Bioinformatics algorithms

Review of algorithmic techniques

A first problem

Recap of 1^{st} time:

- I did battle with technology (+ lost).
- Syllabus review
- Correctness
- Some runtimes

Today

- More on runtimes
- Recursion + Iteration
- Brute force
- The partial digest problem (PDP)

Efficiency (2.7 & 2.8 in book)

- Exact speed can depend on many variables besides the algorithm.

Issues at play:

- machine
- language
- actual algorithm

Alternative approach:

Count primitive operations, which are smallest operations.

In addition: generally only examine worst case running time.

Why? more do-able, & more pessimistic

Now: How to actually compare?

- Remember small difference may be due to processor, language, or any number of things that aren't dependent on the algorithm.

- Also: need a way to account for inputs changing

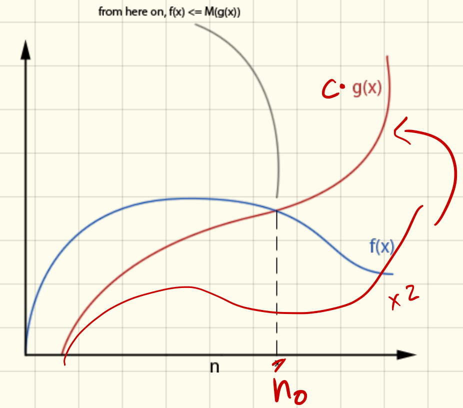
  eg searching a list

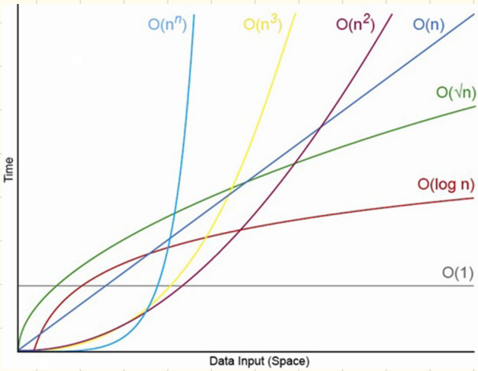
Big-O

Big-O notation

We say $f(n)$ is $O(g(n))$ if $\forall n > n_0$, $\exists c > 0$ such that

$$f(n) \le c \cdot g(n)$$





Common runtimes

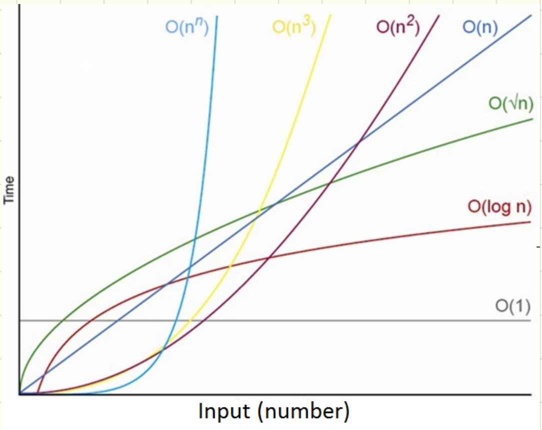
1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$
5. $O(n^2)$

   ⋮

   (polynomial)

And: $O(2^n)$

$O(n!)$



When these appear:

- For loop: often $O(n)$ $\sum_{i=1}^{n} 1 = \underbrace{1+\cdots+1}_{n}$ $= n$
- Nested for loops; ie:

```
for i ← 1 to n
    for j ← 1 to i
        total ← total + j
```

not $O(1)$

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{i} 1\right) = \sum_{i=1}^{n} i = 1+2+3+\cdots+n$$
$$= \frac{n(n-1)}{2} = O(n^2)$$

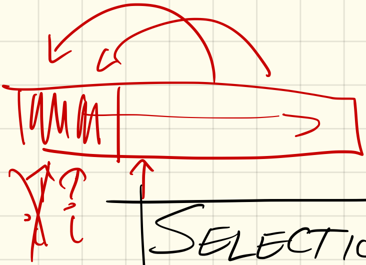
Both of these are examples of <u>iteration</u>. (ie using loops).

Common & useful!

Example: Sorting.

Sorting: input: n distinct integers $A[1..n]$ ← array

output: reordering of A into $B[1..n]$ s.t. $\forall i$, $B[i] < B[i+1]$



```
SELECTIONSORT(A, n):
    for i ← 1 to n
        j ← GETMIN(A, i, n)
        swap A[i] ↔ A[j]
    return A
```

$n-i$

```
GETMIN(A, first, last):
    index ← first
    for k ← first to last
        if A[k] < A[index]
            index ← k
    return index
```

$O(n-i)$

Correctness? At the end of iteration i of my loop, the ith element is in the correct spot.

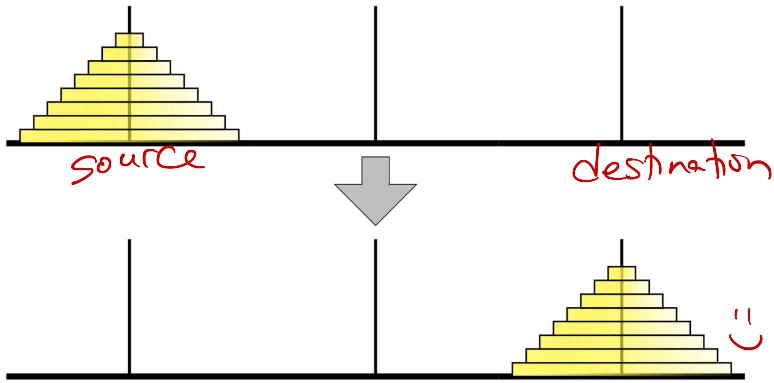
Runtime?

$$\sum_{i=1}^{n} (n-i) = \sum_{i=1}^{n} i = O(n^2)$$

Recursion : an algorithm that calls itself

Simple example: (2.5 in book)
Towers of Hanoi

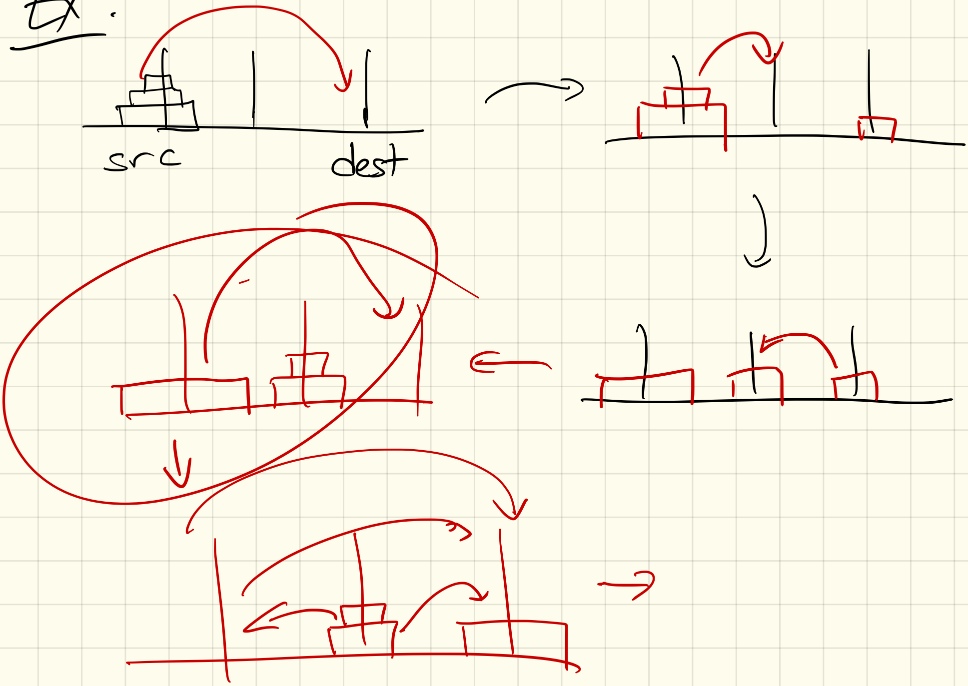
- 3 pegs & n disks, different sizes.
- Goal is to move disks from a source to destination peg, but only putting smaller pegs on larger ones



The Tower of Hanoi puzzle

How?

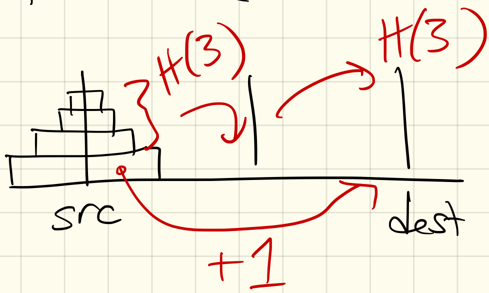
Ex:



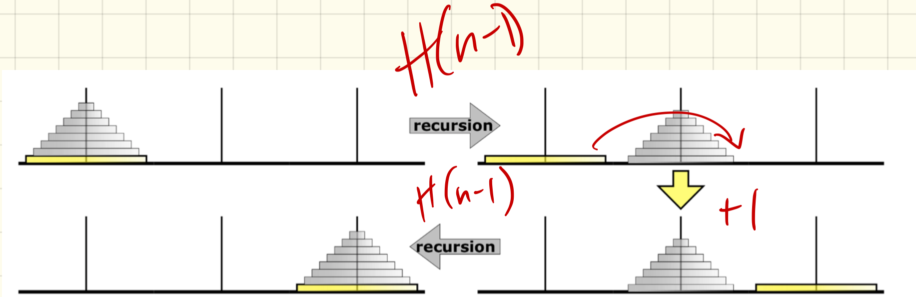
$H(3) = 7$

But stop for a minute:

How would we do 4?



$$H(4) = H(3) + 1 + H(3)$$



The Tower of Hanoi algorithm; ignore everything but the bottom disk

Recursive algorithm:

```
HANOI(n, src, dst, tmp):
  if n > 0
      HANOI(n − 1, src, tmp, dst)
      move disk n from src to dst
      HANOI(n − 1, tmp, dst, src)
```

Runtime? (# moves)

$$H(1) = 1$$

$$H(n) = H(n-1) + 1 + H(n-1)$$

$$= 2H(n-1) + 1$$

$$\rightsquigarrow 2^n - 1$$

exponential

Sometimes both recursion **and** iteration make sense:

Fibonacci numbers: $F_0 = 0$

$F_1 = 1$

$F_n = F_{n-1} + F_{n-2}$

$0, 1, 1, 2, 3, 5, 8, 13, \ldots$

2 ways to compute:

```
RecFib(n):
    if n=0 or n=1
        return n
    else
        return RecFib(n-1)
                 + RecFib(n-2)
```

```
IterativeFib(n):
    Create a blank array F[0..n]
    F[0] = 0
    F[1] = 1
    for i ← 2 to n
        F[i] ← F[i-1] + F[i-2]
```

$2 + \sum_{i=2}^{n} 1$

Compare:

- Both are correct
- Efficiency?

RecFib:

$$R(n) = \overbrace{R(n-1)} + R(n-2) + 1$$

$$= \overbrace{R(n-2) + R(n-3) + 1} + R(n-2) + 1$$

$$= 2R(n-2) + R(n-3) + 2$$

$$= O(\phi^n) \text{ exponential}$$

Iterative Fib: $O(n)$

Rest of Ch 2:

- More big-O examples
- Brief overview of types of algorithmic approaches:
  - exhaustive search ⇐ Ch 4
  - branch & bound ⇐ 3 paragraphs
  - greedy
  - dynamic programing
  - divide & conquer
  - ML
  - Randomized

(Useful to read, but I'll discuss these as we see bioinformatics examples in more detail.)

Ch 3: Molecular Biology Primer

(This was super useful for me - but I suspect you all know it.
Please skim, just so you know the terms I'll be using.)

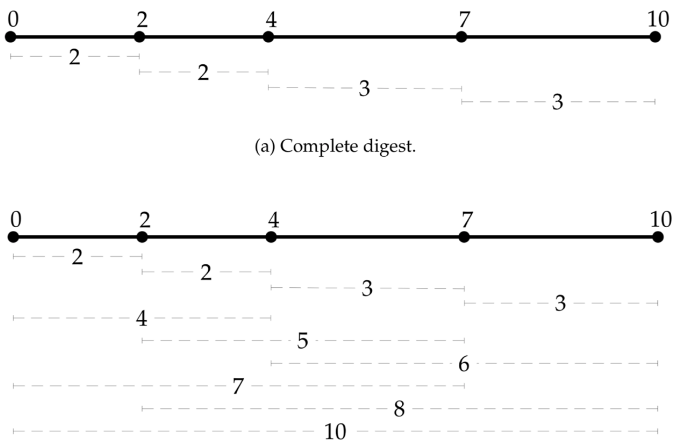
Also: Ch 2 + 3 are background for your first essay.
(due in 1 week)

Ch 4: Finally, some bio problems!

A first (if older) example:
DNA restriction mapping

Story: In 1970, Smith discovered how to break long DNA molecules at sites holding GTGCAC or GTTAAC.

Result: maps of these restriction sites, or *restriction maps*, become important.



(a) Complete digest.

(b) Partial digest.

Figure 4.1 Different methods of digesting a DNA molecule. A complete digest produces only fragments between consecutive restriction sites, while a partial digest yields fragments between any two restriction sites. Each of the dots represents a restriction site.

Turning this into a concrecte problem:

Partial digest problem (PDP):

Dfn: A multiset:

ex: $\{2,2,2,3,3,4,5\}$

$\{2_3, 3_2, 4, 5\}$

Dfn: If X is a set of n points on a line segment,

$$\Delta X = \{x_i - x_j : 1 \le i < j \le n\}$$

Aside: How big is ΔX?

$$\binom{n}{2} = \frac{n(n-1)}{2} = \frac{n!}{2!\,(n-2)!}$$

Ex: Let $X = \{0, 2, 4, 7, 10\}$.

$$\Delta X = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}$$

$$= \{2_2, 3_2, 4, 5, 6, 7, 8, 10\}$$

PDP: Given ΔX, reconstruct X.

Partial Digest Problem:
Given all pairwise distances between points on a line, reconstruct the positions of those points.

> **Input:** The multiset of pairwise distances L, containing $\binom{n}{2}$ integers.
>
> **Output:** A set X, of n integers, such that $\Delta X = L$

Aside: CS people also studied this!
(We called it the Turnpike problem.)

Note: These aren't unique!
Given a set A & value v,
let $A \oplus \{v\} = \{a+v : a \in A\}$
Then $\Delta(A + \{v\}) = \Delta A$

Ex: $A = \{0, 2, 4, 7, 10\}$
$A \oplus 100 = \{100, 102, 104, 107, 110\}$

In general, 2 sets A & B are called <u>homometric</u> if $\Delta A = \Delta B$.

Can show that if U & V are two sets of numbers,

$$U \oplus V = \{u+v : u \in U, v \in V\}$$

& $$U \ominus V = \{u-v : u \in U, v \in V\}$$

are always homometric.

<u>Ex</u>: $U = \{6, 7, 9\}$

$V = \{-6, 2, -6\}$

$U \oplus V$	-6	2	6
6	0	8	12
7	1	9	13
9	3	11	15

$U \ominus V$	-6	2	6
6	12	4	0
7	13	5	1
9	15	7	3

Both have $\Delta(U \oplus V) = \Delta(U \ominus V)$

$$= \{1_4, 2_4, 3_4, 4_3, 5_2, 6_2, 7_2, 8_3, 9_2, 10_2, 11_2, 12_3, 13, 14, 15\}$$

Note: PDP asks for one X, but biologists often want <u>all</u> X.

We'll always include 0.

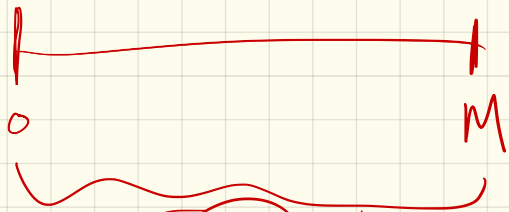
Brute force:

ΔX, size $\binom{n}{2}$

$|X|$

Ex at end of chapter

```
BruteForcePDP(L, n)
M ← maximum element in L
for every set of n − 2 integers 0 < x_2 < ··· < x_{n−1} < M
    X ← {0, x_2, ..., x_{n−1}, M}
    Form ΔX from X
    if ΔX = L
        return X
output "No Solution"
```



$M-1$ integers in between

Correctness:
Our alg. tries everything.

Runtime: $\binom{M-1}{n-2} = \frac{(M-1)!}{(n-2)!\,(M-1-(n-2))!}$

$\overset{?}{=} O(M^{n-2})$

(you can bound this a bit more carefully - see book)

Improved brute force:

Do we really need all items $\le M$?

Observation: If L does not contain the value y, then y can't be in X.

Why? Spps it were:

Result:

AnotherBruteForcePDP(L, n)

```
M ← maximum element in L
for every set of n − 2 integers 0 < x_2 < ··· < x_{n−1} < M from L
    X ← {0, x_2, ..., x_{n−1}, M}
    Form ΔX from X
    if ΔX = L
        return X
output "No Solution"
```

Runtime: $\binom{L}{n-2}$ & $L \ll M$

Correctness: trying everything

Next time:
- A more practical approach
- & then on to motif finding