 + 5;
return digitSum(25) + 6;
int a = digitSum(256);
```

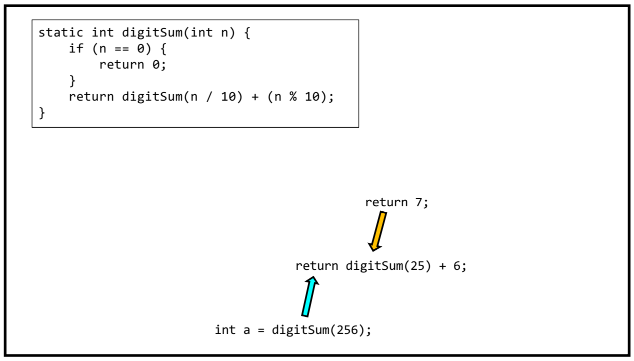
631



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 7;
return digitSum(25) + 6;
int a = digitSum(256);
```

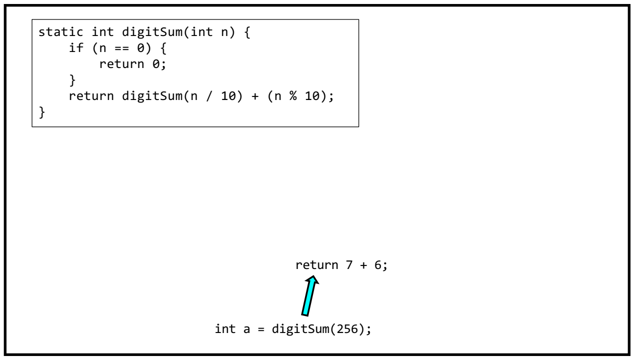
632



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 7;
return digitSum(25) + 6;
int a = digitSum(256);
```

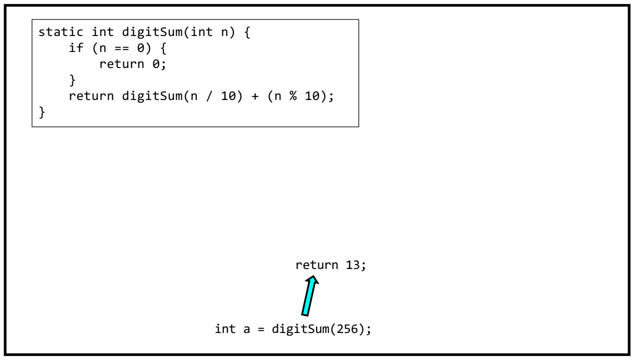
633



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 7 + 6;

int a = digitSum(256);
```

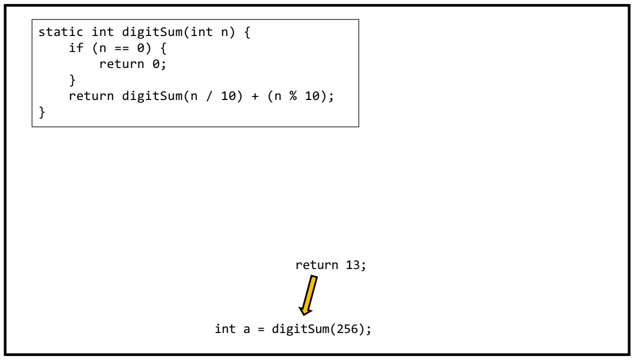
634



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 13;

int a = digitSum(256);
```

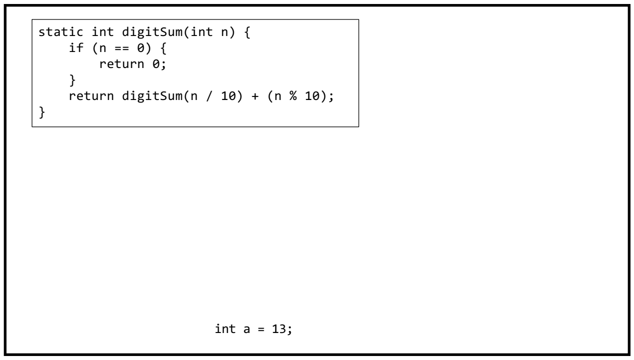
635



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
return 13;

int a = digitSum(256);
```

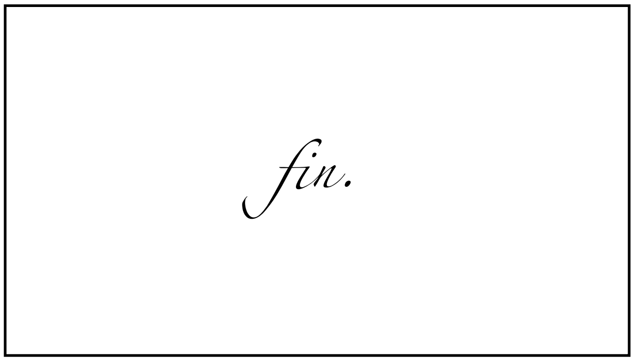
636



```
static int digitSum(int n) {
    if (n == 0) {
        return 0;
    }
    return digitSum(n / 10) + (n % 10);
}
```

```
int a = 13;
```

637



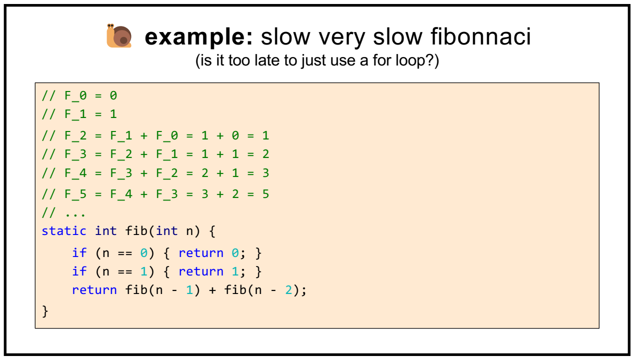
fin.

638

⚠ recursion hazard 1

repeated computation

639

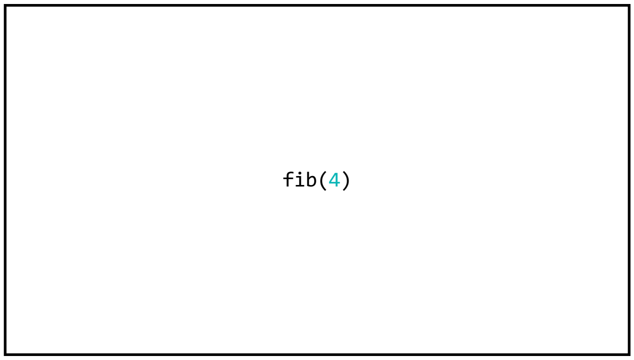


🐌 **example:** slow very slow fibonnaci

(is it too late to just use a for loop?)

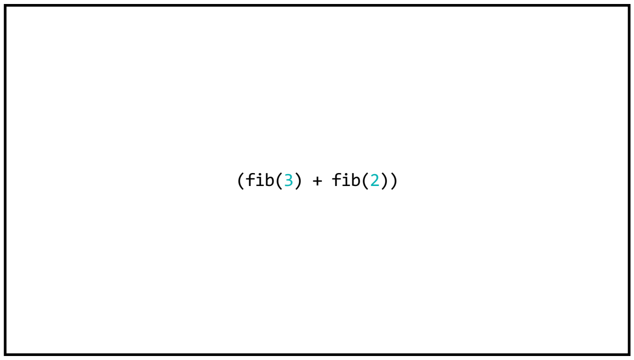
```
// F_0 = 0
// F_1 = 1
// F_2 = F_1 + F_0 = 1 + 0 = 1
// F_3 = F_2 + F_1 = 1 + 1 = 2
// F_4 = F_3 + F_2 = 2 + 1 = 3
// F_5 = F_4 + F_3 = 3 + 2 = 5
// ...
static int fib(int n) {
    if (n == 0) { return 0; }
    if (n == 1) { return 1; }
    return fib(n - 1) + fib(n - 2);
}
```

640



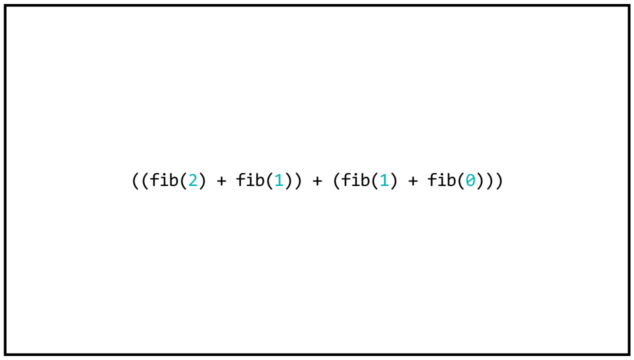
```
fib(4)
```

641



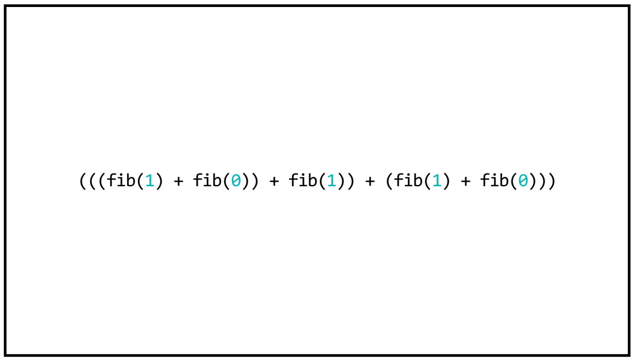
```
(fib(3) + fib(2))
```

642



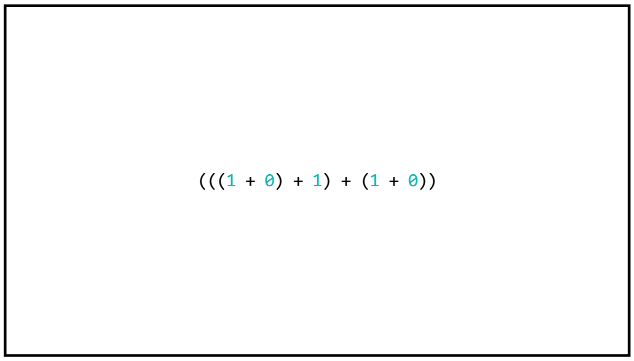
```
((fib(2) + fib(1)) + (fib(1) + fib(0)))
```

643



```
(((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0)))
```

644



```
(((1 + 0) + 1) + (1 + 0))
```

645

```
((1 + 1) + 1)
```

646

```
(2 + 1)
```

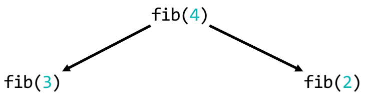
647

```
3
```

648

```
fib(4)
```

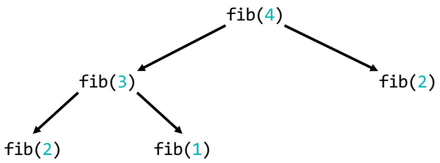
649


fib(4)

fib(3)

fib(2)

650


fib(4)

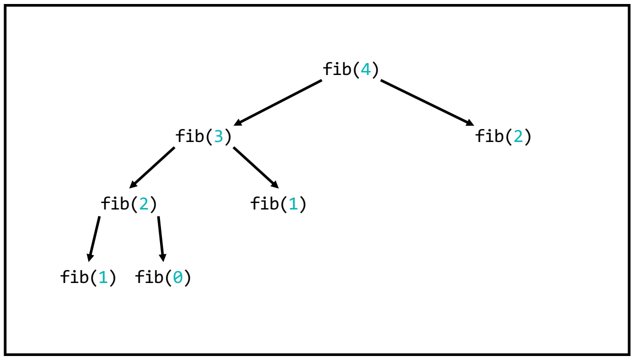
fib(3)

fib(2)

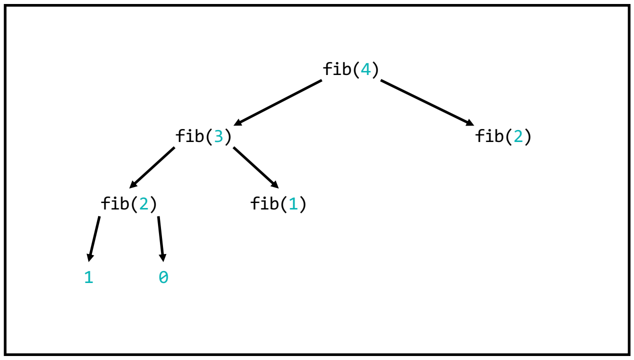
fib(2)

fib(1)

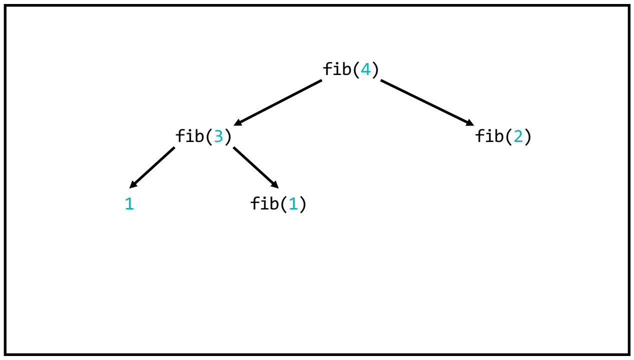
651



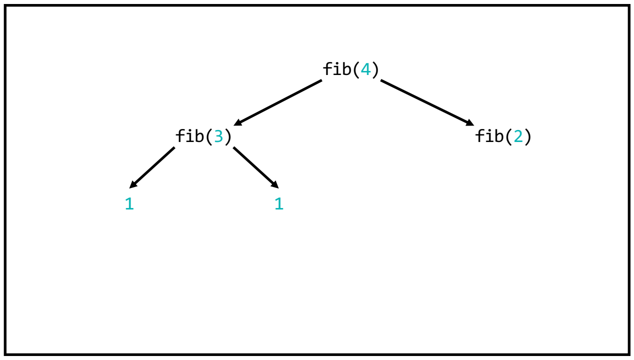
652



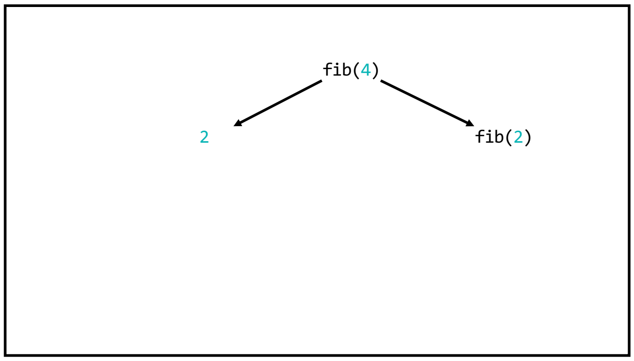
653



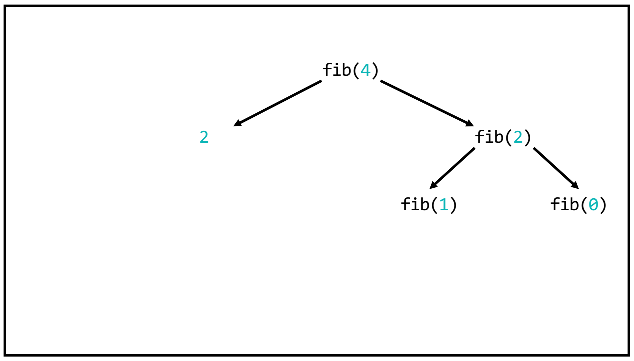
654



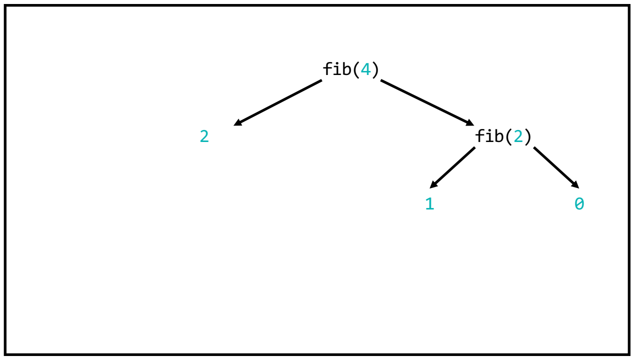
655



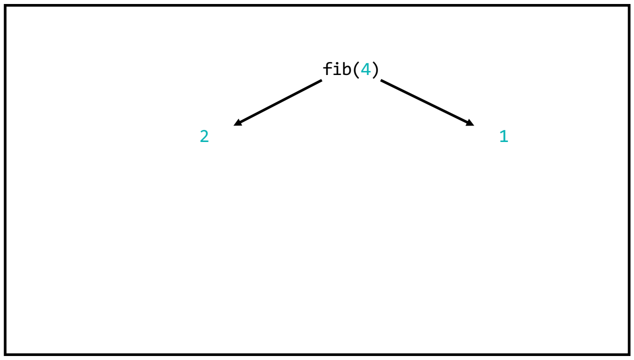
656



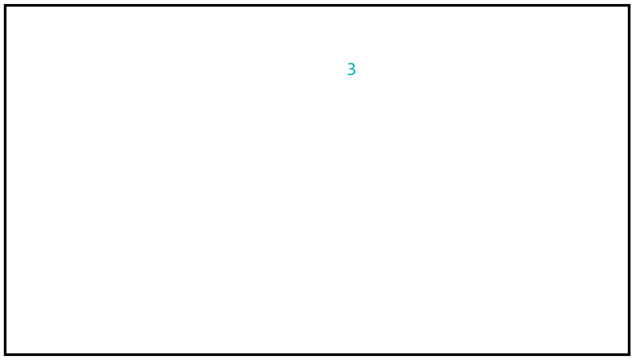
657



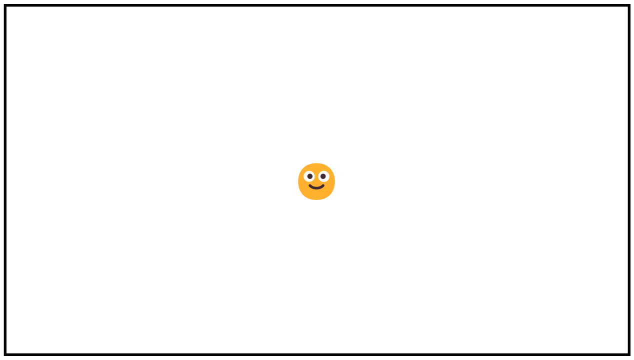
658



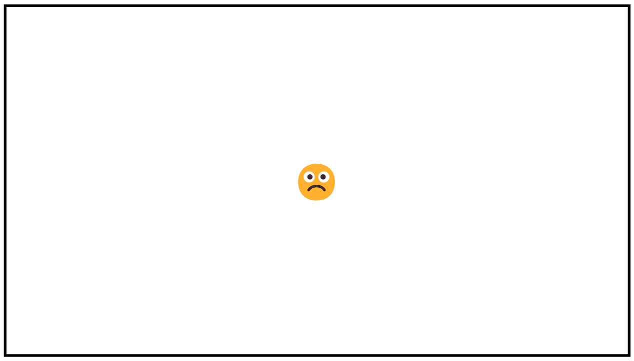
659


3

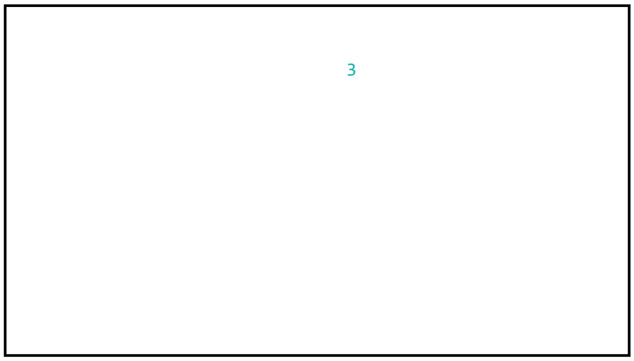
660



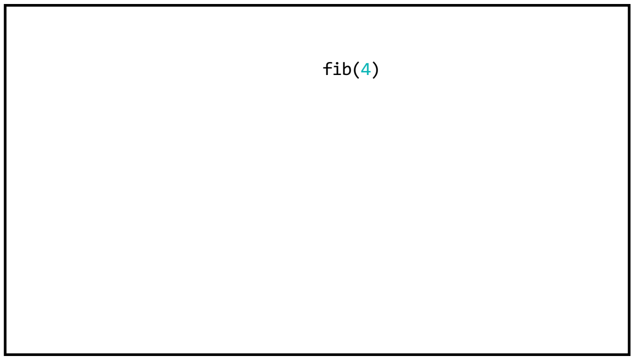
661



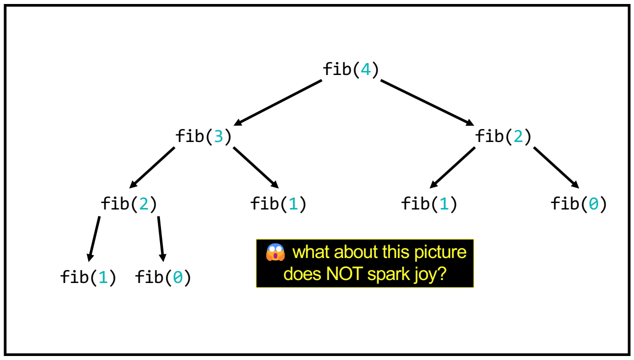
662


3

663

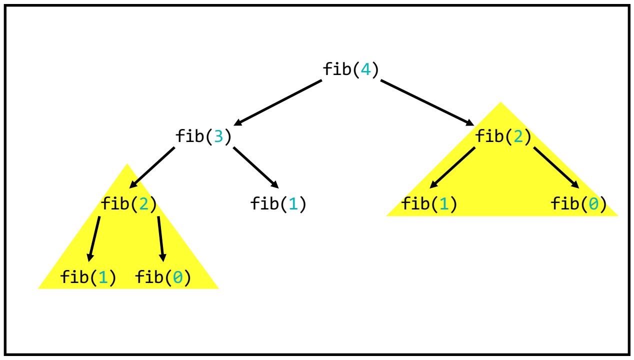


664

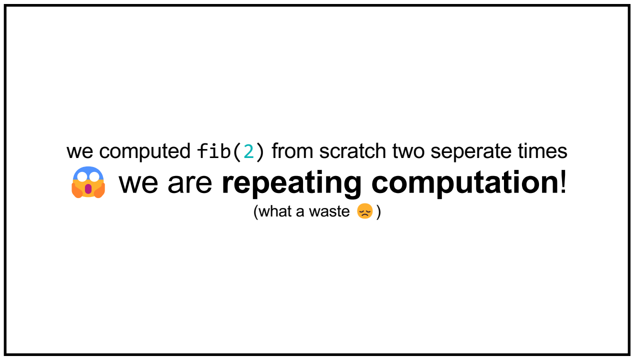


😱 what about this picture does NOT spark joy?

665



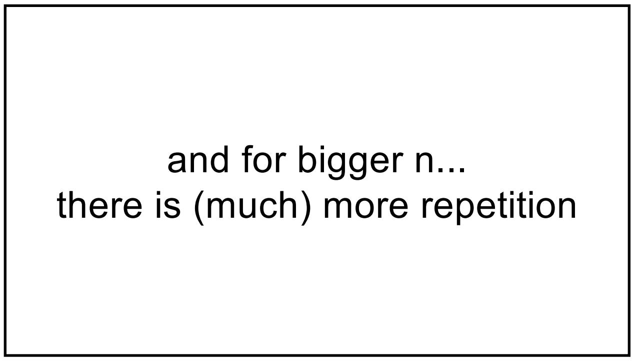
666


we computed `fib(2)` from scratch two seperate times

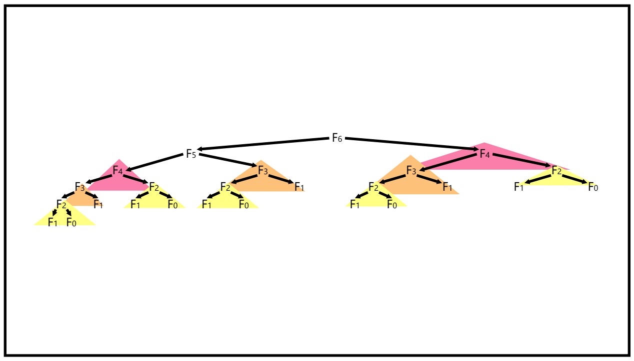
😱 we are **repeating computation**!

(what a waste 😞)

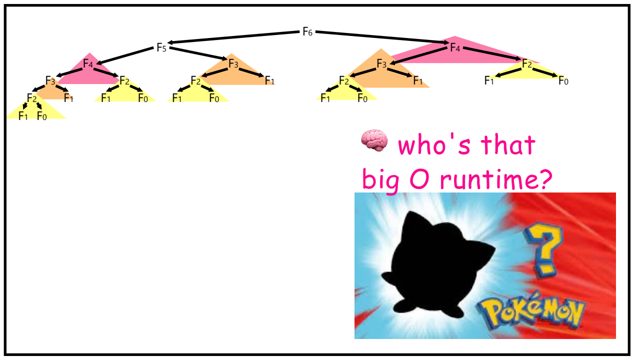
667


and for bigger n...
there is (much) more repetition

668

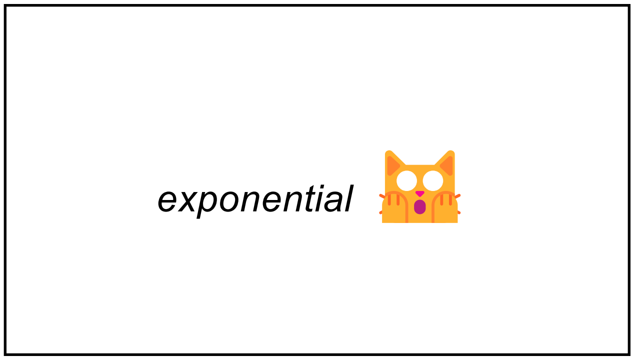


669



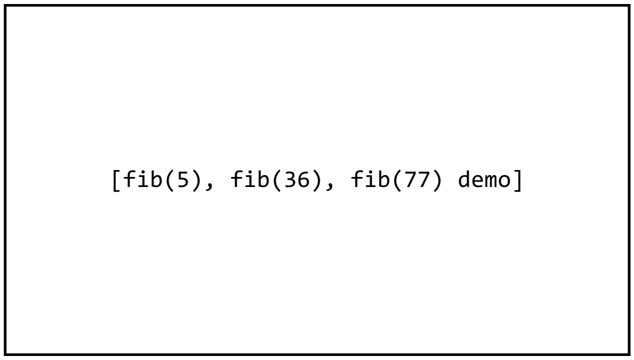
🧠 who's that big O runtime?

670



exponential 🙀

671



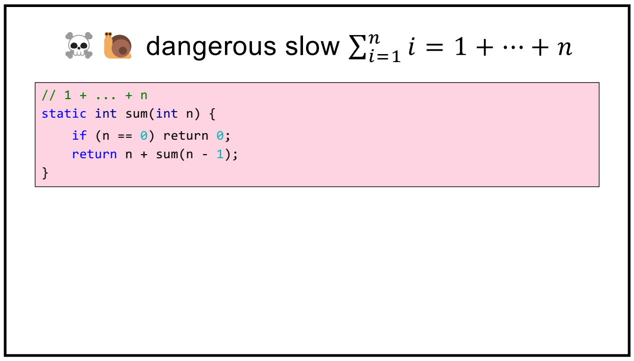
[fib(5), fib(36), fib(77) demo]

672

⚠️ recursion hazard 2

stack overflow

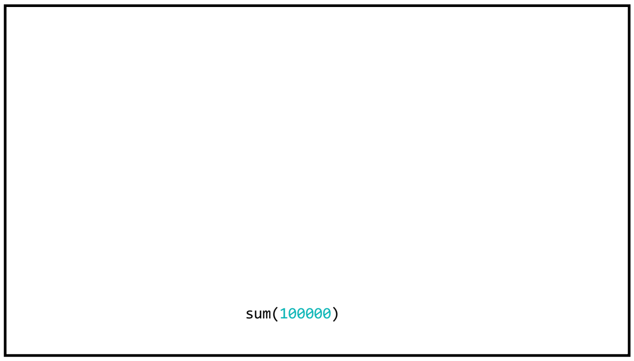
673



☠️ 🐌 dangerous slow $\sum_{i=1}^{n} i = 1 + \cdots + n$

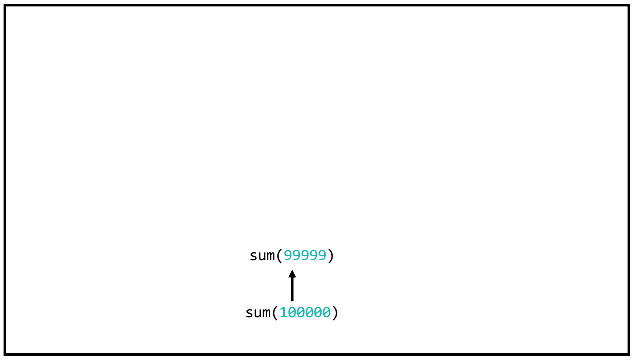
```
// 1 + ... + n
static int sum(int n) {
    if (n == 0) return 0;
    return n + sum(n - 1);
}
```

674



```
sum(100000)
```

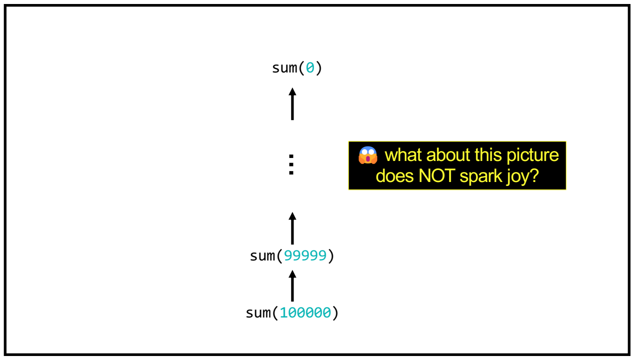
675



sum(99999)

sum(100000)

676



sum(0)

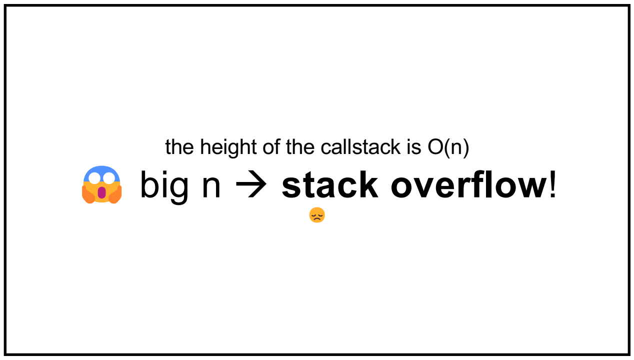
⋮

sum(99999)

sum(100000)

😱 what about this picture does NOT spark joy?

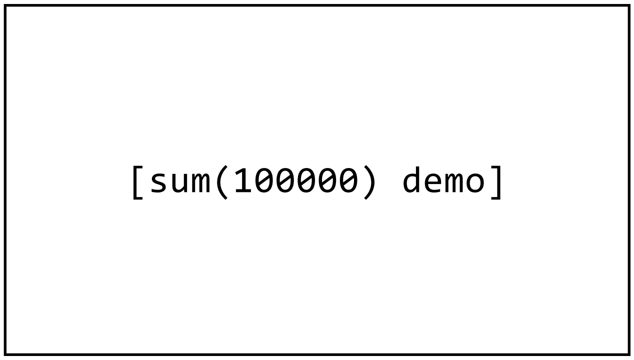
677



the height of the callstack is O(n)

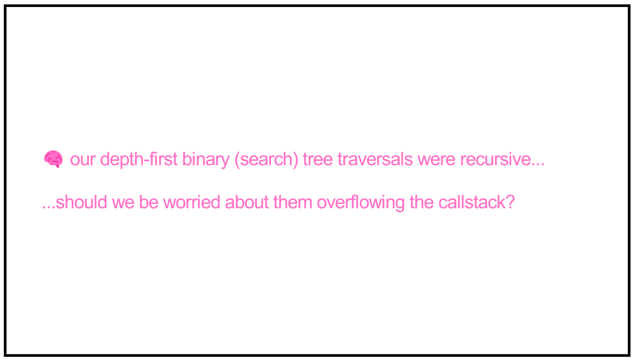
😱 big n → **stack overflow**! 😞

678



[sum(100000) demo]

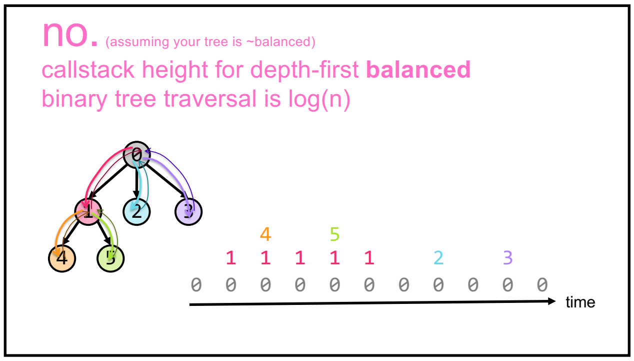
679



🧠 our depth-first binary (search) tree traversals were recursive...

...should we be worried about them overflowing the callstack?

680



no. (assuming your tree is ~balanced)

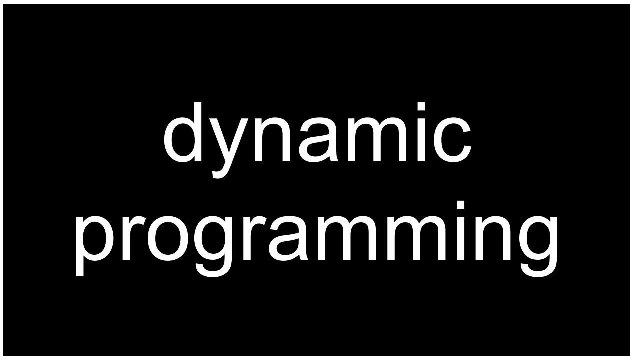
callstack height for depth-first **balanced** binary tree traversal is log(n)

681

*additionally, some languages/compilers** have "tail-call optimization," which would prevent a stack overflow for `sum(100000)`

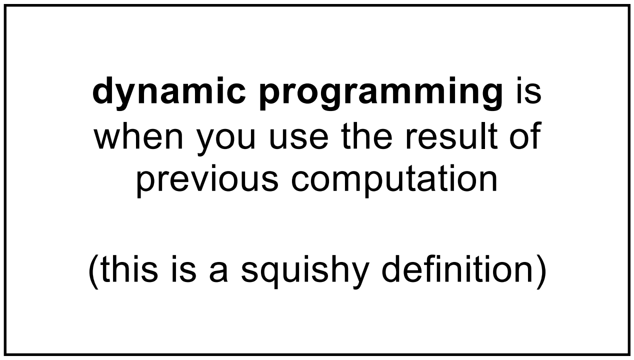
**Java is not one of these languages (as our demo showed)

682



dynamic programming

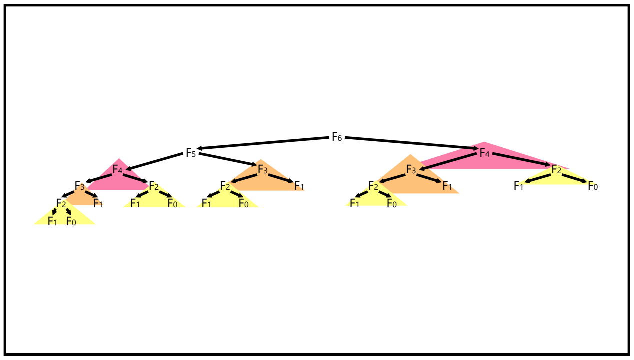
683



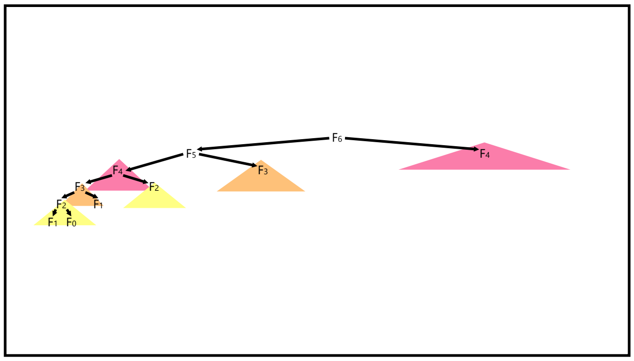
dynamic programming is when you use the result of previous computation

(this is a squishy definition)

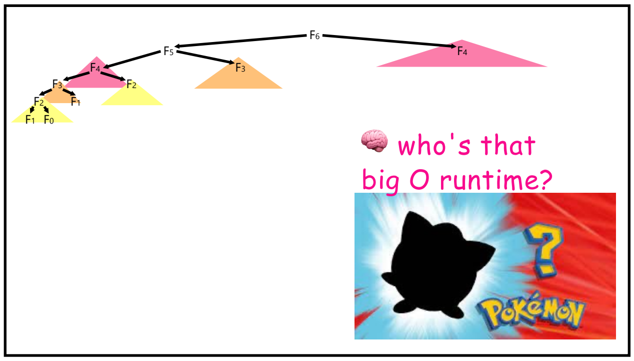
684



685

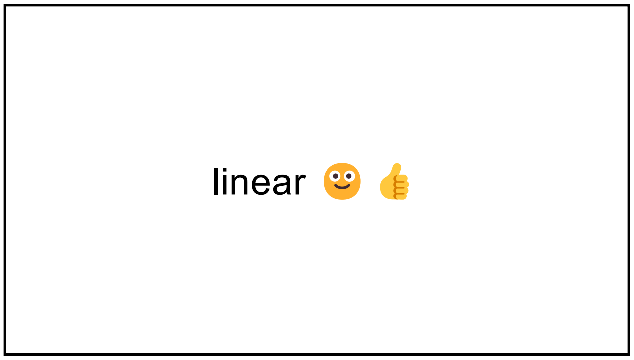


686



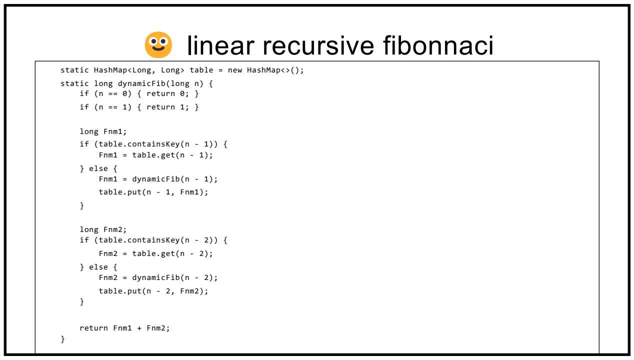
🧠 who's that big O runtime?

687



linear 🙂 👍

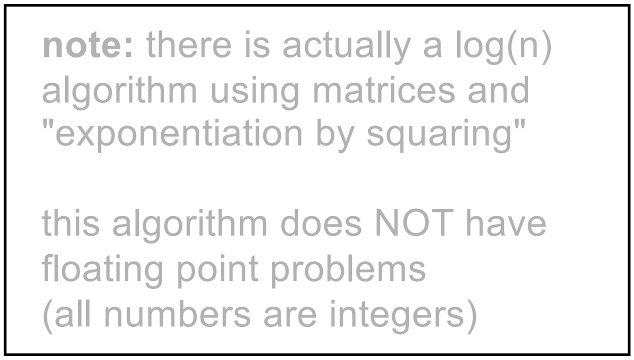
688



🙂 linear recursive fibonnaci

```
static HashMap<Long, Long> table = new HashMap<>();
static long dynamicFib(long n) {
    if (n == 0) { return 0; }
    if (n == 1) { return 1; }

    long Fnm1;
    if (table.containsKey(n - 1)) {
        Fnm1 = table.get(n - 1);
    } else {
        Fnm1 = dynamicFib(n - 1);
        table.put(n - 1, Fnm1);
    }

    long Fnm2;
    if (table.containsKey(n - 2)) {
        Fnm2 = table.get(n - 2);
    } else {
        Fnm2 = dynamicFib(n - 2);
        table.put(n - 2, Fnm2);
    }

    return Fnm1 + Fnm2;
}
```

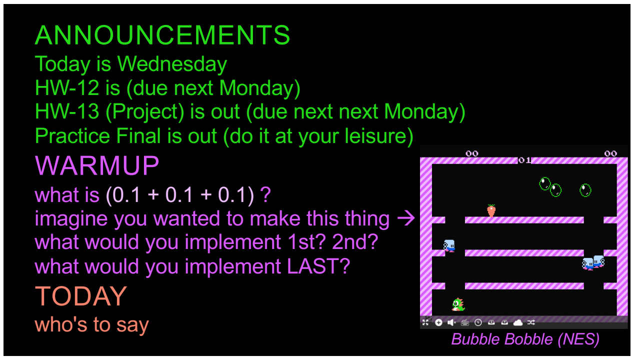
689



note: there is actually a log(n) algorithm using matrices and "exponentiation by squaring"

this algorithm does NOT have floating point problems (all numbers are integers)

690



ANNOUNCEMENTS
Today is Wednesday
HW-12 is (due next Monday)
HW-13 (Project) is out (due next next Monday)
Practice Final is out (do it at your leisure)

WARMUP
what is (0.1 + 0.1 + 0.1) ?
imagine you wanted to make this thing →
what would you implement 1st? 2nd?
what would you implement LAST?

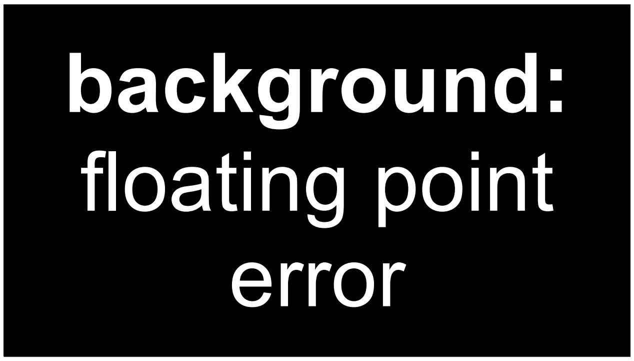
TODAY
who's to say

Bubble Bobble (NES)

691

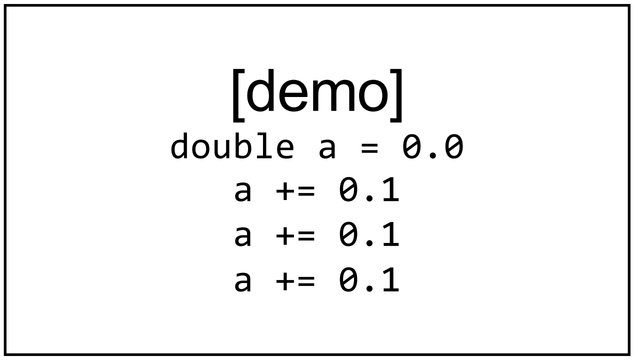
record LEC-02

692



background:
floating point
error

693



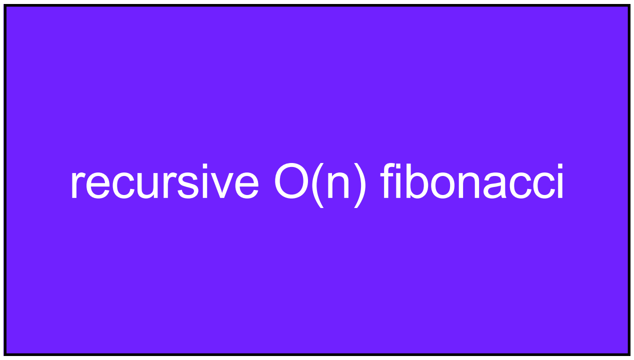
[demo]

```
double a = 0.0
a += 0.1
a += 0.1
a += 0.1
```

694

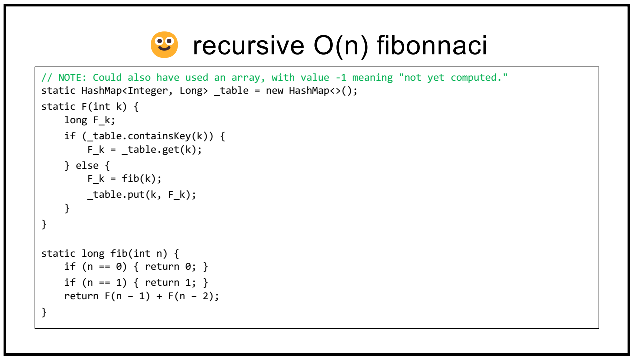
fibonacci wrapup

695



recursive O(n) fibonacci

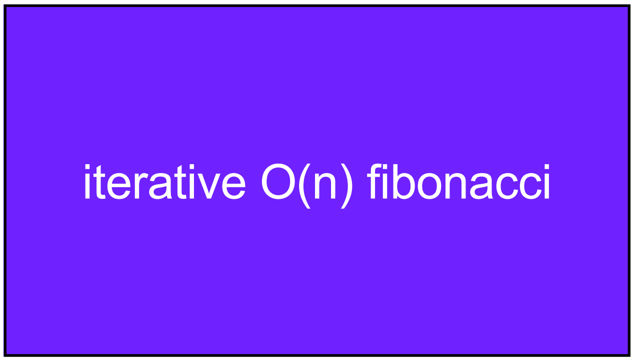
696



recursive O(n) fibonnaci

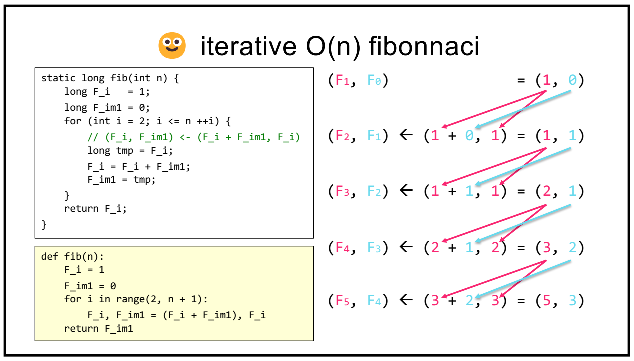
```
// NOTE: Could also have used an array, with value -1 meaning "not yet computed."
static HashMap<Integer, Long> _table = new HashMap<>();
static F(int k) {
    long F_k;
    if (_table.containsKey(k)) {
        F_k = _table.get(k);
    } else {
        F_k = fib(k);
        _table.put(k, F_k);
    }
}

static long fib(int n) {
    if (n == 0) { return 0; }
    if (n == 1) { return 1; }
    return F(n - 1) + F(n - 2);
}
```

697



iterative O(n) fibonacci

698



iterative O(n) fibonnaci

```
static long fib(int n) {
    long F_i   = 1;
    long F_im1 = 0;
    for (int i = 2; i <= n ++i) {
        // (F_i, F_im1) <- (F_i + F_im1, F_i)
        long tmp = F_i;
        F_i = F_i + F_im1;
        F_im1 = tmp;
    }
    return F_i;
}
```

```
def fib(n):
    F_i = 1
    F_im1 = 0
    for i in range(2, n + 1):
        F_i, F_im1 = (F_i + F_im1), F_i
    return F_im1
```

$(F_1, F_0) = (1, 0)$

$(F_2, F_1) \leftarrow (1 + 0, 1) = (1, 1)$

$(F_3, F_2) \leftarrow (1 + 1, 1) = (2, 1)$

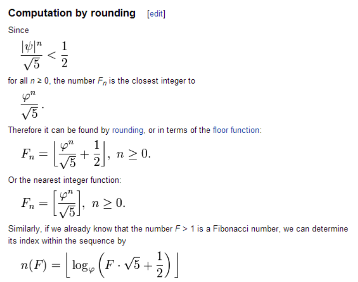
$(F_4, F_3) \leftarrow (2 + 1, 2) = (3, 2)$

$(F_5, F_4) \leftarrow (3 + 2, 3) = (5, 3)$

699

closed form fibonacci

700



Computation by rounding [edit]

Since

$$\frac{|\psi|^n}{\sqrt{5}} < \frac{1}{2}$$

for all $n \geq 0$, the number F_n is the closest integer to

$$\frac{\varphi^n}{\sqrt{5}}.$$

Therefore it can be found by rounding, or in terms of the floor function:

$$F_n = \left\lfloor \frac{\varphi^n}{\sqrt{5}} + \frac{1}{2} \right\rfloor,\ n \geq 0.$$

Or the nearest integer function:

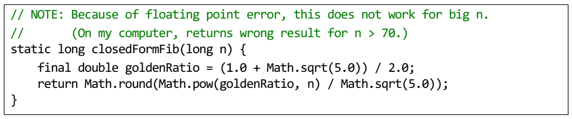
$$F_n = \left[\frac{\varphi^n}{\sqrt{5}} \right],\ n \geq 0.$$

Similarly, if we already know that the number $F > 1$ is a Fibonacci number, we can determine its index within the sequence by

$$n(F) = \left\lfloor \log_\varphi \left(F \cdot \sqrt{5} + \frac{1}{2} \right) \right\rfloor$$

701

approximate closed-form fibonnaci



```
// NOTE: Because of floating point error, this does not work for big n.
//       (On my computer, returns wrong result for n > 70.)
static long closedFormFib(long n) {
    final double goldenRatio = (1.0 + Math.sqrt(5.0)) / 2.0;
    return Math.round(Math.pow(goldenRatio, n) / Math.sqrt(5.0));
}
```

702

recursion example

subset sum

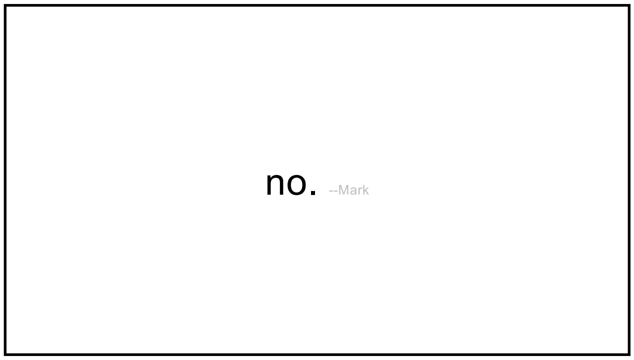
703

problem overview

704

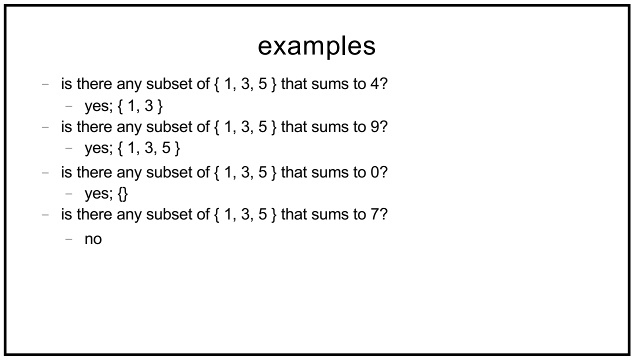
given a finite set of numbers $\{a, b, c, \ldots\}$,
is there **any** subset that sums to target T?

705



no. --Mark

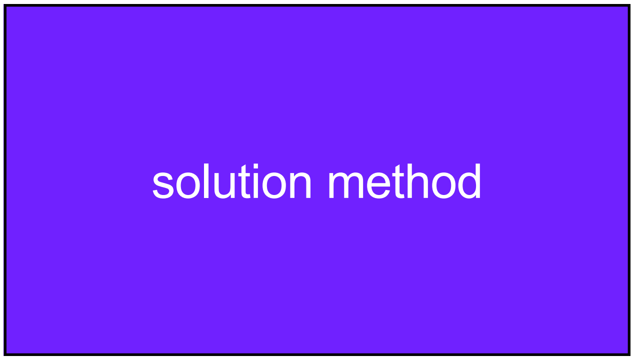
706



examples

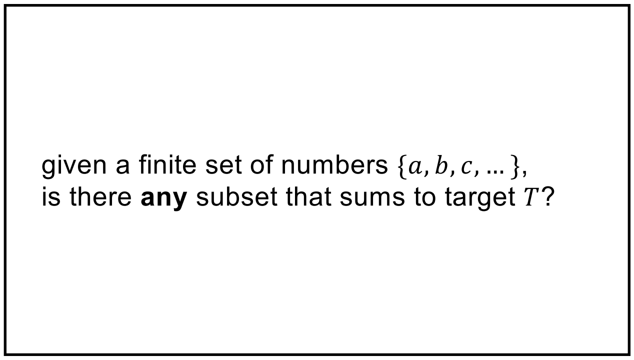
- is there any subset of { 1, 3, 5 } that sums to 4?
  - yes; { 1, 3 }
- is there any subset of { 1, 3, 5 } that sums to 9?
  - yes; { 1, 3, 5 }
- is there any subset of { 1, 3, 5 } that sums to 0?
  - yes; {}
- is there any subset of { 1, 3, 5 } that sums to 7?
  - no

707



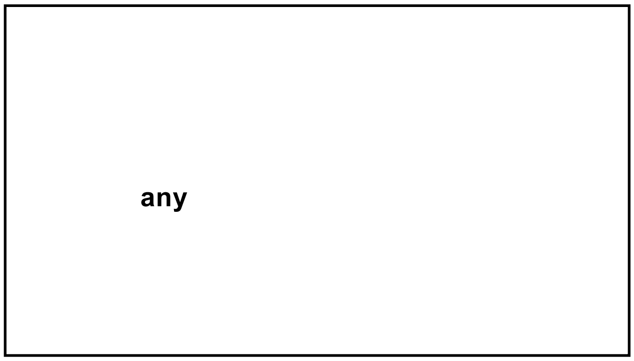
solution method

708



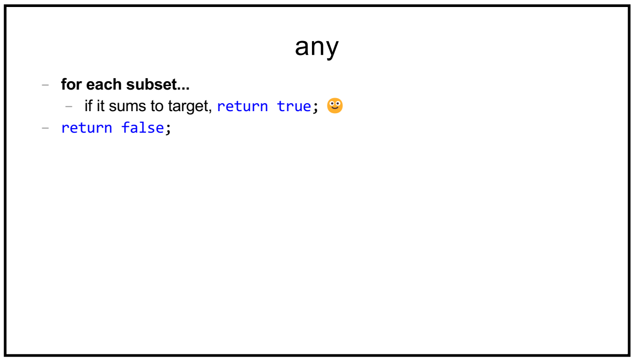
given a finite set of numbers $\{a, b, c, \dots\}$,
is there **any** subset that sums to target T?

709



any

710



any

- **for each subset...**
  - if it sums to target, `return true;` 🙂
- `return false;`

711

🧠 why is this maybe not so easy?

"hard" means "hard for a computer to solve quickly"

what are the subsets of $\{ 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$?

∅ | {1} | {2} | {3} | {4} | {5} | {6} | {7} | {8} | {9} | {1, 2} | {1, 3} | {1, 4} | {1, 5} | {1, 6} | {1, 7} | {1, 8} | {1, 9} | {2, 3} | {2, 4} | {2, 5} | {2, 6} | {2, 7} | {2, 8} | {2, 9} | {3, 4} | {3, 5} | {3, 6} | {3, 7} | {3, 8} | {3, 9} | {4, 5} | {4, 6} | {4, 7} | {4, 8} | {4, 9} | {5, 6} | {5, 7} | {5, 8} | {5, 9} | {6, 7} | {6, 8} | {6, 9} | {7, 8} | {7, 9} | {8, 9} | {1, 2, 3} | {1, 2, 4} | {1, 2, 5} | {1, 2, 6} | {1, 2, 7} | {1, 2, 8} | {1, 2, 9} | {1, 3, 4} | {1, 3, 5} | {1, 3, 6} | {1, 3, 7} | {1, 3, 8} | {1, 3, 9} | {1, 4, 5} | {1, 4, 6} | {1, 4, 7} | {1, 4, 8} | {1, 4, 9} | {1, 5, 6} | {1, 5, 7} | {1, 5, 8} | {1, 5, 9} | {1, 6, 7} | {1, 6, 8} | {1, 6, 9} | {1, 7, 8} | {1, 7, 9} | {1, 8, 9} | {2, 3, 4} | {2, 3, 5} | {2, 3, 6} | {2, 3, 7} | {2, 3, 8} | {2, 3, 9} | {2, 4, 5} | {2, 4, 6} | {2, 4, 7} | {2, 4, 8} | {2, 4, 9} | {2, 5, 6} | {2, 5, 7} | {2, 5, 8} | {2, 5, 9} | {2, 6, 7} | {2, 6, 8} | {2, 6, 9} | {2, 7, 8} | {2, 7, 9} | {2, 8, 9} | {3, 4, 5} | {3, 4, 6} | {3, 4, 7} | {3, 4, 8} | {3, 4, 9} | {3, 5, 6} | {3, 5, 7} | {3, 5, 8} | {3, 5, 9} | {3, 6, 7} | {3, 6, 8} | {3, 6, 9} | {3, 7, 8} | {3, 7, 9} | {3, 8, 9} | {4, 5, 6} | {4, 5, 7} | {4, 5, 8} | {4, 5, 9} | {4, 6, 7} | {4, 6, 8} | {4, 6, 9} | {4, 7, 8} | {4, 7, 9} | {4, 8, 9} | {5, 6, 7} | {5, 6, 8} | {5, 6, 9} | {5, 7, 8} | {5, 7, 9} | {5, 8, 9} | {6, 7, 8} | {6, 7, 9} | {6, 8, 9} | {7, 8, 9} | {1, 2, 3, 4} | {1, 2, 3, 5} | {1, 2, 3, 6} | {1, 2, 3, 7} | {1, 2, 3, 8} | {1, 2, 3, 9} | {1, 2, 4, 5} | {1, 2, 4, 6} | {1, 2, 4, 7} | {1, 2, 4, 8} | {1, 2, 4, 9} | {1, 2, 5, 6} | {1, 2, 5, 7} | {1, 2, 5, 8} | {1, 2, 5, 9} | {1, 2, 6, 7} | {1, 2, 6, 8} | {1, 2, 6, 9} | {1, 2, 7, 8} | {1, 2, 7, 9} | {1, 2, 8, 9} | {1, 3, 4, 5} | {1, 3, 4, 6} | {1, 3, 4, 7} | {1, 3, 4, 8} | {1, 3, 4, 9} | {1, 3, 5, 6} | {1, 3, 5, 7} | {1, 3, 5, 8} | {1, 3, 5, 9} | {1, 3, 6, 7} | {1, 3, 6, 8} | {1, 3, 6, 9} | {1, 3, 7, 8} | {1, 3, 7, 9} | {1, 3, 8, 9} | {1, 4, 5, 6} | {1, 4, 5, 7} | {1, 4, 5, 8} | {1, 4, 5, 9} | {1, 4, 6, 7} | {1, 4, 6, 8} | {1, 4, 6, 9} | {1, 4, 7, 8} | {1, 4, 7, 9} | {1, 4, 8, 9} | {1, 5, 6, 7} | {1, 5, 6, 8} | {1, 5, 6, 9} | {1, 5, 7, 8} | {1, 5, 7, 9} | {1, 5, 8, 9} | {1, 6, 7, 8} | {1, 6, 7, 9} | {1, 6, 8, 9} | {1, 7, 8, 9} | {2, 3, 4, 5} | {2, 3, 4, 6} | {2, 3, 4, 7} | {2, 3, 4, 8} | {2, 3, 4, 9} | {2, 3, 5, 6} | {2, 3, 5, 7} | {2, 3, 5, 8} | {2, 3, 5, 9} | {2, 3, 6, 7} | {2, 3, 6, 8} | {2, 3, 6, 9} | {2, 3, 7, 8} | {2, 3, 7, 9} | {2, 3, 8, 9} | {2, 4, 5, 6} | {2, 4, 5, 7} | {2, 4, 5, 8} | {2, 4, 5, 9} | {2, 4, 6, 7} | {2, 4, 6, 8} | {2, 4, 6, 9} | {2, 4, 7, 8} | {2, 4, 7, 9} | {2, 4, 8, 9} | {2, 5, 6, 7} | {2, 5, 6, 8} | {2, 5, 6, 9} | {2, 5, 7, 8} | {2, 5, 7, 9} | {2, 5, 8, 9} | {2, 6, 7, 8} | {2, 6, 7, 9} | {2, 6, 8, 9} | {2, 7, 8, 9} | {3, 4, 5, 6} | {3, 4, 5, 7} | {3, 4, 5, 8} | {3, 4, 5, 9} | {3, 4, 6, 7} | {3, 4, 6, 8} | {3, 4, 6, 9} | {3, 4, 7, 8} | {3, 4, 7, 9} | {3, 4, 8, 9} | {3, 5, 6, 7} | {3, 5, 6, 8} | {3, 5, 6, 9} | {3, 5, 7, 8} | {3, 5, 7, 9} | {3, 5, 8, 9} | {3, 6, 7, 8} | {3, 6, 7, 9} | {3, 6, 8, 9} | {3, 7, 8, 9} | {4, 5, 6, 7} | {4, 5, 6, 8} | {4, 5, 6, 9} | {4, 5, 7, 8} | {4, 5, 7, 9} | {4, 5, 8, 9} | {4, 6, 7, 8} | {4, 6, 7, 9} | {4, 6, 8, 9} | {4, 7, 8, 9} | {5, 6, 7, 8} | {5, 6, 7, 9} | {5, 6, 8, 9} | {5, 7, 8, 9} | {6, 7, 8, 9} | {1, 2, 3, 4, 5} | {1, 2, 3, 4, 6} | {1, 2, 3, 4, 7} | {1, 2, 3, 4, 8} | {1, 2, 3, 4, 9} | {1, 2, 3, 5, 6} | {1, 2, 3, 5, 7} | {1, 2, 3, 5, 8} | {1, 2, 3, 5, 9} | {1, 2, 3, 6, 7} | {1, 2, 3, 6, 8} | {1, 2, 3, 6, 9} | {1, 2, 3, 7, 8} | {1, 2, 3, 7, 9} | {1, 2, 3, 8, 9} | {1, 2, 4, 5, 6} | {1, 2, 4, 5, 7} | {1, 2, 4, 5, 8} | {1, 2, 4, 5, 9} | {1, 2, 4, 6, 7} | {1, 2, 4, 6, 8} | {1, 2, 4, 6, 9} | {1, 2, 4, 7, 8} | {1, 2, 4, 7, 9} | {1, 2, 4, 8, 9} | {1, 2, 5, 6, 7} | {1, 2, 5, 6, 8} | {1, 2, 5, 6, 9} | {1, 2, 5, 7, 8} | {1, 2, 5, 7, 9} | {1, 2, 5, 8, 9} | {1, 2, 6, 7, 8} | {1, 2, 6, 7, 9} | {1, 2, 6, 8, 9} | {1, 2, 7, 8, 9} | {1, 3, 4, 5, 6} | {1, 3, 4, 5, 7} | {1, 3, 4, 5, 8} | {1, 3, 4, 5, 9} | {1, 3, 4, 6, 7} | {1, 3, 4, 6, 8} | {1, 3, 4, 6, 9} | {1, 3, 4, 7, 8} | {1, 3, 4, 7, 9} | {1, 3, 4, 8, 9} | {1, 3, 5, 6, 7} | {1, 3, 5, 6, 8} | {1, 3, 5, 6, 9} | {1, 3, 5, 7, 8} | {1, 3, 5, 7, 9} | {1, 3, 5, 8, 9} | {1, 3, 6, 7, 8} | {1, 3, 6, 7, 9} | {1, 3, 6, 8, 9} | {1, 3, 7, 8, 9} | {1, 4, 5, 6, 7} | {1, 4, 5, 6, 8} | {1, 4, 5, 6, 9} | {1, 4, 5, 7, 8} | {1, 4, 5, 7, 9} | {1, 4, 5, 8, 9} | {1, 4, 6, 7, 8} | {1, 4, 6, 7, 9} | {1, 4, 6, 8, 9} | {1, 4, 7, 8, 9} | {1, 5, 6, 7, 8} | {1, 5, 6, 7, 9} | {1, 5, 6, 8, 9} | {1, 5, 7, 8, 9} | {1, 6, 7, 8, 9} | {2, 3, 4, 5, 6} | {2, 3, 4, 5, 7} | {2, 3, 4, 5, 8} | {2, 3, 4, 5, 9} | {2, 3, 4, 6, 7} | {2, 3, 4, 6, 8} | {2, 3, 4, 6, 9} | {2, 3, 4, 7, 8} | {2, 3, 4, 7, 9} | {2, 3, 4, 8, 9} | {2, 3, 5, 6, 7} | {2, 3, 5, 6, 8} | {2, 3, 5, 6, 9} | {2, 3, 5, 7, 8} | {2, 3, 5, 7, 9} | {2, 3, 5, 8, 9} | {2, 3, 6, 7, 8} | {2, 3, 6, 7, 9} | {2, 3, 6, 8, 9} | {2, 3, 7, 8, 9} | {2, 4, 5, 6, 7} | {2, 4, 5, 6, 8} | {2, 4, 5, 6, 9} | {2, 4, 5, 7, 8} | {2, 4, 5, 7, 9} | {2, 4, 5, 8, 9} | {2, 4, 6, 7, 8} | {2, 4, 6, 7, 9} | {2, 4, 6, 8, 9} | {2, 4, 7, 8, 9} | {2, 5, 6, 7, 8} | {2, 5, 6, 7, 9} | {2, 5, 6, 8, 9} | {2, 5, 7, 8, 9} | {2, 6, 7, 8, 9} | {3, 4, 5, 6, 7} | {3, 4, 5, 6, 8} | {3, 4, 5, 6, 9} | {3, 4, 5, 7, 8} | {3, 4, 5, 7, 9} | {3, 4, 5, 8, 9} | {3, 4, 6, 7, 8} | {3, 4, 6, 7, 9} | {3, 4, 6, 8, 9} | {3, 4, 7, 8, 9} | {3, 5, 6, 7, 8} | {3, 5, 6, 7, 9} | {3, 5, 6, 8, 9} | {3, 5, 7, 8, 9} | {3, 6, 7, 8, 9} | {4, 5, 6, 7, 8} | {4, 5, 6, 7, 9} | {4, 5, 6, 8, 9} | {4, 5, 7, 8, 9} | {4, 6, 7, 8, 9} | {5, 6, 7, 8, 9} | {1, 2, 3, 4, 5, 6} | {1, 2, 3, 4, 5, 7} | {1, 2, 3, 4, 5, 8} | {1, 2, 3, 4, 5, 9} | {1, 2, 3, 4, 6, 7} | {1, 2, 3, 4, 6, 8} | {1, 2, 3, 4, 6, 9} | {1, 2, 3, 4, 7, 8} | {1, 2, 3, 4, 7, 9} | {1, 2, 3, 4, 8, 9} | {1, 2, 3, 5, 6, 7} | {1, 2, 3, 5, 6, 8} | {1, 2, 3, 5, 6, 9} | {1, 2, 3, 5, 7, 8} | {1, 2, 3, 5, 7, 9} | {1, 2, 3, 5, 8, 9} | {1, 2, 3, 6, 7, 8} | {1, 2, 3, 6, 7, 9} | {1, 2, 3, 6, 8, 9} | {1, 2, 3, 7, 8, 9} | {1, 2, 4, 5, 6, 7} | {1, 2, 4, 5, 6, 8} | {1, 2, 4, 5, 6, 9} | {1, 2, 4, 5, 7, 8} | {1, 2, 4, 5, 7, 9} | {1, 2, 4, 5, 8, 9} | {1, 2, 4, 6, 7, 8} | {1, 2, 4, 6, 7, 9} | {1, 2, 4, 6, 8, 9} | {1, 2, 4, 7, 8, 9} | {1, 2, 5, 6, 7, 8} | {1, 2, 5, 6, 7, 9} | {1, 2, 5, 6, 8, 9} | {1, 2, 5, 7, 8, 9} | {1, 2, 6, 7, 8, 9} | {1, 3, 4, 5, 6, 7} | {1, 3, 4, 5, 6, 8} | {1, 3, 4, 5, 6, 9} | {1, 3, 4, 5, 7, 8} | {1, 3, 4, 5, 7, 9} | {1, 3, 4, 5, 8, 9} | {1, 3, 4, 6, 7, 8} | {1, 3, 4, 6, 7, 9} | {1, 3, 4, 6, 8, 9} | {1, 3, 4, 7, 8, 9} | {1, 3, 5, 6, 7, 8} | {1, 3, 5, 6, 7, 9} | {1, 3, 5, 6, 8, 9} | {1, 3, 5, 7, 8, 9} | {1, 3, 6, 7, 8, 9} | {1, 4, 5, 6, 7, 8} | {1, 4, 5, 6, 7, 9} | {1, 4, 5, 6, 8, 9} | {1, 4, 5, 7, 8, 9} | {1, 4, 6, 7, 8, 9} | {1, 5, 6, 7, 8, 9} | {2, 3, 4, 5, 6, 7} | {2, 3, 4, 5, 6, 8} | {2, 3, 4, 5, 6, 9} | {2, 3, 4, 5, 7, 8} | {2, 3, 4, 5, 7, 9} | {2, 3, 4, 5, 8, 9} | {2, 3, 4, 6, 7, 8} | {2, 3, 4, 6, 7, 9} | {2, 3, 4, 6, 8, 9} | {2, 3, 4, 7, 8, 9} | {2, 3, 5, 6, 7, 8} | {2, 3, 5, 6, 7, 9} | {2, 3, 5, 6, 8, 9} | {2, 3, 5, 7, 8, 9} | {2, 3, 6, 7, 8, 9} | {2, 4, 5, 6, 7, 8} | {2, 4, 5, 6, 7, 9} | {2, 4, 5, 6, 8, 9} | {2, 4, 5, 7, 8, 9} | {2, 4, 6, 7, 8, 9} | {2, 5, 6, 7, 8, 9} | {3, 4, 5, 6, 7, 8} | {3, 4, 5, 6, 7, 9} | {3, 4, 5, 6, 8, 9} | {3, 4, 5, 7, 8, 9} | {3, 4, 6, 7, 8, 9} | {3, 5, 6, 7, 8, 9} | {4, 5, 6, 7, 8, 9} | {1, 2, 3, 4, 5, 6, 7} | {1, 2, 3, 4, 5, 6, 8} | {1, 2, 3, 4, 5, 6, 9} | {1, 2, 3, 4, 5, 7, 8} | {1, 2, 3, 4, 5, 7, 9} | {1, 2, 3, 4, 5, 8, 9} | {1, 2, 3, 4, 6, 7, 8} | {1, 2, 3, 4, 6, 7, 9} | {1, 2, 3, 4, 6, 8, 9} | {1, 2, 3, 4, 7, 8, 9} | {1, 2, 3, 5, 6, 7, 8} | {1, 2, 3, 5, 6, 7, 9} | {1, 2, 3, 5, 6, 8, 9} | {1, 2, 3, 5, 7, 8, 9} | {1, 2, 3, 6, 7, 8, 9} | {1, 2, 4, 5, 6, 7, 8} | {1, 2, 4, 5, 6, 7, 9} | {1, 2, 4, 5, 6, 8, 9} | {1, 2, 4, 5, 7, 8, 9} | {1, 2, 4, 6, 7, 8, 9} | {1, 2, 5, 6, 7, 8, 9} | {1, 3, 4, 5, 6, 7, 8} | {1, 3, 4, 5, 6, 7, 9} | {1, 3, 4, 5, 6, 8, 9} | {1, 3, 4, 5, 7, 8, 9} | {1, 3, 4, 6, 7, 8, 9} | {1, 3, 5, 6, 7, 8, 9} | {1, 4, 5, 6, 7, 8, 9} | {2, 3, 4, 5, 6, 7, 8} | {2, 3, 4, 5, 6, 7, 9} | {2, 3, 4, 5, 6, 8, 9} | {2, 3, 4, 5, 7, 8, 9} | {2, 3, 4, 6, 7, 8, 9} | {2, 3, 5, 6, 7, 8, 9} | {2, 4, 5, 6, 7, 8, 9} | {3, 4, 5, 6, 7, 8, 9} | {1, 2, 3, 4, 5, 6, 7, 8} | {1, 2, 3, 4, 5, 6, 7, 9} | {1, 2, 3, 4, 5, 6, 8, 9} | {1, 2, 3, 4, 5, 7, 8, 9} | {1, 2, 3, 4, 6, 7, 8, 9} | {1, 2, 3, 5, 6, 7, 8, 9} | {1, 2, 4, 5, 6, 7, 8, 9} | {1, 3, 4, 5, 6, 7, 8, 9} | {2, 3, 4, 5, 6, 7, 8, 9} | {1, 2, 3, 4, 5, 6, 7, 8, 9}

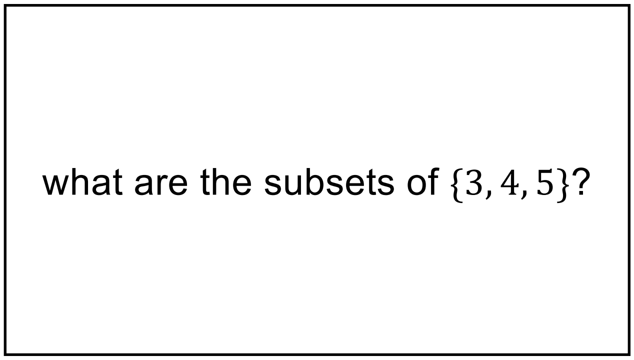
there are 2^9 of them

🧠 why?

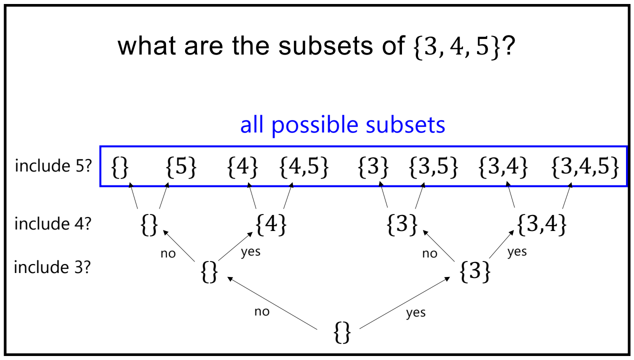
each of the 9 elements is either included or not included (excluded) in the subset

9 include/exclude decisions => 2^9

718


what are the subsets of $\{3,4,5\}$?

719


what are the subsets of $\{3,4,5\}$?

all possible subsets

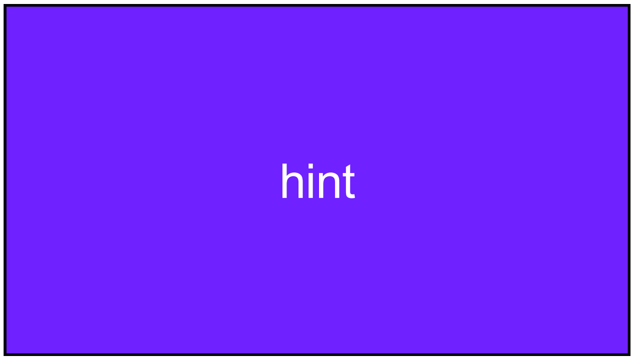
include 5? {} {5} {4} {4,5} {3} {3,5} {3,4} {3,4,5}

include 4? {} {4} {3} {3,4}

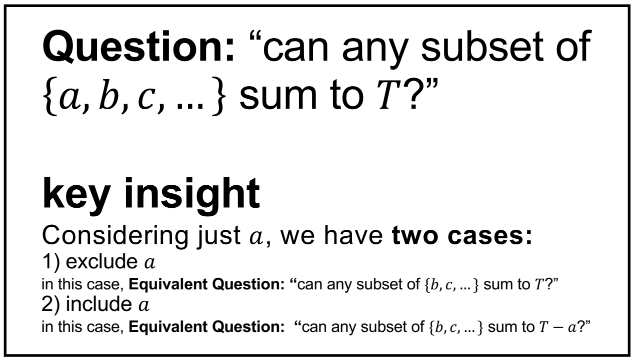
include 3? {} {3}

{}

720


hint

721


Question: “can any subset of $\{a, b, c, \ldots\}$ sum to T?”

key insight

Considering just a, we have **two cases:**

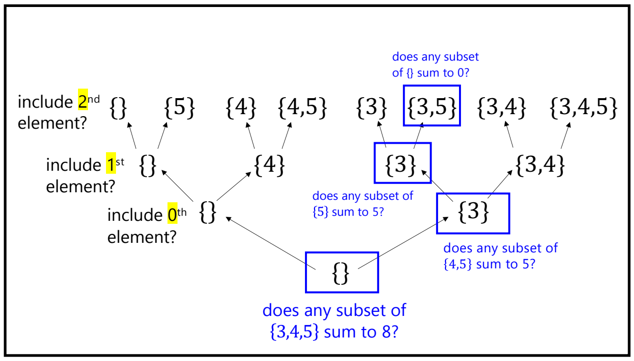
1) exclude a

in this case, **Equivalent Question:** “can any subset of $\{b, c, \ldots\}$ sum to T?”

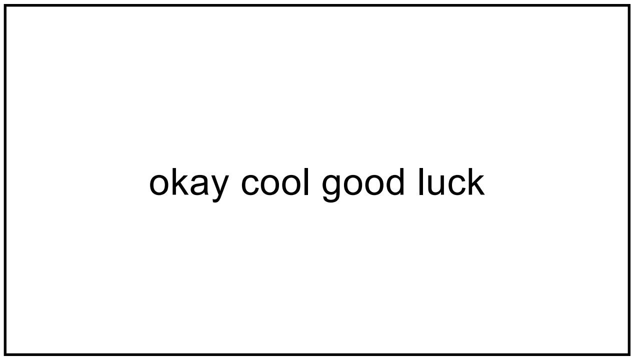
2) include a

in this case, **Equivalent Question:** “can any subset of $\{b, c, \ldots\}$ sum to $T - a$?”

722


does any subset of {} sum to 0?
include 2nd element? {} {5} {4} {4,5} {3} {3,5} {3,4} {3,4,5}
include 1st element? {} {4} {3} {3,4}
does any subset of {5} sum to 5?
include 0th element? {} {3}
does any subset of {4,5} sum to 5?
{}
does any subset of {3,4,5} sum to 8?

723



okay cool good luck

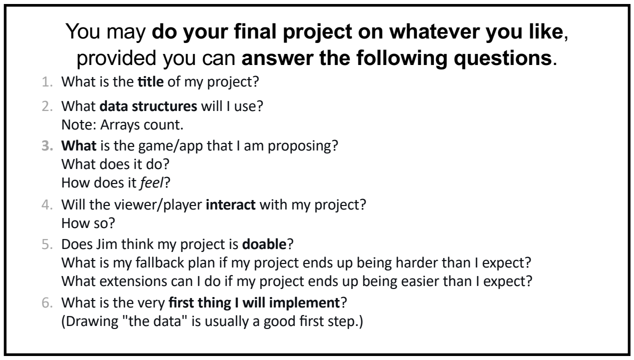
724

final project

727

final project

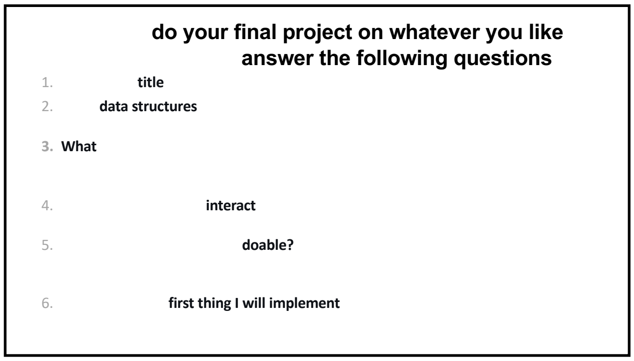
728



You may **do your final project on whatever you like**, provided you can **answer the following questions**.

1. What is the **title** of my project?
2. What **data structures** will I use?
   Note: Arrays count.
3. **What** is the game/app that I am proposing?
   What does it do?
   How does it *feel*?
4. Will the viewer/player **interact** with my project?
   How so?
5. Does Jim think my project is **doable**?
   What is my fallback plan if my project ends up being harder than I expect?
   What extensions can I do if my project ends up being easier than I expect?
6. What is the very **first thing I will implement**?
   (Drawing "the data" is usually a good first step.)

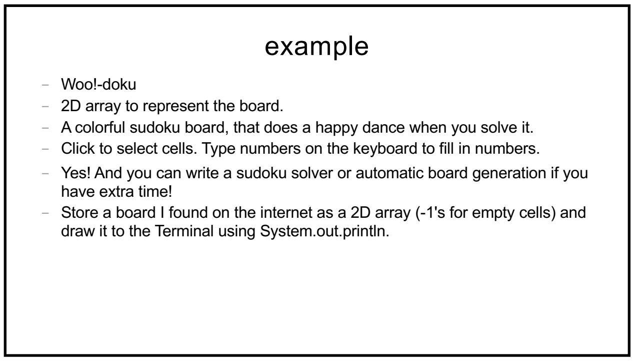
729



do your final project on whatever you like
answer the following questions

1. **title**
2. **data structures**
3. **What**
4. **interact**
5. **doable?**
6. **first thing I will implement**

730



example

- Woo!-doku
- 2D array to represent the board.
- A colorful sudoku board, that does a happy dance when you solve it.
- Click to select cells. Type numbers on the keyboard to fill in numbers.
- Yes! And you can write a sudoku solver or automatic board generation if you have extra time!
- Store a board I found on the internet as a 2D array (-1's for empty cells) and draw it to the Terminal using System.out.println.

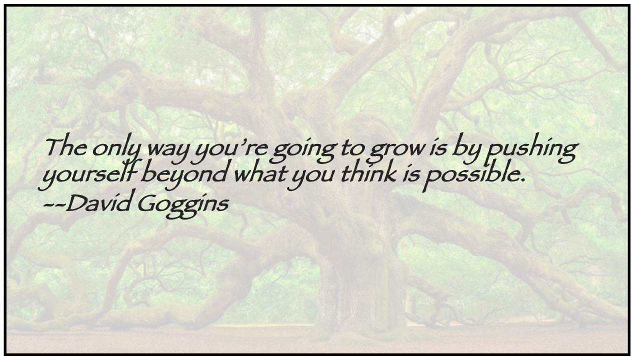
731

how are we feeling?

732

why am i making you make a thing

733



The only way you're going to grow is by pushing yourself beyond what you think is possible.
--David Goggins

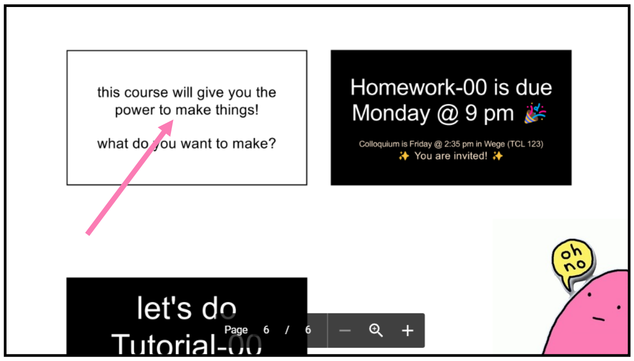
734

This ~~season~~ semester we're upping our intensity!
--Alice Lacrosse Coach

735

also you were warned 😳

736



this course will give you the power to make things!

what do you want to make?

Homework-00 is due Monday @ 9 pm 🎉

Colloquium is Friday @ 2:35 pm in Wege (TCL 123)
✨ You are invited! ✨

let's do Tutorial-00

737

how to
make a thing

738

how not to
make a thing

739

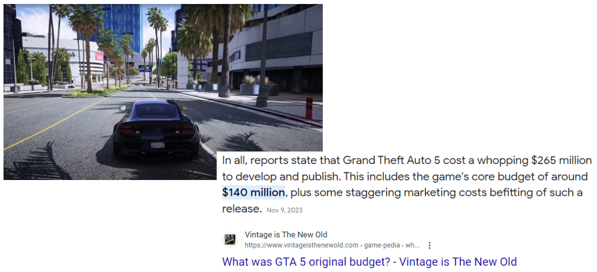
note: this advice is like...just advice
feel free to ignore (at your own peril) 🙃

740

project selection

741

⚠️ picking something really, really hard



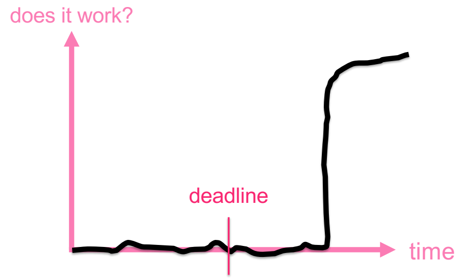
In all, reports state that Grand Theft Auto 5 cost a whopping $265 million to develop and publish. This includes the game's core budget of around **$140 million**, plus some staggering marketing costs befitting of such a release. Nov 9, 2023

Vintage is The New Old
https://www.vintageisthenewold.com › game-pedia › wh...

What was GTA 5 original budget? - Vintage is The New Old

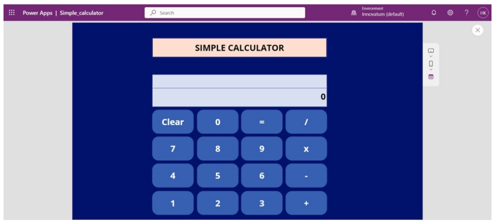
742

⚠️ picking something with a spooky progress curve



743

⚠️ picking something you aren't really interested in
(or can't really be **extended**)



note: though if you love calculators, go for it!

744

who (doesn't) have a final project idea?

discuss amongst each other

745

focus

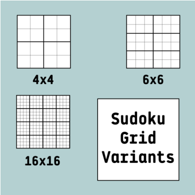
(just make sudoku)

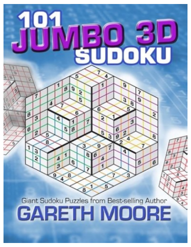
746

⚠️ solving a more general problem than you need to

don't support arbitrary board sizes!

just make sudoku!





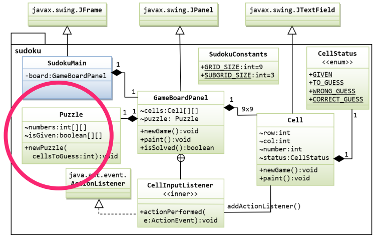
what?!

747

⚠️ writing more general code than you need to

don't support multiple different kinds of puzzles

just make sudoku!



748

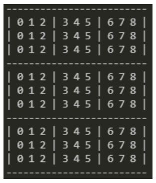
⚠️ doing the fun part first

don't make a fun animated menu screen

just make sudoku!

749

⚠ skipping development steps



```
-------------------------
| 0 1 2 | 3 4 5 | 6 7 8 |
| 0 1 2 | 3 4 5 | 6 7 8 |
| 0 1 2 | 3 4 5 | 6 7 8 |
-------------------------
| 0 1 2 | 3 4 5 | 6 7 8 |
| 0 1 2 | 3 4 5 | 6 7 8 |
| 0 1 2 | 3 4 5 | 6 7 8 |
-------------------------
| 0 1 2 | 3 4 5 | 6 7 8 |
| 0 1 2 | 3 4 5 | 6 7 8 |
| 0 1 2 | 3 4 5 | 6 7 8 |
-------------------------
```

consider making a board print to the terminal
before trying to get it working in Cow.java

consider just filling it with nonsense
before filling it with a real board

consider hardcoding a starting board by hand
before implementing automatic board generation

750

then again, sometimes you just gotta go for it.

YOLO!

751

who (doesn't) know what they're going to implement first?

discuss amongst each other

752

final thoughts

753

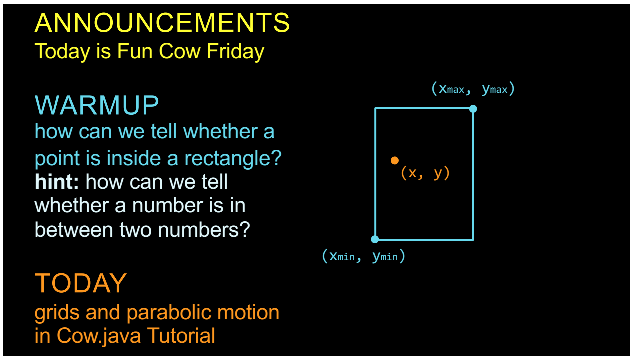
final thoughts

- don't be afraid to write code
- don't be afraid to delete code
- don't be (too) afraid to fail
- you will be graded primarily on effort

754

how are we feeling?

755



ANNOUNCEMENTS
Today is Fun Cow Friday

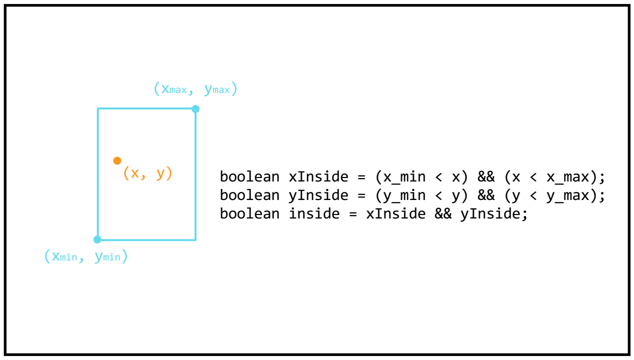
WARMUP
how can we tell whether a point is inside a rectangle?
hint: how can we tell whether a number is in between two numbers?

TODAY
grids and parabolic motion in Cow.java Tutorial

756

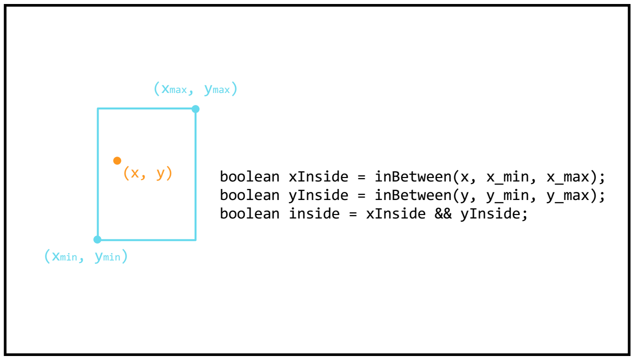
record LEC-02

757



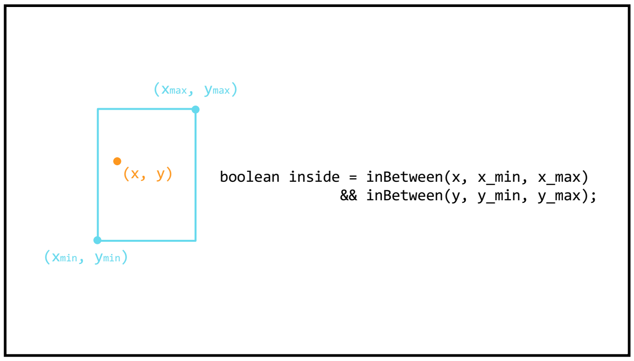
```
boolean xInside = (x_min < x) && (x < x_max);
boolean yInside = (y_min < y) && (y < y_max);
boolean inside = xInside && yInside;
```

758



```
boolean xInside = inBetween(x, x_min, x_max);
boolean yInside = inBetween(y, y_min, y_max);
boolean inside = xInside && yInside;
```

759

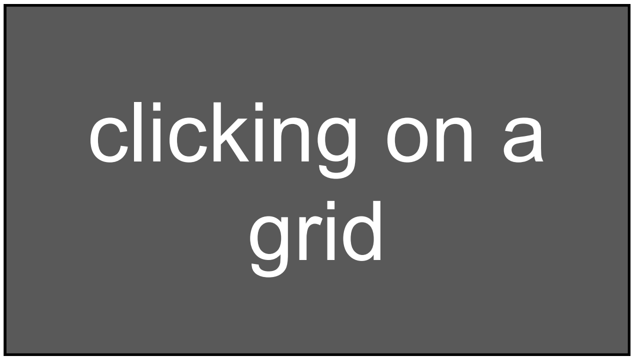


```
boolean inside = inBetween(x, x_min, x_max)
              && inBetween(y, y_min, y_max);
```

760

let's do it in cow!

761

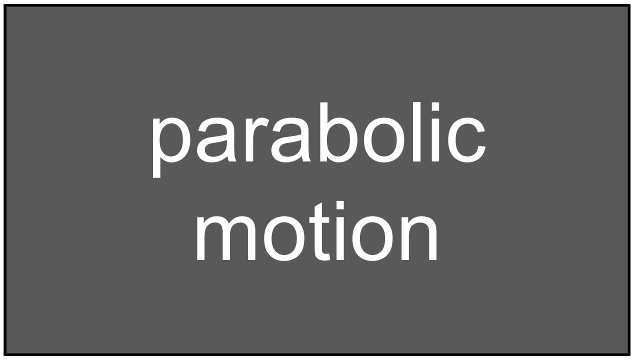


clicking on a grid

762

cat game

763



parabolic motion

764