http://algs4.cs.princeton.edu

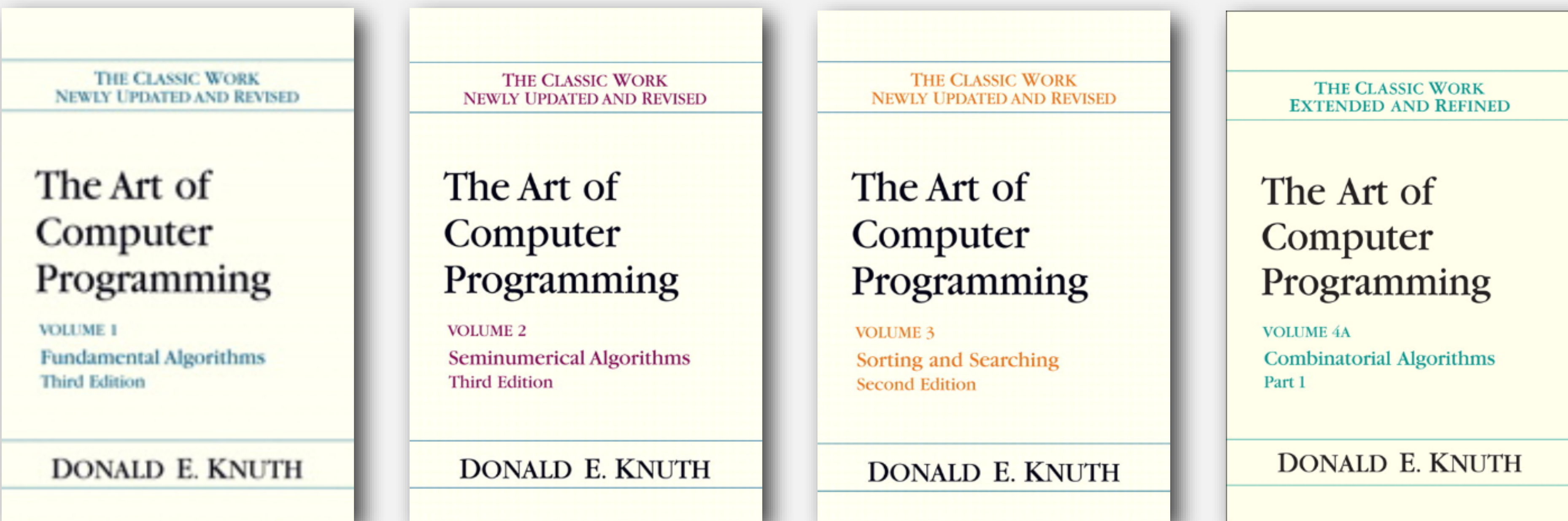
1.4 ANALYSIS OF ALGORITHMS

- *introduction*
- *observations*
- *mathematical models*
- *order-of-growth classifications*
- *theory of algorithms*
- *memory*

Mathematical models for running time

Total running time: sum of cost × frequency for all operations.

- Need to analyze program to determine set of operations.
- Cost depends on machine, compiler.
- Frequency depends on algorithm, input data.



Donald Knuth
1974 Turing Award

In principle, accurate mathematical models are available.

Cost of basic operations

Challenge. How to estimate constants.

operation	example	nanoseconds †
integer add	`a + b`	2.1
integer multiply	`a * b`	2.4
integer divide	`a / b`	5.4
floating-point add	`a + b`	4.6
floating-point multiply	`a * b`	4.2
floating-point divide	`a / b`	13.5
sine	`Math.sin(theta)`	91.3
arctangent	`Math.atan2(y, x)`	129.0
...	...	...

† Running OS X on Macbook Pro 2.2GHz with 2GB RAM

Cost of basic operations

Observation. Most primitive operations take constant time.

operation	example	nanoseconds †
variable declaration	`int a`	c_1
assignment statement	`a = b`	c_2
integer compare	`a < b`	c_3
array element access	`a[i]`	c_4
array length	`a.length`	c_5
1D array allocation	`new int[N]`	$c_6\ N$
2D array allocation	`new int[N][N]`	$c_7\ N^2$

Caveat. Non-primitive operations often take more than constant time.

novice mistake: abusive string concatenation

Example: 1-SUM

Q. How many instructions as a function of input size N ?

```
int count = 0;
for (int i = 0; i < N; i++)
   if (a[i] == 0)
      count++;
```

N array accesses

operation	frequency
variable declaration	2
assignment statement	2
less than compare	$N + 1$
equal to compare	N
array access	N
increment	N to $2\,N$

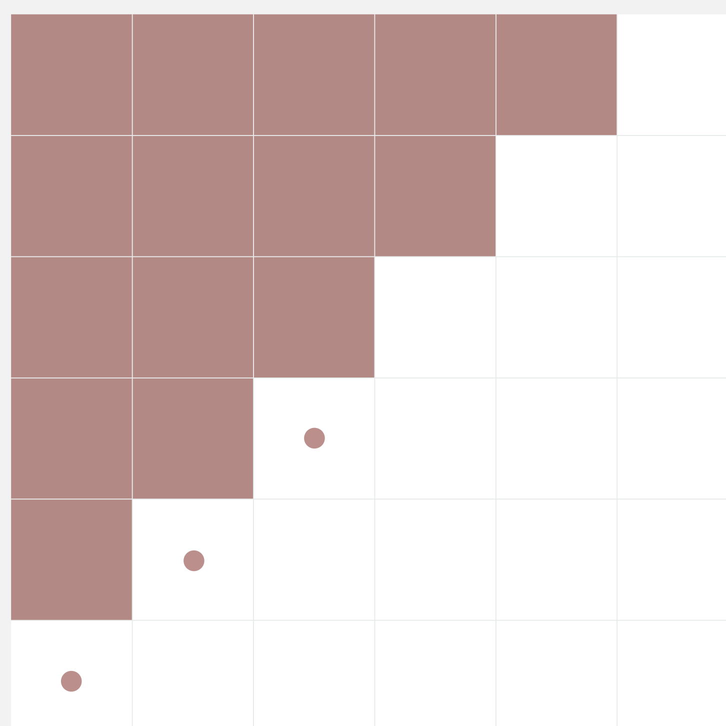
Example: 2-SUM

Q. How many instructions as a function of input size N?

```
int count = 0;
for (int i = 0; i < N; i++)
   for (int j = i+1; j < N; j++)
      if (a[i] + a[j] == 0)
         count++;
```

$$0 + 1 + 2 + \ldots + (N-1) \;=\; \frac{1}{2}N\,(N-1) \;=\; \binom{N}{2}$$

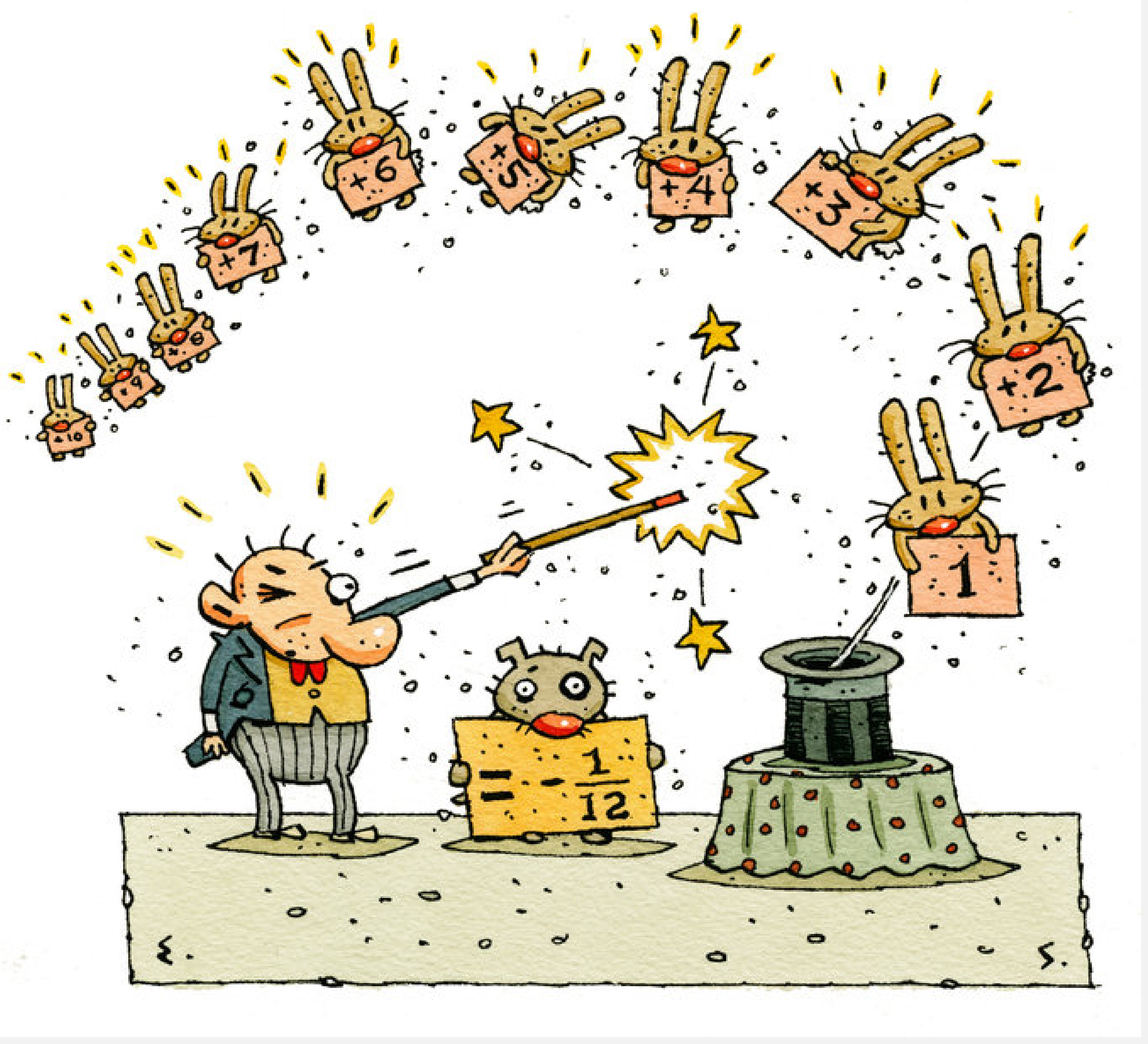
Pf. [n even]



$$0 + 1 + 2 + \ldots + (N-1) \;=\; \underbrace{\frac{1}{2}N^2}_{\text{half of square}} \;-\; \underbrace{\frac{1}{2}N}_{\text{half of diagonal}}$$

String theory infinite sum

$$1 + 2 + 3 + 4 + \ \ldots = \ -\frac{1}{12}$$



http://www.nytimes.com/2014/02/04/science/in-the-end-it-all-adds-up-to.html

Example: 2-SUM

Q. How many instructions as a function of input size N?

```
int count = 0;
for (int i = 0; i < N; i++)
   for (int j = i+1; j < N; j++)
      if (a[i] + a[j] == 0)
         count++;
```

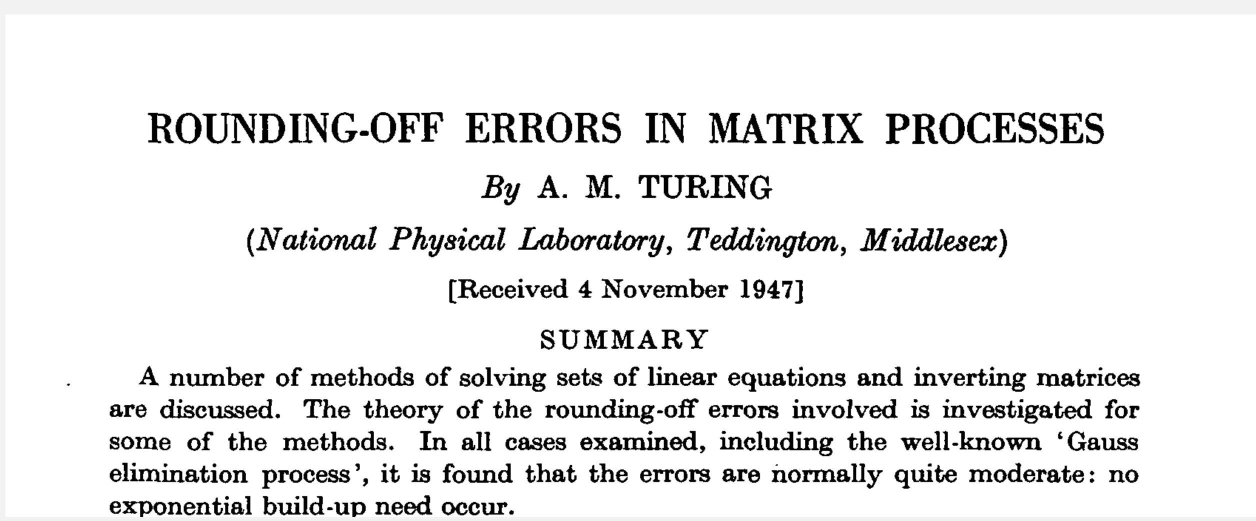
$$0 + 1 + 2 + \ldots + (N-1) = \frac{1}{2} N(N-1) = \binom{N}{2}$$

operation	frequency
variable declaration	$N + 2$
assignment statement	$N + 2$
less than compare	$\frac{1}{2}(N+1)(N+2)$
equal to compare	$\frac{1}{2} N(N-1)$
array access	$N(N-1)$
increment	$\frac{1}{2} N(N-1)$ to $N(N-1)$

tedious to count exactly

Simplifying the calculations

" *It is convenient to have a measure of the amount of work involved in a computing process, even though it be a very crude one. We may count up the number of times that various elementary operations are applied in the whole process and then given them various weights. We might, for instance, count the number of additions, subtractions, multiplications, divisions, recording of numbers, and extractions of figures from tables. In the case of computing with matrices most of the work consists of multiplications and writing down numbers, and we shall therefore only attempt to count the number of multiplications and recordings.* " — *Alan Turing*


ROUNDING-OFF ERRORS IN MATRIX PROCESSES

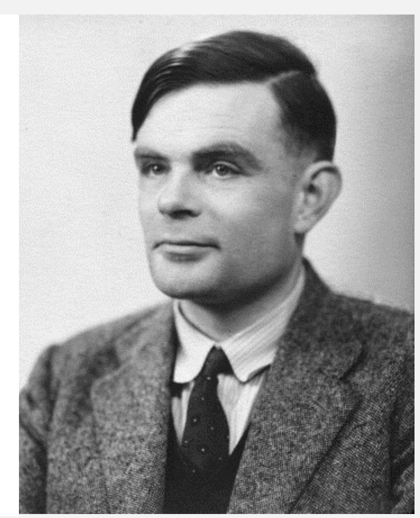
By A. M. TURING

(*National Physical Laboratory, Teddington, Middlesex*)

[Received 4 November 1947]

SUMMARY

A number of methods of solving sets of linear equations and inverting matrices are discussed. The theory of the rounding-off errors involved is investigated for some of the methods. In all cases examined, including the well-known 'Gauss elimination process', it is found that the errors are normally quite moderate: no exponential build-up need occur.



Simplification 1: cost model

Cost model. Use some basic operation as a proxy for running time.

```
int count = 0;
for (int i = 0; i < N; i++)
   for (int j = i+1; j < N; j++)
      if (a[i] + a[j] == 0)
         count++;
```

$$0 + 1 + 2 + \ldots + (N-1) \;=\; \frac{1}{2}\, N\,(N-1) \;=\; \binom{N}{2}$$

operation	frequency
variable declaration	$N + 2$
assignment statement	$N + 2$
less than compare	$\frac{1}{2}\,(N+1)\,(N+2)$
equal to compare	$\frac{1}{2}\,N\,(N-1)$
array access	$N\,(N-1)$
increment	$\frac{1}{2}\,N\,(N-1)$ to $N\,(N-1)$

cost model = array accesses

(we assume compiler/JVM do not optimize any array accesses away!)

Simplification 2: tilde notation

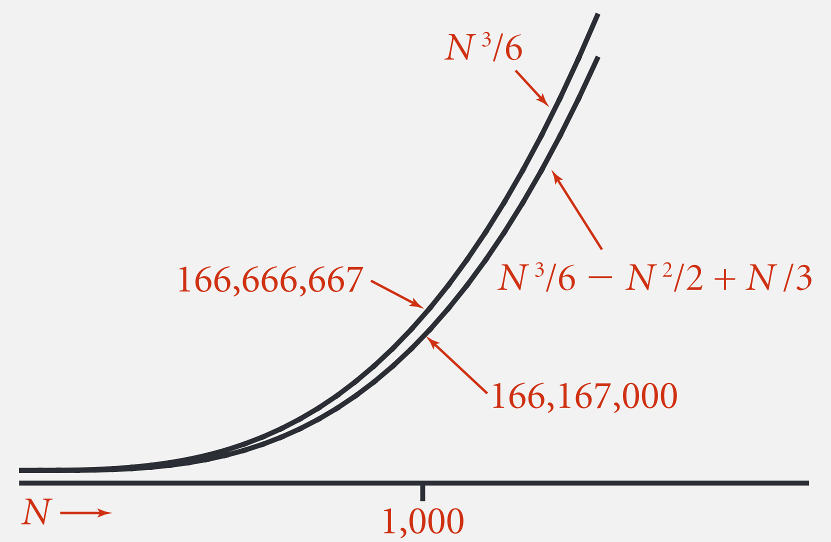
- Estimate running time (or memory) as a function of input size N.
- Ignore lower order terms.
  - when N is large, terms are negligible
  - when N is small, we don't care

Ex 1. $\frac{1}{6} N^3 + 20 N + 16 \quad \sim \quad \frac{1}{6} N^3$

Ex 2. $\frac{1}{6} N^3 + 100 N^{4/3} + 56 \quad \sim \quad \frac{1}{6} N^3$

Ex 3. $\frac{1}{6} N^3 - \underbrace{\frac{1}{2} N^2 + \frac{1}{3} N}_{} \quad \sim \quad \frac{1}{6} N^3$

discard lower-order terms
(e.g., N = 1000: 166.67 million vs. 166.17 million)



Leading-term approximation

Technical definition. $f(N) \sim g(N)$ means $\displaystyle \lim_{N \to \infty} \frac{f(N)}{g(N)} = 1$

Simplification 2: tilde notation

- Estimate running time (or memory) as a function of input size N.
- Ignore lower order terms.
  - when N is large, terms are negligible
  - when N is small, we don't care

operation	frequency	tilde notation
variable declaration	$N + 2$	$\sim N$
assignment statement	$N + 2$	$\sim N$
less than compare	$\frac{1}{2}(N+1)(N+2)$	$\sim \frac{1}{2} N^2$
equal to compare	$\frac{1}{2} N (N-1)$	$\sim \frac{1}{2} N^2$
array access	$N(N-1)$	$\sim N^2$
increment	$\frac{1}{2} N (N-1)$ to $N(N-1)$	$\sim \frac{1}{2} N^2$ to $\sim N^2$

Example: 2-SUM

Q. Approximately how many array accesses as a function of input size N?

```
int count = 0;
for (int i = 0; i < N; i++)
   for (int j = i+1; j < N; j++)
      if (a[i] + a[j] == 0)
         count++;
```

"inner loop"

$$0 + 1 + 2 + \ldots + (N-1) \;=\; \frac{1}{2} N (N-1) \;=\; \binom{N}{2}$$

A. $\sim N^2$ array accesses.

Bottom line. Use cost model and tilde notation to simplify counts.

Example: 3-SUM

Q. Approximately how many array accesses as a function of input size N?

```
int count = 0;
for (int i = 0; i < N; i++)
   for (int j = i+1; j < N; j++)
      for (int k = j+1; k < N; k++)
         if (a[i] + a[j] + a[k] == 0)
            count++;
```

"inner loop"

$$\binom{N}{3} = \frac{N(N-1)(N-2)}{3!} \sim \frac{1}{6}N^3$$

A. $\sim \frac{1}{2} N^3$ array accesses.

Bottom line. Use cost model and tilde notation to simplify counts.

Diversion: estimating a discrete sum

Q. How to estimate a discrete sum?

A1. Take a discrete mathematics course.

A2. Replace the sum with an integral, and use calculus!

Ex 1. $1 + 2 + \ldots + N.$

$$\sum_{i=1}^{N} i \;\sim\; \int_{x=1}^{N} x\,dx \;\sim\; \frac{1}{2}N^2$$

Ex 2. $1^k + 2^k + \ldots + N^k.$

$$\sum_{i=1}^{N} i^k \;\sim\; \int_{x=1}^{N} x^k dx \;\sim\; \frac{1}{k+1}N^{k+1}$$

Ex 3. $1 + 1/2 + 1/3 + \ldots + 1/N.$

$$\sum_{i=1}^{N} \frac{1}{i} \;\sim\; \int_{x=1}^{N} \frac{1}{x}dx \;=\; \ln N$$

Ex 4. 3-sum triple loop.

$$\sum_{i=1}^{N}\sum_{j=i}^{N}\sum_{k=j}^{N} 1 \;\sim\; \int_{x=1}^{N}\int_{y=x}^{N}\int_{z=y}^{N} dz\,dy\,dx \;\sim\; \frac{1}{6}N^3$$

Estimating a discrete sum

Q. How to estimate a discrete sum?

A1. Take a discrete mathematics course.

A2. Replace the sum with an integral, and use calculus!

Ex 4. $1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots$

$$\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = 2$$

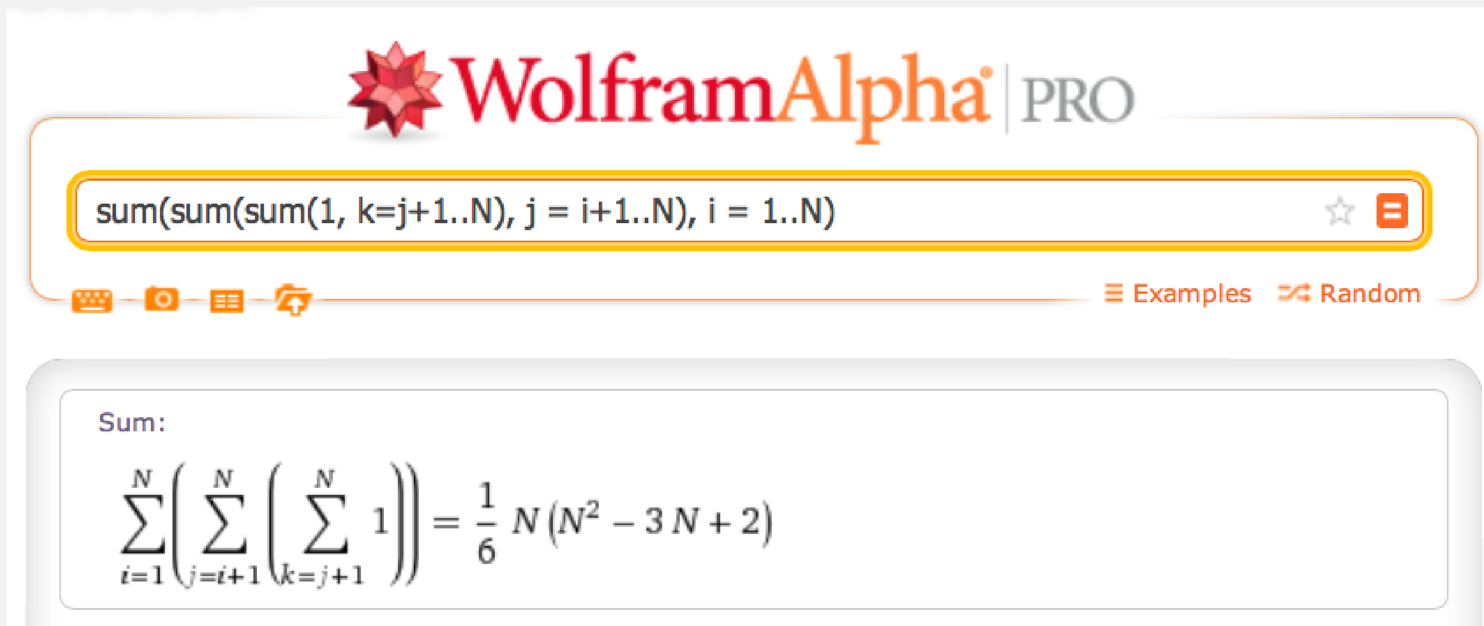
$$\int_{x=0}^{\infty} \left(\frac{1}{2}\right)^x dx = \frac{1}{\ln 2} \approx 1.4427$$

Caveat. Integral trick doesn't always work!

Estimating a discrete sum

Q. How to estimate a discrete sum?

A3. Use Maple or Wolfram Alpha.



wolframalpha.com

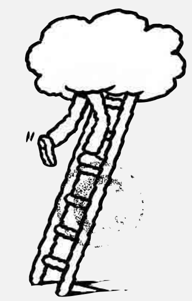
```
[wayne:nobel.princeton.edu] > maple15
    |\^/|     Maple 15 (X86 64 LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2011
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> factor(sum(sum(sum(1, k=j+1..N), j = i+1..N), i = 1..N));

                            N (N - 1) (N - 2)
                            -----------------
                                    6
```

Mathematical models for running time

In principle, accurate mathematical models are available.

In practice,

- Formulas can be complicated.
- Advanced mathematics might be required.
- Exact models best left for experts.



costs (depend on machine, compiler)

$$T_N = c_1 A + c_2 B + c_3 C + c_4 D + c_5 E$$

$A =$ array access
$B =$ integer add
$C =$ integer compare
$D =$ increment
$E =$ variable assignment

frequencies
(depend on algorithm, input)

Bottom line. We use approximate models in this course: $T(N) \sim c\, N^3$.