Algorithm Design and Analysis

shereen@pacificu.edu

What is an Algorithm?

- A sequence of computational steps that transforms the *input* into the desired *output*.
- To be interesting, an algorithm has to solve a general, specified problem. An algorithmic problem is specified be describing the set of *instances* that it must work on and the desired properties of the output.

Performance

- Algorithms is the study of computer-program performance
- What is more important than performance in computer programs?
  - o
  - o
  - o
  - o
  - o

Why Study Algorithms?

-
-
-
-
-
-
-

Example: Sorting

- **Input:** A sequence of n numbers $<a_1, a_2, \ldots, a_n>$
- **Output:** A permutation (reordering) $<a'_1, a'_2, \ldots, a'_n>$ of the input sequence such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$
- We seek algorithms that are *correct* and *efficient*

Correctness

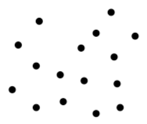
- For any algorithm, we must prove that it *always* returns the desired output for all legal *instances* of the problem.
- What does this mean for sorting?

Correctness is Not Obvious!

- Suppose you have a robot arm equipped with a tool, say a soldering iron. To enable the robot arm to do a soldering job we must construct an ordering of the contact points so the robot visits (and solders) the first contact point, then visits the second point, third, and so forth until the job is done.
- Since robots are expensive, we need to find the order which minimizes the time (ie. travel distance) it takes to assemble the circuit board.

Correctness is Not Obvious!

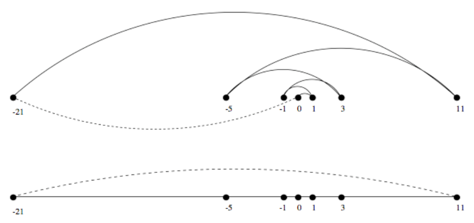
- You are given the job to program the robot arm. Give me an algorithm to find the best tour.



Nearest Neighbor Tour

- A very popular solution starts at some point p_0 and then walks to its nearest neighbor p_1 first, then repeats from p_1, etc. until done.
- Pick and visit an initial point p_0
- $p = p_0$
- $i = 0$
- While there are still unvisited points
  - $i = i + 1$
  - Let p_i be the closest unvisited point to p_{i-1}
  - Visit p_i
- Return to p_0 from p_i

Nearest Neighbor Tour

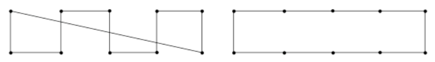


Closest Pair Tour

- In this case we repeatedly connect the closest pair of points whose connection will not cause a cycle or a three-way branch to be formed, until we have a single chain with all the points in it.

```
Let n be the number of points in the set
d = ∞
For i = 1 to n-1 do
    For each pair of endpoints (x, y) of partial paths
        If dist(x, y) ≤ d then
            x_m = x, y_m = y, d = dist(x, y)
    Connect (x_m, y_m) by an edge
Connect the two endpoints by an edge
```

Closest Pair Tour



- So, is there a correct algorithm to solve this problem?

A Correct Algorithm

Expressing Algorithms

- What are the possible ways to express an algorithm?
  - English
  - Pseudocode
  - Programming Language

The RAM Model

- Algorithms can be studied in a machine and language independent way.
- Each “simple” operation (+, -, =, if, call) takes exactly one step.
- Loops and subroutines are not simple operations.
- Each memory access takes one step.

Best, Worst, and Average-Case

- Worst case: is the function defined by the maximum number of steps taken on any instance of size n.
- Best case: is the function defined by the minimum number of steps taken on any instance of size n.
- Average case: is the function defined by an average number of steps taken on any instance of size n.

Insertion Sort

- INSERTION-SORT(A,n) ∇ A[1..n]

```
for j ← 2 to n
 do key ← A[j]
  ∇ Insert A[j]
  i ← j - 1
  while i > 0 and A[i] > key
   do A[i+1] ← A[i]
     i ← i - 1
  A[i+1] ← key
```

Example

- How would insertion sort work on the following numbers?
  - 3 1 7 4 8 2 6

Your Turn

- Problem: How would insertion sort work on the following characters to sort them alphabetically (from A -> Z)? Show each step.
  - S O R T E D

Insertion Sort

- Is the algorithm correct?
- How efficient is the algorithm?
- How does insertion sort do on sorted permutations?
- How about unsorted permutations?

Analysis of Insertion Sort

- Best Case

Analysis of Insertion Sort

- Worst Case

For Next Time

- Read Chapters 1 and 2 from the book.