Greedy Algorithms

Introduction

Algorithms: Design and Analysis, Part II

Algorithm Design Paradigms

Algorithm Design: No single "silver bullet" for solving problems.

Design Paradigms:

- Divide & conquer (see Part I)
- Randomized algorithms (touched in Part I)
- Greedy algorithms (next)
- Dynamic programming (later in Part II)

Greedy Algorithms

"Definition": Iteratively make "myopic" decisions, hope everything works out at the end.

Example: Dijkstra's shortest path algorithm (from Part I)
- Processed each destination once, irrevocably.

Contrast with Divide & Conquer

1. Easy to propose multiple greedy algorithms for many problems.

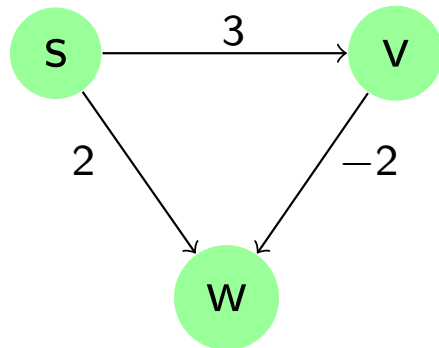
2. Easy running time analysis.
   (Contrast with Master method etc.)

3. Hard to establish correctness.
   (Contrast with straightforward inductive correctness proofs.)

DANGER: Most greedy algorithms are NOT correct. (Even if your intuition says otherwise!)

In(correctness)

Example: Dijkstra's algorithm with negative edge lengths. What does the algorithm compute as the length of a shortest s-w path, and what is the correct answer?



A) 2 and 2 C) 1 and 2

B) 2 and 0 D) 2 and 1

Proofs of Correctness

Method 1: Induction. ("greedy stays ahead")

Example: Correctness proof for Dijkstra's algorithm. (See Part I.)

Method 2: "Exchange argument".

Example: Coming right up!

Method 3: Whatever works!