

CMsc 451 - Algorithm Design

Lecture 1 - Introduction

About this course -

- Second course in algorithm design (after 351)
- Fundamental elements of algorithm design
 - design techniques (greedy, divide + conquer, ...)
 - proving correctness
 - analyzing running times
- Theoretical focus - no programming projects

Overview:

- Graph basics, DFS, + shortest paths
- Greedy algorithms
- Dynamic programming
- Network flows
- NP-Hardness + Approximation algorithms

Prerequisites - (CMsc 351)

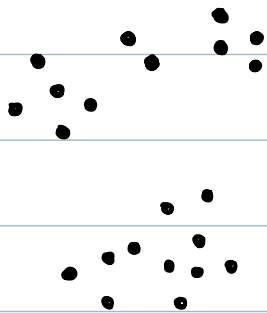
- Basic programming
- Discrete math - induction, sets, probability, ...
- Sorting + basic data structures
- Math - logs + exponentials, calculus, linear algebra

Algorithm Design:

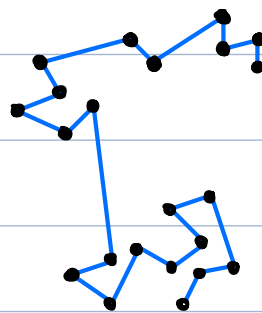
Given a well defined computational problem design an efficient procedure for solving it.

Example 1: Given a set of points in the plane compute a path of min length that visits all the points.

Input:

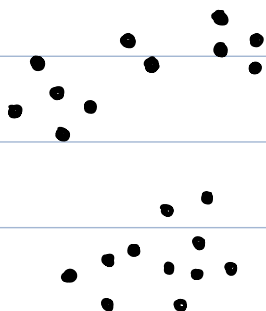


Output:

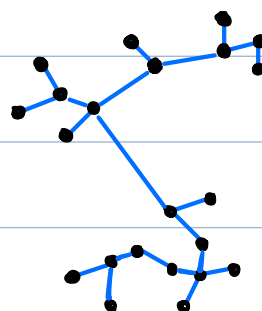


Example 2: Given a set of points in the plane compute a network of min length that connects all the points.

Input:



Output:



These are superficially quite similar, but computational complexities are quite different.

Example 1: Euclidean Traveling Salesman Problem

- NP-Hard

- Can be efficiently approximated

Example 2: Euclidean Minimum Spanning Tree

- Solvable exactly in $O(n^2)$ time

- Practical approximation in $O(n \log n)$ time

Measures of computational complexity -

- Running time

- Space

... as a function of input size (denoted n)

But many inputs of same size...

- Worst-case - max over all inputs of size n

- Average-case - expected case for inputs of size n from some distribution

→ More realistic, but a lot harder!

Asymptotic Notation:

- Simplify complex functions
- Focus on trend for large n
- Ignore constant factors

Example:

$$T(n) = 3.9n + 4.17 \cdot n \log n + 3.5n^2$$

$$\approx 3.9n + 4.17 \cdot n \log n + 3.5n^2 \quad - \text{large } n$$

$$\approx \cancel{3.9n} + \cancel{4.17 \cdot n \log n} + 3.5n^2 \quad - \text{ignore const factors}$$

$$\approx \cancel{3.5n^2}$$

$T(n)$ is $O(n^2)$ "on the Order of"

↳ Usually written $T(n) = O(n^2)$ but not formally correct. (But we'll do it anyway)

" O " is shorthand for asymptotically " $\leq$ "

Also:

Θ $\leftrightarrow$ " $=$ " Big theta

o $\leftrightarrow$ "<" Little o

Ω $\leftrightarrow$ " $\geq$ " Big Omega

ω $\leftrightarrow$ ">" Little omega

Important Complexity Classes - For some constant $c > 0$

- $\leq (\log n)^c = \log^c n$ - Polylogarithmic time
- $\leq n^c$ - Polynomial time
- $\leq c^n \quad (c > 1)$ - Exponential time

Asymptotic Ordering - for any $a, b, c > 0 \quad (c > 1)$

$$\log^a n \leq n^b \leq c^n$$

Summary:

- Basics of algorithm design
- Asymptotic Notation