

Algorithms – Spring '25

Review for
HWD



Last time:

- New classroom!
- Overview of syllabus:

2 main resources:

- My course page
 - ↳ Lots of typos/links fixed
so thanks for emails!
 - Canvas
 - ↳ links to Gradescape
& Pensoft
- HW0: due next Wed.

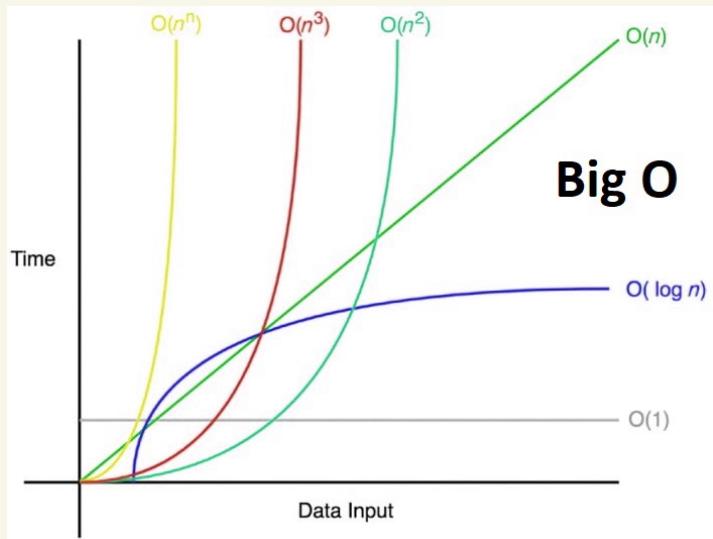
Some background reminders

① Big-O: ignore constants!
Also, smaller terms "disappear":

$$\circ 5 \cdot 2^n + 6n^3 + 3$$

$$\circ \frac{5}{3} n^2 - 33n + 1$$

$$\circ 1,634n^2 + 10n + 1,000,000$$



More formally! $f(n)$ is $O(g(n))$
if

Prove: $5n^2 + 11n - 6$ is $O(n^2)$:

② Logarithms: useful identities!

Find it in
your discrete
math reference,
ie 

$$\log_b(xy) = \log_b(x) + \log_b(y)$$

$$\log_b\left(\frac{x}{y}\right) = \log_b(x) - \log_b(y)$$

$$\log_b(x^y) = y \log_b(x)$$

$$\log_b(\sqrt[y]{x}) = \frac{\log_b(x)}{y}$$

Why? Logarithms are all
about exponents!
What is $2^a \cdot 2^b$?

Another:

$$\log_a b = \frac{\log_x b}{\log_x a} \text{ with any base } x$$

Question: Is $\log_{10} n$ big-O of
 $\log_2 n$?

$$\log_{10} n =$$

Ex: what about
 $\log_a(a^x)$?

Or : $\log_2(n^2)$

③ Summations:

again, your discrete math book has a table.
Find it.

Ex:

Helpful Summation Identities	
$\sum_{i=1}^n c = nc$	for every c that does not depend on i (1)
$\sum_{i=0}^n i = \sum_{i=1}^n i = \frac{n(n+1)}{2}$	Sum of the first n natural numbers (2)
$\sum_{i=1}^n 2i - 1 = n^2$	Sum of first n odd natural numbers (3)
$\sum_{i=0}^n 2i = n(n+1)$	Sum of first n even natural numbers (4)
$\sum_{i=1}^n \log i = \log n!$	Sum of logs (5)
$\sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$	Sum of the first squares (6)
$\sum_{i=0}^n i^3 = \left(\sum_{i=0}^n i\right)^2 = \left(\frac{n(n+1)}{2}\right)^2 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$	Nichomacus' Theorem (7)
$\sum_{i=0}^{n-1} a^i = \frac{1-a^n}{1-a}$	Sum of geometric progression (8)
$\sum_{i=0}^{n-1} \frac{1}{2^i} = 2 - \frac{1}{2^{n-1}}$	Special case for $n = 2$ (9)
$\sum_{i=0}^{n-1} ia^i = \frac{a-na^n+(n-1)a^{n+1}}{(1-a)^2}$	(10)
$\sum_{i=0}^{n-1} (b+id)a^i = b \sum_{i=0}^{n-1} a^i + d \sum_{i=0}^{n-1} ia^i$	(11)
	(12)

Notation:

$$\sum_{i=1}^n f(i)$$

Ex: $\sum_{i=1}^n 1$, or

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

④ Induction
There is a template!

Base case:

Ind hypothesis:

Ind. step:

Think of this as "automating"
a proof.

Learn it, use it, & love it!

Example:

EXAMPLE 3 Use mathematical induction to show that

$$1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1$$

"Structural" induction:

Let n be a positive integer. Show that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes, where these pieces cover three squares at a time, as shown in Figure 4.

3 Pseudo code & runtime: Discrete math examples

ALGORITHM 1 Finding the Maximum Element in a Finite Sequence.

```
procedure max( $a_1, a_2, \dots, a_n$ : integers)
  max :=  $a_1$ 
  for  $i := 2$  to  $n$ 
    if max <  $a_i$  then max :=  $a_i$ 
  return max {max is the largest element}
```

ALGORITHM 2 The Linear Search Algorithm.

```
procedure linear search( $x$ : integer,  $a_1, a_2, \dots, a_n$ : distinct integers)
   $i := 1$ 
  while ( $i \leq n$  and  $x \neq a_i$ )
     $i := i + 1$ 
  if  $i \leq n$  then location :=  $i$ 
  else location := 0
  return location {location is the subscript of the term that equals  $x$ , or is 0 if  $x$  is not found}
```

Pseudocode conventions here:

Variable assignment:

Boolean comparison:

$\text{if } (x = 5)$

Arrays: $A[0..n-1]$
- each $A[i]$

Loops: $\text{for } i \leftarrow 1 \text{ to } 100$
 $A[i] \leftarrow i$

Again, takes practice! Read
intro chapter + then give
HWO/HW1 a try.

Pseudocode format:

In a pinch, pretend you're
in Python/Ruby.

High level + readable.

I realize this is not a "definition"-
that is the point!

It's about effective communication.

Initially:

- lots of examples
- lots of practice
- reach out if you have questions
- in a pinch - peer evaluation!

Example (& tie to runtimes):

Initializing arrays.

FIBONACCIMULTIPLY($X[0..m-1], Y[0..n-1]$):

$hold \leftarrow 0$

for $k \leftarrow 0$ to $n+m-1$

 for all i and j such that $i+j = k$

$hold \leftarrow hold + X[i] \cdot Y[j]$

$Z[k] \leftarrow hold \bmod 10$

$hold \leftarrow \lfloor hold/10 \rfloor$

return $Z[0..m+n-1]$

Space & runtime:

Q: What if all won't get filled?

More Complex: Recursion!

Algorithm 1 Quicksort

```
1: procedure QUICKSORT( $A, p, r$ )
2:   if  $p < r$  then
3:      $q = \text{PARTITION}(A, p, r)$ 
4:     QUICKSORT( $A, p, q - 1$ )
5:     QUICKSORT( $A, q + 1, r$ )
6:   end if
7: end procedure
8: procedure PARTITION( $A, p, r$ )
9:    $x = A[r]$ 
10:   $i = p - 1$ 
11:  for  $j = p$  to  $r - 1$  do
12:    if  $A[j] < x$  then
13:       $i = i + 1$ 
14:      exchange  $A[i]$  with  $A[j]$ 
15:    end if
16:    exchange  $A[i]$  with  $A[r]$ 
17:  end for
18: end procedure
```

QuickSort Pseudocode Example

Recursive Algorithms : Chapter 1

1st half:

Setup, plus (hopefully)
familiar examples:

- Towers of Hanoi
- Merge Sort

2nd half:

- Recap of recurrences
& "Master theorem"
- Linear time Selection
- Multiplication (again)
- Exponentiation