Math 182 Lecture 2

§1.3 Triangular Number Algorithms

```
TRIANGLE(n)
1  Sum = 0
2  // Initializes Sum to 0
3  for j = 0 to n
4      Sum = Sum + j
5      // Replaces the current value of Sum with Sum + j
6      // This has the effect of adding j to Sum
7  return Sum
```

start of iteration

value of Sum

$$\sum_{k=0}^{n} k = 0 + 1 + \cdots + n$$

Why is Triangle correct?

Intuitively: each time one iteration of for-loop finishes, value of sum is $\sum_{k=0}^{j} k$.

Sum = 0, $\sum_{i=0}^{0} i$, $\sum_{i=0}^{1} i$, $\sum_{i=0}^{2} i$, ..., $\sum_{i=0}^{n} i$

Correctness of algorithm

(Loop invariant for TRIANGLE) At the start of each iteration of the **for** loop on lines 3-6 the value of the variable *Sum* is $\sum_{i=0}^{\max(j-1,0)} i$.

want to show this

We need to show 3 things

(1) Initialization: need to show loop invariant true immediately prior to start of first iteration of loop. (≈ base case)

(2) Maintenance: need to show if loop invariant is true before an iteration of loop, then still true after that iteration (≈ inductive step)

(3) Termination After loop terminates, need to show that true loop invariant helps us prove algorithm is correct.

Theorem 1.3.1. *The algorithm* TRIANGLE(n) *outputs the summation* $\sum_{j=0}^{n} j$.

Loop invariant: at start of each iter.
$\text{sum} = \sum_{i=0}^{\max(j-1,0)} i$.

Initialization Before first iteration ✓
sum = 0 and $j = 0$
and $\sum_{i=0}^{\max(j-1,0)} i = \sum_{i=0}^{\max(-1,0)} i = \sum_{i=0}^{0} i = 0$

Maintenance Assume loop inv. true after some iteration w/ $j = k$ $0 \le k < n$.
prior to next iteration now have $j = k+1$
so $\text{Sum} = \sum_{i=0}^{\max(j-1,0)} i = \sum_{i=0}^{k} i$. (since loop inv. assumed true here).
Then in line 4 compute
$\text{Sum} + j = \sum_{i=0}^{k} i + k+1 = \sum_{i=0}^{k+1} i$
and assign $\text{Sum} = \sum_{i=0}^{k+1} i$.
Then j becomes $k+2$ so
$\sum_{i=0}^{\max(j-1,0)} i = \sum_{i=0}^{k+1} i = \text{Sum}$ ✓

Termination After last iteration, $j = n+1$ ✓
since loop inv. is true, $\text{Sum} = \sum_{i=0}^{\max(j-1,0)} i = \sum_{i=0}^{n} i$.

Running Time of algorithm

```
TRIANGLE(n)
1  Sum = 0
2  // Initializes...
3  for j = 0 to n
4      Sum = Sum + j
5      // Replaces the...
6      // This has ...
7  return Sum
```

cost	times
c_1	1
0	1
c_2	$n+2$
c_3	$n+1$
0	$n+1$
0	$n+1$
c_4	1

Rules

(1) each line/instruction takes a constant amount of time.

(2) each ~~constant~~ line gets a different constant

(3) comments take 0 time.

** [Q: how many integers are in the list $a, a+1, \ldots, b$ $(a \le b \in \mathbb{Z})$?

A: $b-a+1$ last − first + 1] **

running time =

$$c_1 + c_2(n+2) + c_3(n+1) + c_4$$
$$= (c_2+c_3)n + (c_1+2c_2+c_3+c_4)$$
$$= an+b \quad \text{(linear function)}$$

Summarize: running time is $\Theta(n)$, $O(n)$, $\Omega(n)$

A faster algorithm

$$\sum_{k=0}^{n} k = \frac{n(n+1)}{2}$$

```
TriangleFast(n)
1  Sum = n
2  // Initializes Sum to n
3  Sum = Sum · (n + 1)
4  // Multiplies Sum by n + 1
5  Sum = Sum/2
6  // Divides by 2
7  return Sum
```

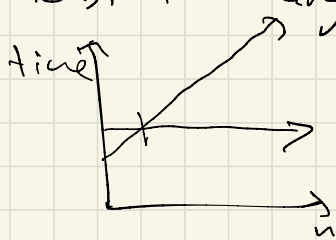
cost	time
c_1	1
0	1
c_2	1
0	1
c_3	1
0	1
c_4	1

running time = $c_1 + c_2 + c_3 + c_4$ = constant

$= \Theta(1), O(1), \Omega(1)$

constant better than linear

(at least for large values of n)



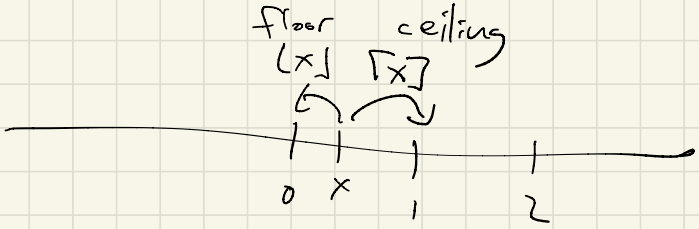
we will care about large n.

```
TriangleFastV2(n)
1  return n · (n + 1)/2
```

§1.4 Common functions

floors and ceilings

$\lfloor x \rfloor$	:= the greatest integer less than or equal to x	(floor of x)
$\lceil x \rceil$	:= the least integer greater than or equal to x	(ceiling of x)



$\lfloor 2.5 \rfloor = 2 \quad \lfloor -2.5 \rfloor = -3$

$\lceil -0.5 \rceil = 0 \quad \lceil 10 \rceil = 10 = \lfloor 10 \rfloor$

$x \in \mathbb{R}$: $x = \lfloor x \rfloor \iff x = \lceil x \rceil \iff x$ is an integer ($x \in \mathbb{Z}$)

$x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$

Reflection principles: $\lfloor -x \rfloor = -\lceil x \rceil, \quad \lceil -x \rceil = -\lfloor x \rfloor$

Fact 1.4.1. Suppose $x \in \mathbb{R}$ and $k \in \mathbb{Z}$. Then

(1) $\lfloor x \rfloor = k$ iff $k \leq x < k+1$
(2) $\lfloor x \rfloor = k$ iff $x - 1 < k \leq x$
(3) $\lceil x \rceil = k$ iff $k - 1 < x \leq k$
(4) $\lceil x \rceil = k$ iff $x \leq k < x + 1$
(5) $\lfloor x + k \rfloor = \lfloor x \rfloor + k$
(6) $\lceil x + k \rceil = \lceil x \rceil + k$
(7) $x < k$ iff $\lfloor x \rfloor < k$
(8) $k < x$ iff $k < \lceil x \rceil$
(9) $x \leq k$ iff $\lceil x \rceil \leq k$
(10) $k \leq x$ iff $k \leq \lfloor x \rfloor$

Fact 1.4.2. For $x \in \mathbb{R}$, if $x \geq 0$, then

$$\left\lfloor \sqrt{\lfloor x \rfloor} \right\rfloor = \lfloor \sqrt{x} \rfloor$$

Proof: Let $m = \lfloor \sqrt{\lfloor x \rfloor} \rfloor$

$0 \leq m \leq \sqrt{\lfloor x \rfloor} < m+1$

squaring:

$m^2 \leq \lfloor x \rfloor < (m+1)^2$

$m^2 \leq x < (m+1)^2$

taking sqrt:

$m \leq \sqrt{x} < m+1$

(m is an int; $\sqrt{x}$ is not necessarily)

$\Rightarrow \lfloor \sqrt{x} \rfloor = m$ bcs $m \in \mathbb{Z}$

$m \leq \lfloor \sqrt{x} \rfloor < m+1$

$\in \mathbb{Z}$ so $m = \lfloor \sqrt{x} \rfloor$

The modulus operator

Division algorithm $a=100$, $b=17$

$100 = 17 \cdot 8 + 15$ $\quad 0 \le 15 < 17$.

8: quotient, 15: remainder

Definition 1.4.3. Fix $n \geq 1$. Define the **modulus operator** (with respect to n) to be the function

$$k \mapsto k \bmod n : \mathbb{Z} \to \{0, \ldots, n-1\}$$

defined for $k \in \mathbb{Z}$ by

$$k \bmod n := \text{the unique } r \in \{0, \ldots, n-1\} \text{ such that there is } q \in \mathbb{Z} \text{ such that } k = nq + r \text{ and } 0 \leq r < n.$$

$100 \bmod 17 = 15$, $100 \,\%\, 17$.

in Python, C, ...

If $k \in \mathbb{Z}$, then

k even $\iff k \bmod 2 = 0$

k odd $\iff k \bmod 2 = 1$.

E.g. Note that $50 = 10 \cdot 5 + 0$

$51 = 10 \cdot 5 + 1$

Definition 1.4.5. Suppose $n \geq 1$ and $k \in \mathbb{Z}$. We say that n **divides** k (notation: $n|k$) if $k \bmod n = 0$. Equivalently, n divides k if there exists $q \in \mathbb{Z}$ such that $k = nq$. If $n|k$ and $q \in \mathbb{Z}$ is such that $na = k$, then we shall call n a **divisor** (or **factor**) of k, and we shall call q the **quotient** of k divided by n.

n divides $k \iff n|k \iff k \bmod n = 0$

$\iff$ exists $q \in \mathbb{Z}$ s.t. $k = qn + 0$

$\iff$ exists $q \in \mathbb{Z}$ s.t. $k = qn$.

Fact 1.4.6. Suppose $a, b \in \mathbb{Z}$ and $b \geq 1$. Set $q := \lfloor a/b \rfloor$ and $r := a \bmod b$. Then

$$a \;=\; bq + r \quad \text{and} \quad 0 \leq r < b.$$

In particular, $a \bmod b = a - b\,\lfloor a/b \rfloor$.

Primes = 2, 3, 5, 7, 11, 13, 17, 19, ...

Composites = 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, ...

Definition 1.4.7. A natural number $p \in \mathbb{N}$ is called a **prime number** if $p \neq 1$ and its only nonnegative divisors are 1 and p; in symbols:

$$p \text{ prime } :\Longleftrightarrow\; p \neq 1 \& \forall d \in \mathbb{N}\ (d|p \to d = 1 \text{ or } d = p)$$

A natural number $n \geq 2$ which is not prime is called **composite**.

0/0

Divisibility Properties 1.4.8. *For every $a, b, c \in \mathbb{Z}$ the following hold:*

(D1) $a|0$, $1|a$, $a|a$
(D2) $a|1$ *if and only if* $a = \pm 1$
(D3) *if* $a|b$ *and* $c|d$, *then* $ac|bd$
(D4) *if* $a|b$ *and* $b|c$, *then* $a|c$
(D5) $a|b$ *and* $b|a$ *if and only if* $a = \pm b$
(D6) *if* $a|b$ *and* $b \neq 0$, *then* $|a| \leq |b|$
(D7) *if* $a|b$ *and* $a|c$, *then for every* $x, y \in \mathbb{Z}$, $a|(bx + cy)$

In particular, the divides relation | is reflexive (D1) and transitive (D4).

Definition 1.4.9. Fix $n \geq 1$. We say that two integers $a, b \in \mathbb{Z}$ are **congruent modulo** n (notation: $a \equiv b \pmod{n}$) if $a \bmod n = b \bmod n$, i.e., if a and b leave the same remainder upon division by n. Equivalently:

$$\begin{aligned} a \equiv b \pmod{n} &\iff n | a - b \\ &\iff \text{there exists } q \in \mathbb{Z} \text{ such that } nq = a - b \\ &\iff a \bmod n = b \bmod n. \end{aligned}$$

$12 \bmod 5 = 2 = 7 \bmod 5$

$\rightsquigarrow 12 \equiv 7 \pmod{5}$

Congruence Properties 1.4.10. *Fix $n \geq 1$. Then for every $a, b, c, d \in \mathbb{Z}$ the following hold:*

(C1) $a \equiv a \pmod{n}$

(C2) *if* $a \equiv b \pmod{n}$, *then* $b \equiv a \pmod{n}$

(C3) *if* $a \equiv b \pmod{n}$ *and* $b \equiv c \pmod{n}$, *then* $a \equiv c \pmod{n}$

(C4) *if* $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$, *then* $a + c \equiv b + d \pmod{n}$ *and* $ac \equiv bd \pmod{n}$

(C5) *if* $a \equiv b \pmod{n}$, *then* $a^m \equiv b^m \pmod{n}$ *for every* $m \in \mathbb{N}$.