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.

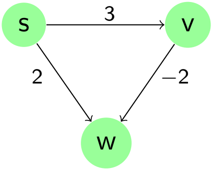
2. Easy running time analysis.
   (Contrast with Master method etc.)

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

Greedy Algorithms

Application: Optimal Caching

Algorithms: Design and Analysis, Part II

The Caching Problem

Small fast memory (the cache).

Big slow memory.

Process sequence of “page requests”.

On a “fault” (that is, a cache miss), need to evict something from cache to make room – but what?

Example

Cache:

~~a~~	~~b~~	c	d
e	f		

Request sequence: c d e f a b

$\Rightarrow$ 4 page faults
- 2 were inevitable (e & f)
- 2 consequences of poor eviction choices (should have evicted c & d instead of a & b)

The Optimal Caching Algorithm

Theorem: [Bélády 1960s] The "furthest-in-future" algorithm is optimal (i.e., minimizes the number of cache misses).

Why useful?

1. Serves as guideline for practical algorithms (e.g., Least Recently Used (LRU) should do well provided data exhibits locality of reference).
2. Serves as idealized benchmark for caching algorithms.

Proof: Tricky exchange argument. Open question: Find a simple proof!

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!