“Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time: **premature optimization is the root of all evil**. Yet we should not pass up our opportunities in that critical 3%."

—Don Knuth

Write the tabulation template for fib

Name

Th. 11/8

(your response)

make-change, infinite coins

Given a value and a set of coins, what is the minimum number of coins required to sum to the value, assuming we have an infinite number of each coin?

longest-common substring (LCS)

How similar are these strings?

The longest-common substring of **s1** and **s2** is the longest string that is a *non-consecutive* substring of both **s1** and **s2**.

```
lcs('x', 'y') == 0        lcs('car', 'cat') == 2

lcs('x', '') == 0         lcs('human', 'chimpanzee') == 4

lcs('', 'x') == 0
```

Theoretical tools: code → math

How many times does the platypus quack?

```
platypus.quack()
```

Theoretical tools: code → math

How many times does the platypus quack?

```
for j in range(N):
    platypus.quack()
```

Theoretical tools: code → math

summations

upper-bound
(inclusive)

(implicit increment)

$$\sum_{i=1}^{5} i = 1 + 2 + 3 + 4 + 5$$

index

lower-bound
(inclusive)

Theoretical tools: code → math

summations

```
for j in range(N):
    platypus.quack()
```

$$\sum_{j=0}^{N-1} 1$$

Theoretical tools: code → math

summations

upper-bound (inclusive)

(implicit increment)

$$\sum_{j=0}^{N-1} 1$$

index

lower-bound (inclusive)

j	cost
0	1
1	1
2	1
3	1
...	...
N-2	1
N-1	1

N ∈O(N)

Theoretical tools: code → math

summations

$$\sum_{i=1}^{N} 1$$

i	cost
1	1
2	1
3	1
4	1
...	...
N-1	1
N	1

$N \in O(N)$

Theoretical tools: code → math

summations

$$\sum_{i=1}^{N} N$$

i	cost
1	N
2	N
3	N
4	N
...	...
N-1	N
N	N

$N^2 \in O(N^2)$

Theoretical tools: code → math

summations

$$\sum_{i=1}^{N} i$$

i	cost
1	1
2	2
3	3
4	4
...	...
N-1	N-1
N	N

$\frac{N(N+1)}{2} \in O(N^2)$

Theoretical tools: code → math

How many times does the platypus quack?

code	`for j in range(N):` `    platypus.quack()`
math	$\sum_{j=0}^{N-1} 1$
Wolfram Alpha	sum 1,j=0 to N-1
closed form	N
asymptotic notation	$O(N)$

Theoretical tools: code → math

How many times does the platypus quack?

```
for i in range(N):
  for j in range(N):
     platypus.quack()
```

Theoretical tools: code → math

How many times does the platypus quack?

code

```
for i in range(N):
  for j in range(N):
     platypus.quack()
```

math

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} 1$$

Wolfram Alpha

sum (sum 1,j=0 to N-1),i=0 to N-1

closed form

$$N^2$$

asymptotic notation

$$O(N^2)$$

Theoretical tools: code → math

How many times does the platypus quack?

```
for i in range(N):
  for j in range(i, N):
     platypus.quack()
```

Theoretical tools: code → math

How many times does the platypus quack?

code

```
for i in range(N):
  for j in range(i, N):
    platypus.quack()
```

math

$$\sum_{i=0}^{N-1} \sum_{j=i}^{N-1} 1$$

Wolfram Alpha

sum (sum 1,j=i to N-1),i=0 to N-1

closed form

$$\frac{N(N+1)}{2}$$

asymptotic notation

$$O(N^2)$$