

ECE697S: Topics in Computational Biology

Lecture 2: Introduction to
Computation



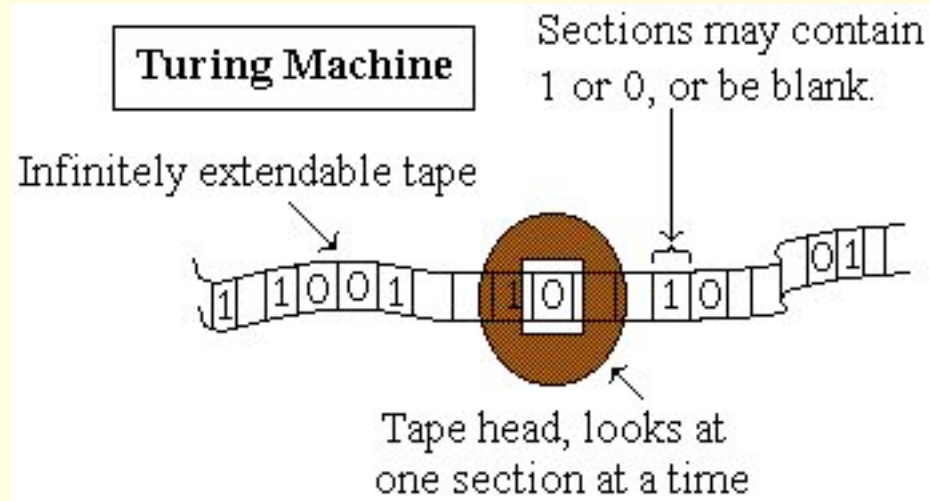
The word *algorithm* comes from the name of the 9th century Persian mathematician Abu Abdullah Muhammad bin Musa al-Khwarizmi. The word algorism originally referred only to the rules of performing arithmetic using Hindu-Arabic numerals but evolved via European Latin translation of al-Khwarizmi's name into *algorithm* by the 18th century. The word evolved to include all definite procedures for solving problems or performing tasks.

History of Computation

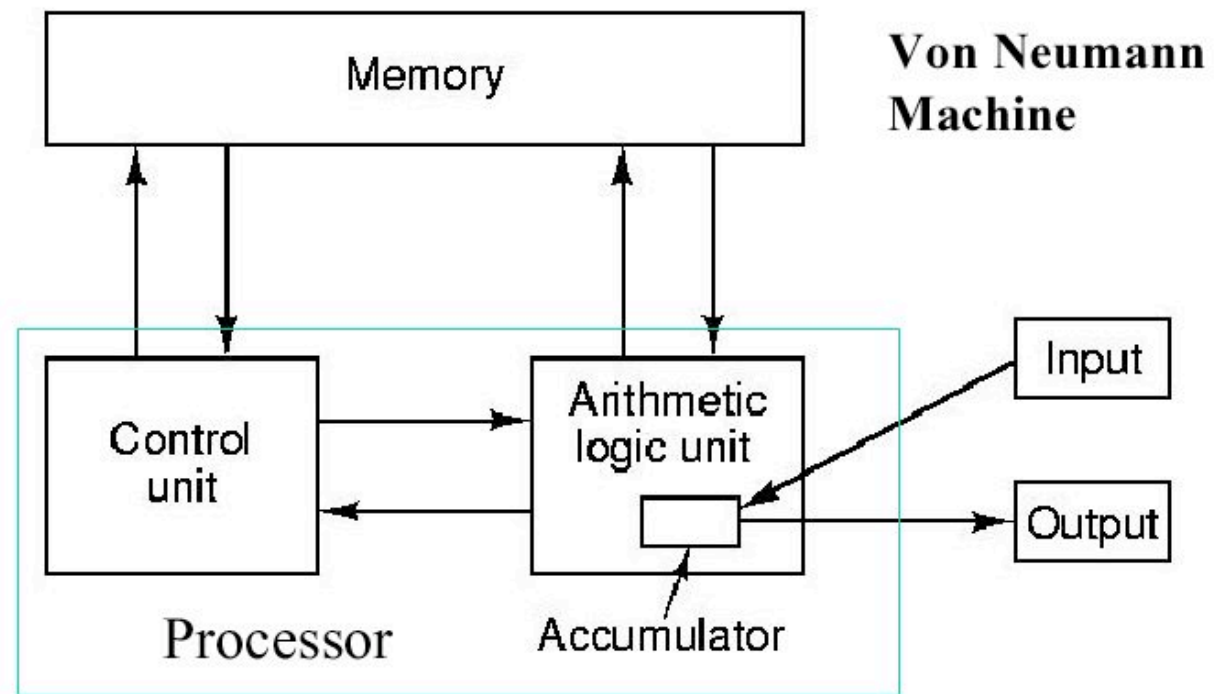
- Euclid (325BC - 270BC)
- Leibniz (mid 1600s)
- Jacquard (early 1800s)
- Lovelace/Babbage (1820-30s)
- Boole (1840s)
- Turing (1930-40s)
- Kilby/Noyce (late 1950s)
- Hardware: Colossus, ENIAC, ..., personal computers



Alan Turing (1912-1954)



John von Neumann (1903-1957)



Computational Framework

- Computable = Turing-computable
- Modern programming languages are “Turing-complete”.
- Primitive operations: +, -, /, *, if/then, loops

Algorithm Analysis

- Does our procedure always provide a “correct” answer?
 - Deterministic: Always
 - Randomized: Probably, e.g., 95% of the time
- How quickly does our procedure provide the answer?
 - Deterministic: x steps, at most
 - Randomized: x steps, usually

Upper Bounds

- We'd like to say “Algorithm A never takes more than $f(n)$ steps for an input of size n ”
- “*Big-O*” Notation gives **worst-case**, i.e., **maximum**, running times.
- A correct algorithm is a constructive upper bound on the complexity of the problem that it solves.

Lower Bounds

- Establishes the **minimum** amount of time needed to solve a given computational problem.
- Note that such arguments also classify the problem; need not be constructive!

Toy Problem #1

- Compute the sum of $1 \dots n$?
- Running Time?

Toy Problem #2

- Does n have any factors, other than 1 and n ?
- Running time is ?

Running Times

- We must be careful -- running time depends on input size, not how the input is used!
- Can we think of better algorithms?
 - For sum: use Gauss' formula
 - For primality: first poly-time algorithm was discovered in 2002!

Searching Ordered Lists

- Given an ordered list of integers, $A[1..n]$, is x in the list?
- Simple algorithm?

Searching Ordered Lists

- More complicated algorithm stems from a guessing game.
- “Binary Search”

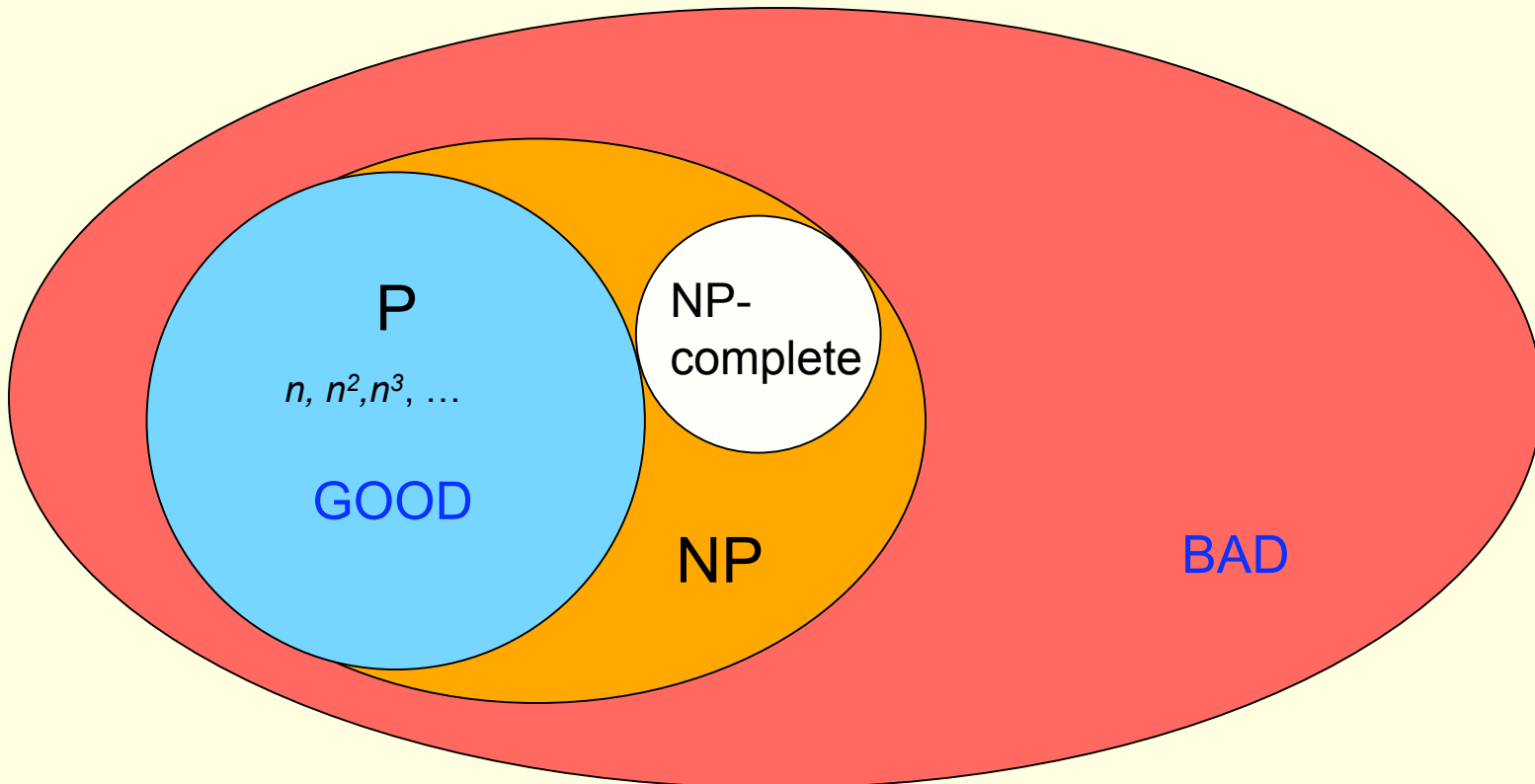
Can we do better?

- In the worst case, no. I can always choose x so that $\log n$ steps are required.
- Thus, we have “tight” bounds.

Why Polynomial Time?

- Suppose an algorithm requires $O(2^n)$ time, for an input of size n .
- Now, what is the largest input our algorithm process in a “reasonable” amount of time?
- It depends on the machine, but the number of atoms in the universe is theorized to be around 2^{240} .

Complexity Classes



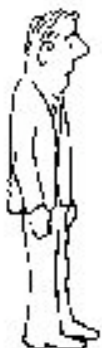
set of all *Turing-computable* problems

Does $P = NP$? We need lower bounds...

NP-completeness

2

COMPUTERS, COMPLEXITY, AND INTRACTABILITY



I can't find an efficient algorithm. I guess I'm just too dumb.



I can't find an efficient algorithm, but neither can all these smart people.

NP-complete problems = problems no one can find polynomial-time algorithms for.

Some NP-complete problems

- **Traveling Salesman:** Given a map, find the shortest route through all n cities that visits every city exactly once.
- **Side-chain packing:** Given a protein with n amino acids, find the conformations of side-chain atoms that yield minimum global energy.

Algorithmic Strategies

- Greedy: “Best-first” approach
- Recursion: Problem instance can be decomposed into smaller instances of same problem
- Dynamic Programming: Like recursion, but explores *all* combinations of “decompositions”.

Recursion

- Often times a problem instance can be solved by decomposing it into smaller versions of the same problem!
- E.g. $\text{sum}(n) = \text{sum}(n-1) + n$
- E.g.: MergeSort

Optimization Problems

- We want to find an answer that maximizes or minimizes some measure.
- Example: Google returns most “authoritative” pages first.
- Example: Given a map, find the shortest path between two points.

Graphs

- $G = (V, E)$, V = nodes/vertices, E = relationships with pairwise edge costs/weights.
- Relationships such as similarity, dissimilarity, distance, etc. can be encoded numerically.
- Graphs are just matrices; so we can compute properties of graphs.

Examples of Graphs

- World Wide Web
- Computer Networks
- Sequence Phylogenies
- Protein Structure
- Protein Interactions

Spanning Trees

- Suppose we had a weighted graph, what is the minimum weight “skeleton”, or tree?
- “Minimum Spanning Tree” can be found greedily.
- Running Time: $O(mn)$, can be made more efficient!

Dynamic Programming

- Problems that can be solved by dynamic programming have a *local optimality* property.
- Dynamic Programming usually involves generating a table of costs of various subproblems, and finding the optimal combination of costs.

Edit Distance

- A **string** is a sequence of characters drawn from some alphabet.
- $D(s_1, s_2)$ = the minimum number of **edits** needed to convert **string** s_1 into **string** s_2 .
- An **edit** is considered to be an insertion, deletion, or substitution.

Edit Distance

- Where else do we see insertions, deletions and substitutions?
- The edit distance between two gene sequences can be viewed as a type of *global* alignment.

Problems on Sequences

- We can modify the penalties for the edits to be “realistic”.
- Longest Common Subsequence -- NP-complete!
- “Optimality” can be defined in a number of ways (BLAST, Needleman-Wunsch, Smith-Waterman, ...).