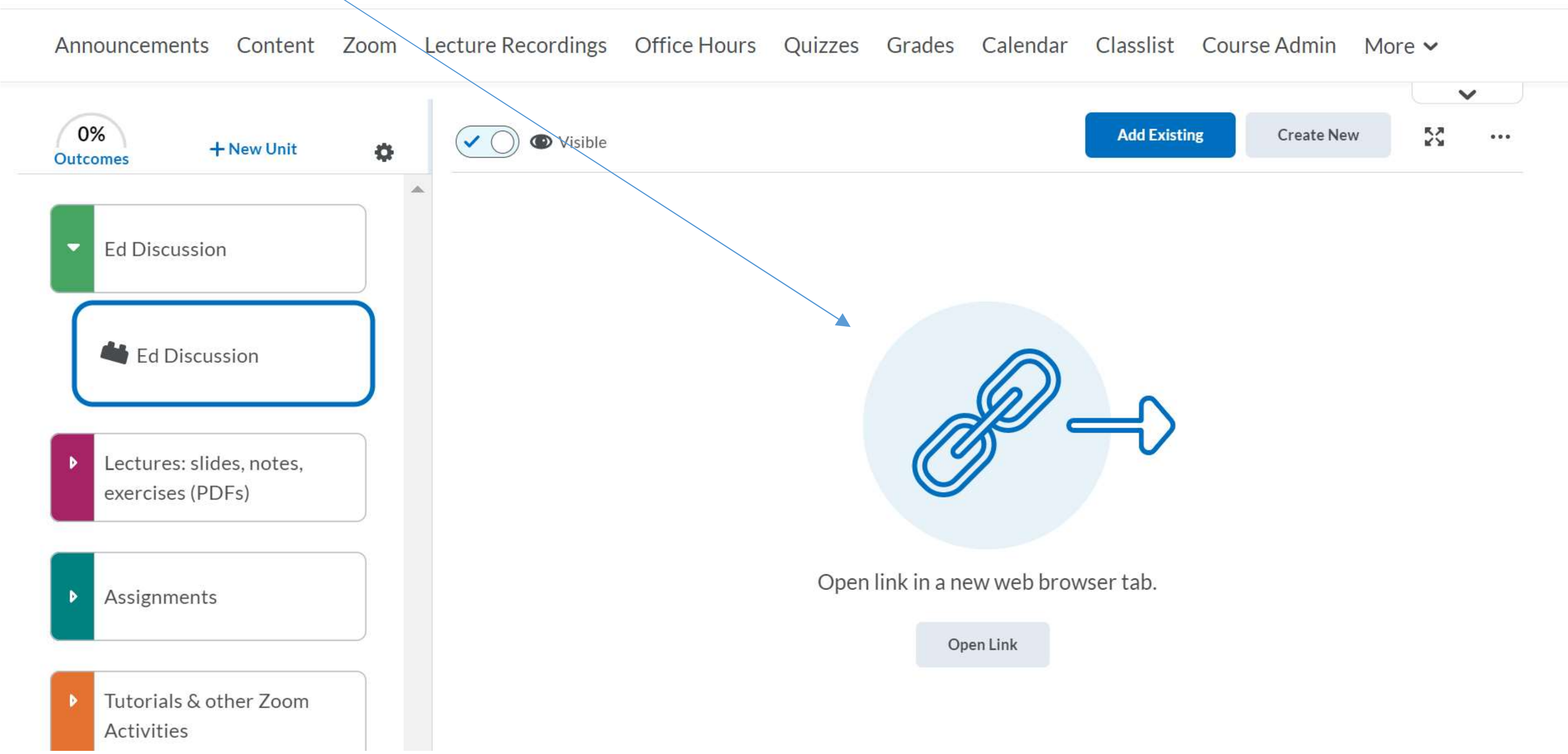
Lecture 1

Grade school algorithms for arithmetic

Fri. Jan. 7, 2022

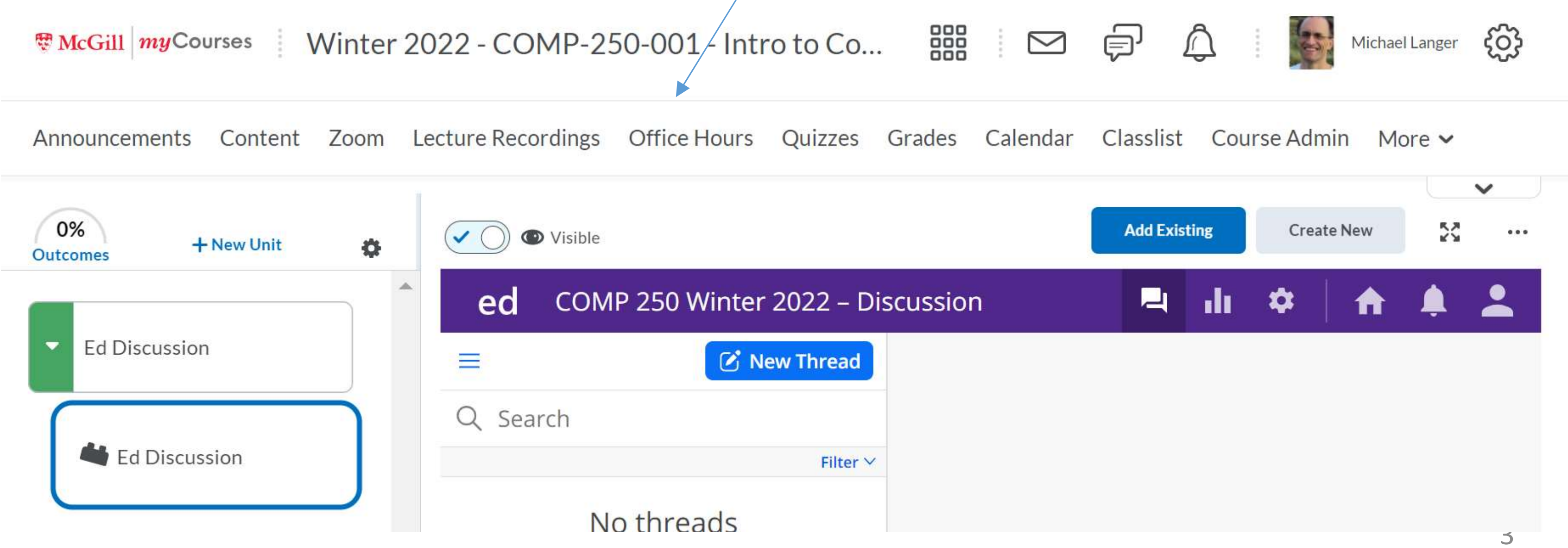
Update: Ed Discussion

This will pop open a new web page which will be our discussion forum



Update: Office Hours

There will be a tutorial as well as one-on-one OH next week to help with IDE set up.



What is an algorithm?

An algorithm is a sequence of instructions or operations for manipulating data to produce some result.

Think of an algorithm as a recipe that works with data such as numbers, text strings, images, sounds,….

[See Khan Academy course on Algorithms for a good intro](#)

[See also this wonderful Netflix series on Algorithms](#).

Today: grade school arithmetic

The first algorithms you ever learned:

- addition
- subtraction
- multiplication
- division

First steps ….

How did you learn arithmetic when you were a child?

First, you memorized the sum of single digit numbers:

$1 + 1 = 2$, $1 + 2 = 3$, $1 + 3 = 4$, … $4 + 7 = 11$, …

Then, you learned to add numbers with multiple digits.

Grade school addition

$$
\begin{array}{cccc c}
 & 2 & 3 & 4 & 9 \\
+ & \underline{5} & \underline{8} & \underline{1} & \underline{3}
\end{array}
$$

Grade school addition

	1	0	1		← carry
	2	3	4	9	
+	5	8	1	3	
	8	1	6	2	

You learned an *algorithm* based on single digit sums.
It used a "for" loop.

You learned a theory of why it works (powers of 10).

What is the *algorithm* for addition $a + b$?

For Python programmers: a[ ] and b[ ] are lists.

In Java, a*[ ] and* b*[ ]* might be "arrays".

The result is a list or array $r[\ ]$.

$$
\begin{array}{rrrrr}
 & a[3] & a[2] & a[1] & a[0] \\
+ & b[3] & b[2] & b[1] & b[0] \\
\hline
r[4] & r[3] & r[2] & r[1] & r[0]
\end{array}
$$

Addition Algorithm

For each column i from 0 to $N-1$ {   // N digits

compute single digit sum : $a[i]$ + $b[i]$ + carry value

determine the single digit result $r[i]$ for that column and the carry value (0 or 1) for the next column

}

$$
\begin{array}{rrrrr}
 & a[3] & a[2] & a[1] & a[0] \\
+ & b[3] & b[2] & b[1] & b[0] \\
\hline
r[4] & r[3] & r[2] & r[1] & r[0]
\end{array}
$$

Addition "Pseudocode"

$$
\begin{array}{l}
carry = 0 \\
\textbf{for } i = 0 \text{ to } N-1 \ \textbf{ do} \\
\quad sum \ \leftarrow \ a[i] + b[i] + carry \\
\quad r[i] \quad \leftarrow \ sum \ \% \ 10 \\
\quad carry \leftarrow \ sum \ / \ 10 \\
\textbf{end for} \\
r[N] \leftarrow carry
\end{array}
$$

Pseudocode looks like code in a high level programming language, but it doesn't have strict syntax rules.

We can and will be quite loose with pseudocode.

(Details to be explained on next slides.)

Addition "Pseudocode"

$\boxed{carry = 0}$

for $i = 0$ to $N-1$ **do**

  $\boxed{sum \leftarrow a[i] + b[i] + carry}$

  $r[i] \leftarrow sum \,\%\, 10$

  $carry \leftarrow sum \,/\, 10$

end for

$r[N] \leftarrow carry$

For the 0th column, there is no incoming carry value.

Compute single digit sum : a[i] + b[i] + carry value

$$\begin{array}{ccccc} & a[3] & a[2] & a[1] & a[0] \\ + & b[3] & b[2] & b[1] & b[0] \\ \hline r[4] & r[3] & r[2] & r[1] & r[0] \end{array}$$

Addition "Pseudocode"

$carry = 0$
for $i = 0$ to $N-1$ **do**
  $sum \leftarrow a[i] + b[i] + carry$
  $\boxed{r[i] \leftarrow sum \% 10}$
  $carry \leftarrow sum / 10$
end for
$r[N] \leftarrow carry$

The result r[i] for that column is the remainder (when dividing by 10)

	$a[3]$	$a[2]$	$a[1]$	$a[0]$
+	$b[3]$	$b[2]$	$b[1]$	$b[0]$
$r[4]$	$r[3]$	$r[2]$	$r[1]$	$r[0]$

Addition "Pseudocode"

$carry = 0$
for $i = 0$ to $N-1$ **do**
$\quad sum \leftarrow a[i] + b[i] + carry$
$\quad r[i] \leftarrow sum \,\%\, 10$
$\quad \boxed{carry \leftarrow sum \,/\, 10}$
end for
$r[N] \leftarrow carry$

The carry value for the next column is the sum divided by 10 (integer division -- ignore remainder)

carry (either 0 or 1)

$$\begin{array}{ccccc} & a[3] & a[2] & a[1] & a[0] \\ + & b[3] & b[2] & b[1] & b[0] \\ \hline r[4] & r[3] & r[2] & r[1] & r[0] \end{array}$$

Addition “Pseudocode”

$carry = 0$
for $i = 0$ to $N-1$ **do**
  $sum \leftarrow a[i] + b[i] + carry$
  $r[i] \leftarrow sum \,\%\, 10$
  $carry \leftarrow sum \,/\, 10$
end for
$\boxed{r[N] \leftarrow carry}$

Add an extra column (which may just be a 0)

$$\begin{array}{cccccc} & 1 & 0 & 1 & & \\ & 2 & 3 & 4 & 9 \\ + & 9 & 8 & 1 & 3 \\ \hline 1 & 2 & 1 & 6 & 2 \end{array}$$

The grade school addition algorithm is *fast*.

It makes use of a clever *number representation*:
each number is represented as *sum of powers of 10*.

(Hindu-Arabic system invented ~2000 years ago)

Imagine instead an algorithm for addition that is based on Roman numerals:

It would be rather awkward!

Roman Numeral Table

1	I	14	XIV	27	XXVII	150	CL
2	II	15	XV	28	XXVIII	200	CC
3	III	16	XVI	29	XXIX	300	CCC
4	IV	17	XVII	30	XXX	400	CD
5	V	18	XVIII	31	XXXI	500	D
6	VI	19	XIX	40	XL	600	DC
7	VII	20	XX	50	L	700	DCC
8	VIII	21	XXI	60	LX	800	DCCC
9	IX	22	XXII	70	LXX	900	CM
10	X	23	XXIII	80	LXXX	1000	M
11	XI	24	XXIV	90	XC	1600	MDC
12	XII	25	XXV	100	C	1700	MDCC
13	XIII	26	XXVI	101	CI	1900	MCM

Grade school subtraction

$$\begin{array}{r} 924 \\ -\ 352 \\ \hline 572 \end{array}$$

How to write an *algorithm* for doing this?

Grade school subtraction

$$
\begin{array}{r}
\overset{8}{\cancel{9}}24 \\
-\ \ 352 \\
\hline
572
\end{array}
$$

How to write an *algorithm* for doing this?

How to express the “borrowing” step?

Grade school subtraction

Here is a slightly more challenging example.

$$\begin{array}{r} 901 \\ -\ \underline{352} \\ 549 \end{array}$$

How to write an algorithm for doing this?

How to express the "borrowing" step?

Grade school subtraction

$$\begin{array}{r} {\scriptstyle 8\ \ 9\ \ 11} \\ \cancel{9}\cancel{0}1 \\ -\ \underline{352} \\ 549 \end{array}$$

8 9 11 means 800 + 90 + 11 (= 901)

How to write an algorithm for doing this?

How to express the "borrowing" step?

Multiplication

Q: What do we mean by $a * b$?

(assuming integers)

Multiplication

Q: What do we mean by $a * b$?

A: We mean: $(a + a + \ldots.. + a)$, b times

a is the "multiplicand"

b is the "multiplier"

Slow Multiplication Algorithm

$$
\begin{array}{l}
product = 0 \\
\textbf{for } i = 1 \text{ to } b \\
\quad product \leftarrow product + a
\end{array}
$$

You learned a *faster* algorithm in grade school…

Grade school multiplication

Example: Suppose the operands each have N digits.

```
  352      a[ ]   "multiplicand"
* 964      b[ ]   "multiplier"
```

Algorithm (pseudocode) starts off with two nested for loops.

for each digit in $b[\ ]$
   for each digit in $a[\ ]$
        multiply the two digits and …. ?

Grade school multiplication

Step 1: make 2D table $table[\ ][\ ]$

```
   352
 * 964
```

$a[N]$

$b[N]$

```
   1 2 0
   1408
  2 3 1
  21120
 3 4 1
 316800
```

row index

$table[N][2\ N]$

column index

Pseudocode for Multiplication

<u>Step 1:</u> make 2D table $table$[ ][ ]

$$
\begin{array}{l}
\textbf{for } j = 0 \text{ to } N-1 \textbf{ do} \\
\quad carry \leftarrow 0 \\
\quad \textbf{for } i = 0 \text{ to } N-1 \textbf{ do} \\
\quad\quad prod \leftarrow (a[i] * b[j] + carry) \\
\quad\quad table[j][i+j] \leftarrow prod\%10 \\
\quad\quad carry \leftarrow prod/10 \\
\quad \textbf{end for} \\
\quad table[j][N+j] \leftarrow carry \\
\textbf{end for}
\end{array}
$$

// for each digit of b

// for each digit of a

you can verify this on your own

a[ ] "multiplicand"

* $\underline{b[\ ]}$ "multiplier"

$table$ [ ][ ]

<u>Step 2:</u> for each column of the table, add up values in that column along with the carry.

352	$a[\,]$
* 964	$b[\,]$
001000	*carry*
1408	
21120	$table\,[\,][\,]$
316800	
339328	$r[\;]$

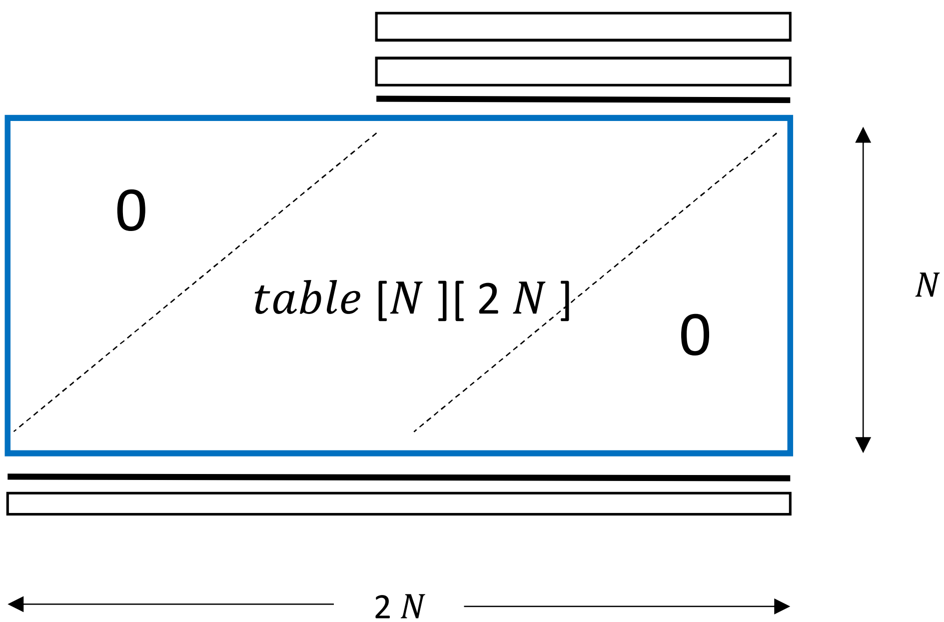
Pseudocode for Multiplication

<u>Step 2:</u> for each column i in table, sum up the single digit numbers over all rows j and add the carry

$carry \leftarrow 0$
for $i = 0$ to $2 * N - 1$ **do** // for each column of table
  $sum \leftarrow carry$
  for $j = 0$ to $N - 1$ **do** // for each row of table
    $sum \leftarrow sum + table[j][i]$
  end for
  $r[i] \leftarrow sum\%10$
  $carry \leftarrow sum/10$
end for

<u>ASIDE:</u> Grade school multiplication uses a $table$ of size $2N * N$.

Q: Is this table necessary?



<u>ASIDE:</u> Grade school multiplication uses a $table$ of size $2N * N$.

Q: Is this table necessary?

A: No.

We could instead just add each row to an accumulating sum.

We still need to compute each row, but we don't need to store them all in one big table. We don't need to two steps (1 and 2).

Division

Q: What do we mean by a/b ?
(Assume $a > b > 0$.)

Division

Q: What do we mean by a/b ?
(Assume $a > b > 0.$)

A: How many times can we subtract b from a until the remainder is in {0,… $b - 1\}$?

Division

Q: What do we mean by a/b ?
(Assume $a > b > 0.$)

A: $a = q * b + r, \qquad 0 \leq r < b$

q is quotient, r is remainder

Slow division algorithm

To compute a/b, repeatedly subtract b from a until the result is less than b.

$$
\begin{aligned}
& q = 0 \\
& r = a \\
& \textbf{while } r \geq b \ \textbf{ do} \\
& \quad q \leftarrow q + 1 \\
& \quad r \leftarrow r - b \\
& \textbf{end while}
\end{aligned}
$$

You learned a much faster algorithm in grade school.

“Long Division”

```
          5 ...
     _______________
723 | 41672542996
      3615
       ----
        552  ...etc
```

To write out an algorithm for doing this, it helps to understand what long division is doing. This is not obvious, and I’m pretty sure your grade 3 teacher didn’t explain it.
(Rather, the teacher probably just taught you how to do it.)

Algorithms and Data Structures

In this course, you will learn many interesting *algorithms* for solving problems with data such as numbers and text (strings).

You will also learn about many *data structures*, which are ways of representing data so that the algorithms can run faster (or are easier to code).

Examples of data structures are array lists, linked lists, trees, hash tables, graphs, …

"Computational Complexity"

What do we mean by a 'fast' versus 'slow' algorithm ?

Let $t(N)$ be the *number of operations* of a computation whose "input size is N".

In arithmetic, the numbers we are operating on each have N digits. The operations include single digit sums and products, assigning a value to a variable, incrementing a counter in loop, etc.

Q: How many operations are required for addition, subtraction, multiplication, division?

Q: How should we express our intuition that multiplication takes longer than addition ?

Grade School Addition

$carry \leftarrow 0$ 1

for $i = 0$ to $N-1$ **do**

$$\left.\begin{array}{lcl} sum & \leftarrow & a[i] + b[i] + carry \\ r[i] & \leftarrow & sum \ \% \ 10 \\ carry & \leftarrow & sum \ / \ 10 \end{array}\right\} N$$

end for

$r[N] \leftarrow carry$ 1

We mean that each part of the program is executed 1 or N times.

Grade School Addition

$$
\begin{array}{l}
carry \leftarrow 0 \\
\textbf{for } i = 0 \text{ to } N-1 \ \ \textbf{do} \\
\quad \left.\begin{array}{lcl}
sum & \leftarrow & a[i] + b[i] + carry \\
r[i] & \leftarrow & sum \ \% \ 10 \\
carry & \leftarrow & sum \ / \ 10
\end{array}\right\} \\
\textbf{end for} \\
r[N] \leftarrow carry
\end{array}
\qquad
\begin{array}{c}
c_1 \\ \\ \\ c_2 N \\ \\ \\ c_3
\end{array}
$$

Some operations take more time than others, which we can express using different constants.

The number of steps of this algorithm is $t(N) = (c_1 + c_3) + c_2 * N$.
When we analyze algorithms, we often ignore these constants.

Grade School Multiplication

Step 1

for $j = 0$ to $N-1$ **do** — N
  $carry \leftarrow 0$
  for $i = 0$ to $N-1$ **do**
    $prod \leftarrow (a[i] * b[j] + carry)$
    $table[j][i+j] \leftarrow prod\%10$ — N^2
    $carry \leftarrow prod/10$
  end for
  $table[j][N+j] \leftarrow carry$ — N
end for

Step 2

$carry \leftarrow 0$ — 1
for $i = 0$ to $2*N-1$ **do**
  $sum \leftarrow carry$ — N
  for $j = 0$ to $N-1$ **do**
    $sum \leftarrow sum + table[j][i]$ — N^2
  end for
  $r[i] \leftarrow sum\%10$ — N
  $carry \leftarrow sum/10$
end for

Similarly, we could include constant factors for each of these terms.

Computational Complexity & “big O”

We will say…

Grade school addition takes time O(N), or “big O of N” where N is the number of digits in our two operands.

Grade school multiplication takes time O(N^2) or “big O of N squared”.

Next week …

Lectures

- Mon Jan 10

  math – log, mod, base arithmetic & conversion

- Wed. Jan 12

  Java primitive types

Homework (TODO)

- See link from last lecture to w3schools.
- Install either Eclipse or IntelliJ.
- See Content -> tutorials.

  Tutorial TA office hours to help with this coming soon.