- A well-specified problem
- problem vs. Instance
- Algorithm: input → output
  - correct
  - Efficient
  - Easy to Implement

1.1 Robot Tour Optimization

Input: A set S of n points
Output: Shortest cycle tour that visits each point

- nearest-neighbor heuristic
  $P_0 \longrightarrow P_1$ wrong.
  nearest.
- ClosestPair(P) heuristic
- OptimalTSP: $n!$ permutations

algorithm ≠ heuristic

1.2 Selecting the Right Jobs

Input: A set I of n intervals
Output: largest subset of mutually non-overlapping intervals.

- EarliestJobFirst(I)
- ShortestJobFirst(I)
- ExhaustiveScheduling(I): 2^n subsets
- OptimalScheduling(I) — earliest completion time

algorithm correctness needs careful examination.

1.3 Reasoning about Correctness

Proof: demonstration

1. precise statement
2. set of assumptions
3. chain of reasoning
4. QED

correctness not incorrectness

1. Problems and Properties

Problem Specification:

1. A set of allowed inputs — narrow
2. required properties of outputs.
   - not ill-defined Qs
   - don't create compound goals for simplicity

2. Expressing Algorithms.

- English (w/ Pictures)
- Pseudocode.
- Programming language

Goal: Clarity of IDEA
— so not too low level to describe it

3. Demonstrating Incorrectness

Counterexamples:

1 — Verifiability — be able to calculate, then display a better ans.

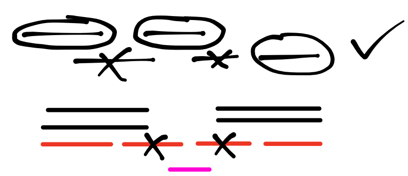
2 — Simplicity — essence

Techniques to find c.e.:

1. Think Small
2. Think Exhaustively
3. Hunt for weakness
4. Go for a tie
5. Seek extremes

Ex 1.2 Greedy Heuristic:

possible soln: select interval i overlaps the least number of intervals. remove the overlapping ones



1.4 Induction & Recursion

- base case
- assume good to n-1
- prove true for n using assumptions

Ex Insertion Sort.

- base case: one-element.
- assume n-1 completely sorted
- insert last element: putting it in spot smaller/equal x bigger/equal

Careful:

- Boundary errors
- cavalier extension claims: one more instance could change optimal soln.

1.5 Modeling the Problem

Modeling: most important step
- Relating problem to what has already been done
- Describe problem ABSTRACTLY: structures like permutations, graphs, sets..

①. Combinatorial Objects

Formulate in terms of fundamental structures:

1- Permutations. - arrangement.
2- Subsets - selections from a set, ~~order~~
3- Trees - hierarchy
4- Graphs - relationships between objects
5- Points - locations in a space
6- Polygons - regions in space
7- Strings - sequences, patterns

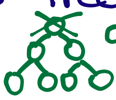
Caution: Modeling is constraining, and may be done in different ways too

②. Recursive Objects
- decomposition rules
- basis cases

Learn to think recursively:
big things are made of smaller things of the exact same type
Delete a part

1- Permutation
$n \to n-1$ things by deleting, renumbering

2- Subsets
$\{1, \ldots, n\}$ contains $\{1, \ldots, n-1\}$

3- Trees

collection of smaller trees

4- Graphs


5- Point sets

2 smaller clouds

6- Polygons
Inserting internal chord. - triangle is the basis case

7- Strings
Alex → A | lex

1.6 Proof by Contradiction

- Assume Hypothesis is false
- develop Consequences if false
- Show consequence is false

Ex Euclid's proof
there are an infinite number of prime numbers: integers ∩ that have no non-trivial factor

1.9 Estimation

Principled Guessing:
1- Principled calculations
- a function you already know

2- Analogies
- past experience.

solve problem in different ways and see if they generally agree in magnitude (ratio 2-10)