Unit #2: Complexity Theory and Asymptotic Analysis

CPSC 221: Algorithms and Data Structures

Lars Kotthoff[1]
larsko@cs.ubc.ca

[1]With material from Will Evans, Steve Wolfman, Alan Hu, Ed Knorr, and Kim Voll.

Runtime Example #8: Longest Common Subsequence

Problem: Given two strings (A and B), find the longest sequence of characters that appears, in order, in both strings.

Example:

$A = \texttt{search me}$ $B = \texttt{insane method}$

A longest common subsequence is "`same`" (so is "`seme`")

Applications:
DNA sequencing, revision control systems, `diff`, ...

Example #9

Find a tight bound on $T(n) = \lg(n!)$.

Aside: Who Cares About $\Omega(\lg(n!))$?

Can You Beat $O(n \log n)$ Sort?

Chew these over:

- How many values can you represent with n bits?
- Comparing two values ($x < y$) gives you one bit of information.
- There are $n!$ possible ways to reorder a list. We could number them: $1, 2, \ldots, n!$
- Sorting basically means choosing which of those reorderings/numbers you'll apply to your input.
- How many comparisons does it take to pick among $n!$ numbers?

Asymptotic Analysis Summary

- Determine what is the input size
- Express the resources (time, memory, etc.) an algorithm requires as a function of input size
  - worst case
  - best case
  - average case
  - common case
- Use asymptotic notation, O, Ω, Θ, to express the function simply

Problem Complexity

The **complexity of a problem** is the complexity of the best algorithm for the problem.

- We can sometimes prove a lower bound on a problem's complexity. To do so, we must show a lower bound on any possible algorithm.
- A correct algorithm establishes an upper bound on the problem's complexity.

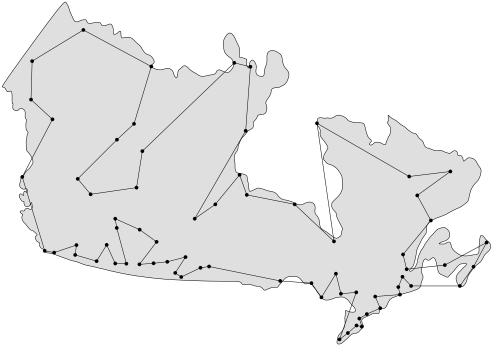
Searching an unsorted list using comparisons takes $\Omega(n)$ time (lower bound).
Linear search takes $O(n)$ time (matching upper bound).

Sorting a list using comparisons takes $\Omega(n \log n)$ time (lower bound).
Mergesort takes $O(n \log n)$ time (matching upper bound).

Problem Complexity

Sorting: solvable in polynomial time, tractable
Traveling Salesman Problem (TSP): In 1,290,319km, can I drive to all the cities in Canada, visiting each exactly once, and return home? www.math.uwaterloo.ca/tsp/

Checking a solution takes polynomial time. Current fastest way to find a solution takes exponential time in the worst case.



Are problems in NP really in P? $1,000,000 prize

Problem Complexity

Searching and Sorting: P, tractable
Traveling Salesman Problem: NP, intractable[2]
Kolmogorov Complexity: Uncomputable

Kolmogorov Complexity of a string is the length of the shortest description of it.

Can't be computed. Pithy but hand-wavy proof: What's:

> *The smallest positive integer that cannot be described in fewer than fourteen words.*

[2] Assuming $P \neq NP$.