

lecture 1

grade school algorithms
(addition and multiplication)

An **algorithm** is a sequence of instructions for accomplishing a task. In computer science, the task is to compute something.

Addition

$$\begin{array}{r} 101 \\ 2343 \\ + 5819 \\ \hline 8162 \end{array}$$

Note: you needed to memorize single digit sums to do this.

Inputs: two arrays $a[]$, $b[]$ of digits from 0, ... 9 that represent numbers as a sum of powers of 10

$$\text{e.g. } 5819 = 5 \times 10^3 + 8 \times 10^2 + 1 \times 10^1 + 9 \times 10^0$$

Output: an array $r[]$ which represents the sum of the input numbers, again expressed as a sum of powers of 10.

What is the grade school algorithm?

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

Algorithm in "Pseudo code"

// $a[]$, $b[]$ are arrays of size N

$c \leftarrow 0$

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

$r[i] \leftarrow (a[i] + b[i] + c) \bmod 10$

$c \leftarrow (a[i] + b[i] + c) / 10$

end

$r[N] \leftarrow c$

You can compute a sum of two arbitrary numbers by representing each number as a sum of powers of 10 and using simple operations only (sum of single digit numbers !)

Trivial? No.

Roman	vs.	Hindu - Arabic Numerals
		I 1
		II 2
		III 3
No obvious		IV 4
algorithm for		V 5
summing Roman		VI 6
numerals		VII 7
		VIII 8
		IX 9
(FYI, they did		X 10
arithmetic using		⋮
an abacus)		XLIX 49
		⋮

Multiplication

$$\begin{array}{r} 352 \\ \times 264 \\ \hline \end{array}$$

Note: you needed to memorize single digit products to do this.

Multiplication

$$\begin{array}{r} 352 \\ \times 964 \\ \hline 1408 \\ 21120 \\ 316800 \\ \hline 339328 \end{array}$$

$a[]$
 $b[]$
 $t[][]$
 $r[]$

Multiplication Algorithm

// Step 1: compute table $t[]$

```

for row = 0 to N-1
  c ← 0
  for col = 0 to N-1
    prod ← a[col] * b[row] + c
    t[row][col + row] ← prod mod 10
    c ← prod / 10
  end
  t[row][col + row + 1] = c
end

```

Multiplication Algorithm

// Step 2: compute result $r[]$

```

c ← 0
for col = 0 to 2*N-1
  sum ← 0
  for row = 0 to N-1
    sum ← sum + t[row][col] + c
  end
  r[col] = sum mod 10
  c = sum / 10
end
r[2*N] = c

```

"Analysis of Algorithms"

We all know how to add and multiply two numbers. But we see that "coding" this "algorithm" is not straight forward.

(Wait a day and then try it yourself.)

Addition

```

c ← 0
for i = 0 to N-1
  r[i] ← (a[i] + b[i] + c) mod 10
  c ← (a[i] + b[i] + c) / 10
end
r[N] ← c
    
```

} 1 time
 } N times
 } 1 time

Q: How many operations?

A: At most $C_1 + C_2 N$ for some constants C_1 and C_2 .

Analysis of Algorithms - Multiplication

```

for row = 0 to N-1
  c ← 0
  for col = 0 to N-1
    prod ← a[col] * b[row] + c
    t[row][col + row] ← prod mod 10
    c ← prod / 10
  end
  t[row][col + row + 1] = c
end
c ← 0
for col = 0 to 2*N-1
  sum ← 0
  for row = 0 to N-1
    sum ← sum + t[row][col] + c
  end
  r[col] = sum mod 10
  c = sum / 10
end
r[2*N] = c
    
```

} N
 } N² So total is at most
 } N C₁ + C₂ N
 } 1 + C₃ N²
 } 2N
 } 2N²
 } 2N for some constants
 } 1 C₁ C₂ C₃

Analysis of Algorithms

Typically we care less about the values of C_1, C_2, C_3 etc and more about the dependence on N (eg. N^0 vs N^1 vs N^2 ...)

Why? The constants C_1, C_2, C_3 depend on the machine, which we as COMP 250 students don't yet know about.

Binary Numbers

vs.

Decimal Numbers

Base 10

$$5819 = 5 \times 10^3 + 8 \times 10^2 + 1 \times 10^1 + 9 \times 10^0$$

Base 2

$$\begin{aligned}
 11010 &= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\
 &= 16 + 8 + 2 \\
 &= 26
 \end{aligned}$$

To convert from binary to decimal, you need to know the powers of 2.

<u>n</u>	<u>2^n</u>
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
⋮	⋮

} memorize

Counting in Binary

<u>decimal</u>	<u>binary</u>
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
⋮	⋮