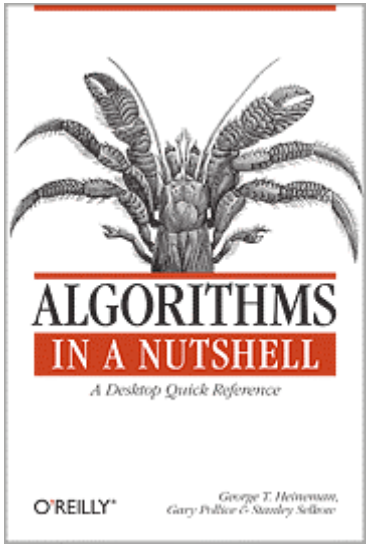
Algorithms in a Nutshell



Session 9

Recap Algorithms and SE

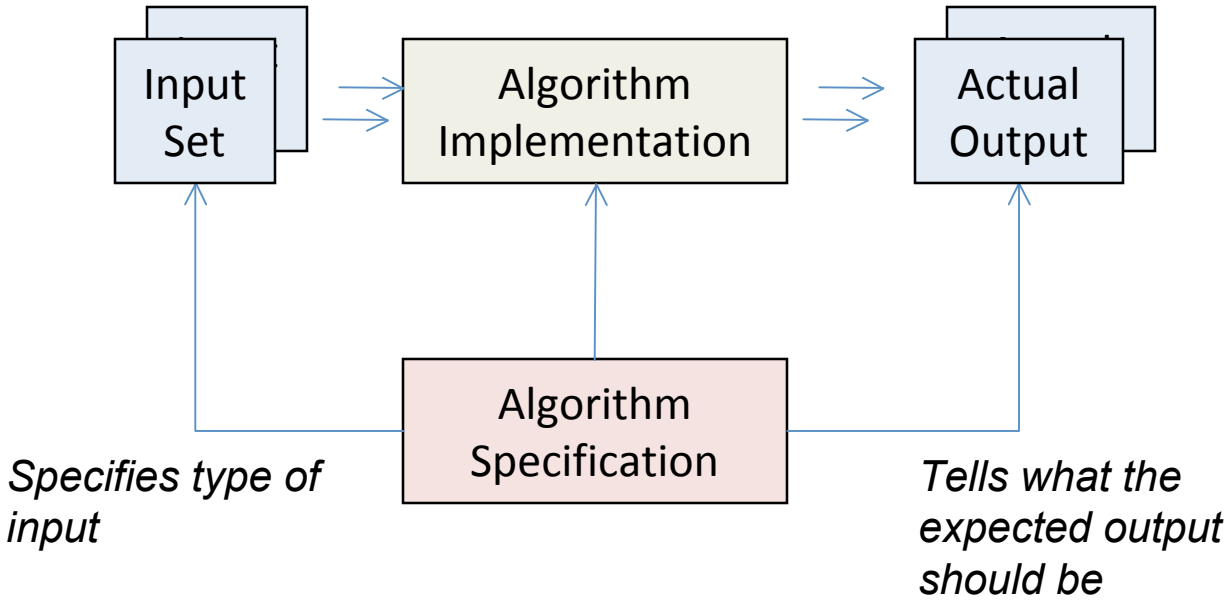
3:50 – 4:20

Outline

- Unit testing
  - Challenges, specifically for algorithms
  - Importance of visualization
- Stress testing
  - Performance testing
- Concepts
  - Floating point values

Unit Testing

- Given specification for a unit of code, validate implementation against a set of inputs



Brute Force Approach

- Many algorithms solve problems for which a Brute Force algorithm exists
  - FORD-FULKERSON (Chapter 8)
- Simply execute both algorithms on same data set(s)
  - Assumes you can generate random uniform inputs
  - Unlikely that both implementations will share same defect
- Also compare algorithms to textbook solutions

Validating algorithm implementations

- Assume algorithm is correct
  - How do we validate the implementation?
- In this book, we had several simple cases
  - Chapter 4: Sorting
  - Chapter 5: Searching
- Two specific challenges to face
  - What if you can only verify output by computing all possible solutions?
  - What if you can't predict the output in advance?

Validating Graph Algorithms

- How do you validate DIJKSTRA'S ALGORITHM?
  - single-source shortest path
  - Impossible to compute Brute Force approach
- Rely on examples from text books
- Coding technique

```
int v = ci->first;
long newLen = dist[u];
newLen += ci->second;
if (newLen < dist[v])
```

Math overflow protection

```
int v = ci->first;
int newLen = dist[u];
newLen += ci->second;
if (newLen > 0 && newLen < dist[v])
```

extra check for negative values

Validating AStarSearch (Chapter 7)

- How do you validate heuristic search?
  - Heuristics not guaranteed to find answer!
- Relied on Artificial Intelligence literature
  - Examples found in books, lecture notes
- Canonical example domains
  - Tic Tac Toe for search trees
  - The sliding 8-puzzle for game trees
- Overall strategy
  - Try to replicate examples from other textbooks

Validating Floating Point Algorithms

- How do you validate imprecise calculations?
- Line Segment intersection
  - $O(n^2)$ Brute Force Algorithm exists
  - Output differs (at strange and inopportune times)
  - floating-point values on the order of 10^{-15} which should otherwise be treated as zero

```
(424.2213883396885,83.64382123041435,430.21898827125784,83.4741301556963)
(424.27530553919314,85.91543442004598,424.2754719964361,79.91543442235498)
(429.8421554463252,85.39816284280639,424.56541286695244,82.54229384177802)
```

How to discover these segments?

- Randomly generated input sets of 100 line segments
  - Found one set that produced different results
- Manually removed line segments until error disappeared
  - Divide and conquer
  - Eventually was able to produce minimal set

Unit Testing Concepts

- Separate Interface from implementation
  - `algs.model.searchtree.INode` (Search Trees)
  - `algs.model.IBinaryNode` (Binary Trees)
- C examples
  - Standardize drivers for algorithms
  - Code/Sorting/buildPointerBasedInput.c (Sorting)
  - Timing/timing.c

Visualization

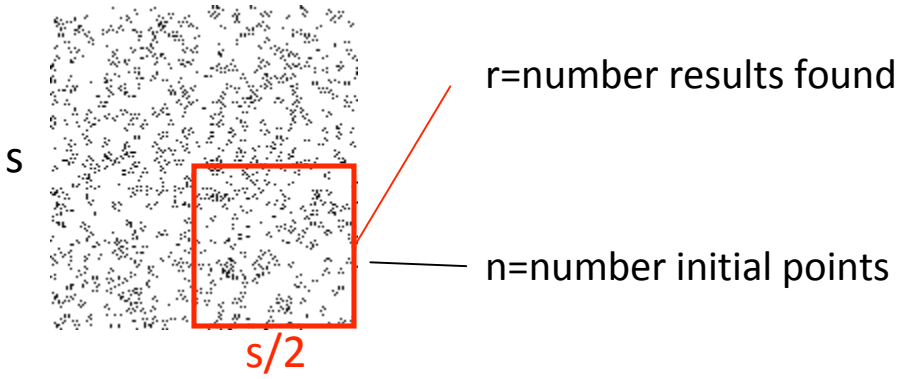
- GraphViz used extensively throughout ADK
  - www.graphviz.org
  - Automatically visualize execution of algorithms
  - Avoids error-prone by-hand computed examples
- Infrastructure developed
  - `algs.debug` package
  - Parallel development: `algs.model.searchtree` and `algs.model.searchtree.debug`

Performance Testing

- Infrastructure to time algorithm execution
  - Monitor performance time as input size doubles
  - Standard approach used throughout book
- Standard machine platforms used throughout
  - Dedicated Linux cluster machine (i.e., only user)
  - Windows desktop PC (Windows XP)
- Encourage experimentation
  - Install ADK and follow instructions for executing your own experiments

Performance Testing Challenge

- RANGE QUERY algorithm proved interesting
  - Uses d-dimensional KD-trees (Chapter 9)
  - Two interrelated factors (n=number points, r=number results found)



As you apply to higher dimensions, keeping query box at s/2 will return fewer and fewer points, which skews timing results; you actually need to increase the size of query