

For Friday

- Read Weiss, chapter 10, section 2
- No homework
- Program 4 due

Program 4

- Any questions?

Research Paper

- Any questions?

Difficulty Levels

- Undecidable
 - Example is halting problem
- Intractable
 - Example is ?
- Exponential problems are considered intractable. Why?

Another Class of Intractable Problems

- Polynomial problems are considered tractable.
- What does NP mean?

Non-deterministic Polynomial

- A deterministic machine must always make a single choice.
- Suppose you had a non-deterministic computer.
- Then you could “pick” all of the different choices at once (or automatically pick the best solution).

The Class NP

- Can determine that a solution is the correct solution in polynomial time.
- All problems with polynomial time solutions fit into this class.
- Some decidable problems do not. Consider problems for which the solution is of exponential length.

The Big Question

- Are there problem in NP that are not in P?
- Brings us to the class of NP-complete and NP-hard problems.
- NP-complete problems are reducible to one another.

Examples

- Traveling Salesman
- Hamiltonian Cycle
- Satisfiability (technically, 3Sat)
- Graph coloring
- Knapsack
- Bin packing

Algorithm Design

- How do you design an algorithm?
- We're going to talk in these last weeks about some different strategies for designing algorithms.
 - Greedy algorithms
 - Divide and Conquer algorithms
 - Dynamic Programming
 - Randomized
 - Backtracking

Algorithm Design Methods

- For each design method, we'll ask:
 - What is the strategy?
 - When is the strategy applicable?
 - What is the general complexity of algorithms designed using this strategy?
- We'll also look at several algorithms that fall into the design category.

Optimization Problems

- In many problems, there are two types of solutions
 - **feasible** solutions
 - **optimal** solutions
- These types of problems are called **optimization** problems

Applying Algorithms to Optimization Problems

- When applying an algorithm design method to a problem, we may get
 - an unacceptable solution
 - a feasible solution that is clearly not optimal
 - a feasible solution that is close to optimal
 - an optimal solution

Greedy Algorithms

- With the greedy method, at each step of the algorithm we make the decision that appears best at that time.
- We must always define a **greedy criterion** that determines the best decision.
- Decisions are never changed.
- Each decision must take feasibility of the final solution into account.

Making Change

Examples

- Making Change
- Shortest Path
- Topological Sort
- Minimum Spanning Tree
- Bin Packing
- Job Scheduling
- Huffman Code

Bin Packing

- What's the problem?

On-Line Strategies

- Next Fit
- First Fit
- Best Fit

Off-Line Strategies

- First Fit Decreasing
- Best Fit Decreasing

Job Scheduling

- What's the problem?
- What is an optimal solution?
- Can we achieve an optimal solution using a greedy algorithm?

Greedy Heuristic

- Assume we want to minimize average completion time: how can we do that?
- Assume we want to minimize final completion time. How would we do that?

Huffman Codes

- File compression technique
- Greedy algorithm is optimal