CS472 Module 7 Part B - Greedy Algorithms

Athens State University

March 31, 2014

Outline

Contents

1 Definition **1**

2 Example: Huffman Codes **3**

3 Approximation: Greedy Traveling Salesmen **4**

4 Key Points **5**

1 Definition

What is a "greedy algorithm"?

- A meta-heuristic that says to make the best choice at particular time in the hope that will lead you to the best overall solution
- Attempting to solve an optimization problem piece by piece by stepping through a sequence of choices that are
  - feasible
  - locally optimal
  - irrevocable
- Sometimes leads to the right answer, but can give a fast approximation even when they don't give you the best solution

Let's reconsider spanning trees

- Prim's Algorithm: at each iteration, chose the vertex not in the tree that is closest to those vertices already in the tree
- Kruskal's Algorithm: at each step, select the edge with smallest weight remaining in the set of edges
- Dijkstra's Algorithm: Find the vertex not already in the tree that adds the smallest sum to the path length

Reconsider the change-making algorithm

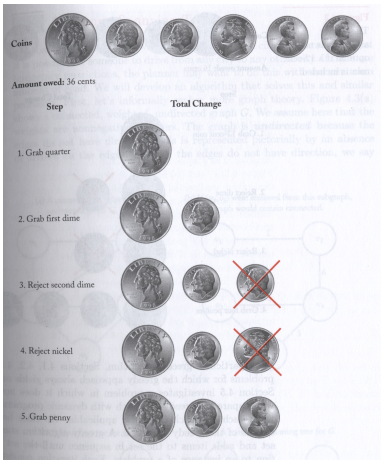
- For certain cases, a greedy approach makes more sense than using dynamic programming
- At each step, select as many of the largest denomination remaining in the set of coins
- Consider what would happen if you only had 25-cent, 10-cent, and 4-cent coins
  - How would the greedy algorithm attempt to make change for 41 cents?
  - What would be the correct combination of coins to use?

Greedy-solution to change-making

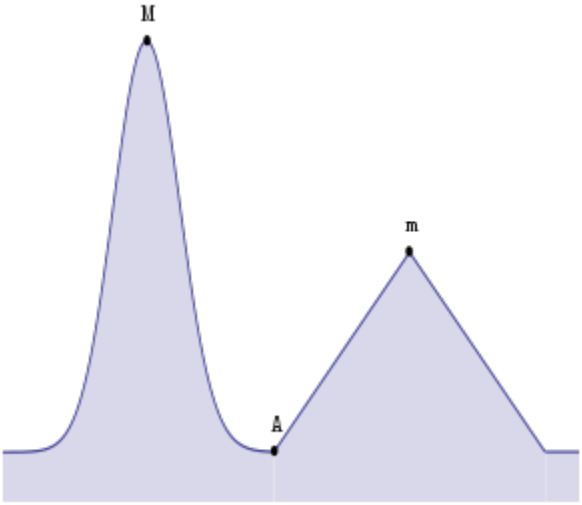
Algorithm 1: Change-making(greedy): minimize total number of coins returned as change by a vending machine

Input: Amount of change to be returned
Output: The smallest possible set of coins whose denominations sum to that amount
while *there are more coins and have not reached change amount* **do**
    Grab the largest remaining coin;
    if *Adding coin exceeds amount being returned* **then**
        Reject the coin;
    else
        Add the coin to change being returned;
    if *Total value of the change equals the amount owned* **then**
        Return the set of coins;

Greedy Change-Making Algorithm: Example



Why greedy algorithms often fail to find the optimal solution?



2 Example: Huffman Codes

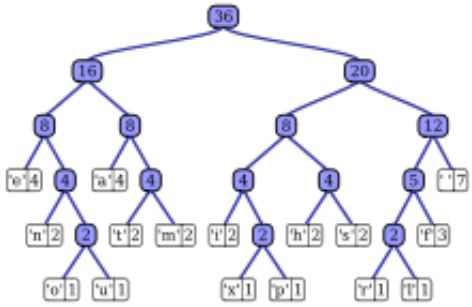
Example: Huffman Codes: Coding problem

- *Coding problem*: assignment of bit strings to alphabet characters
- *Codewords*: bit strings assigned for characters of alphabet
- Two types of codes:
  - *fixed-length encoding*: ASCII, EBCDIC
  - *variable-length encoding*: Morse code
- *Prefix-free codes*: no codeword is a prefix of another codeword
- Problem: if frequencies of character occurrences are known, what is the best binary prefix-free code?

Example: Huffman Codes

- Any binary tree with edges labeled with 0 or 1 yields a prefix-free code of characters assgined to its leaves
- Can construct an optimal binary tree minimizing the expected (weighted average) length of a codeword using Huffman's Algorithm
  - Initialize n one-node trees with alphabet characters and the tree weights with their frequencies
  - Repeat $n-1$ times
    - Join two binary trees with smallest weights into one tree
    - Make the weight of the new tree the sum of the weights of the two trees
    - Makre edges leading to left and right subtrees with 0 or 1 respectivly

Example: Huffman Codes



Huffman tree generated from "This is an example of a Huffman tree"

3 Approximation: Greedy Traveling Salesmen

The Traveling Salesman Problem (TSP)

- Suppose we have a weighted connected graph G that we wish traverse s.t. we visit each vertex once and only once
- This is known as a Hamiltonian circuit (or cycle) of the graph
- The Traveling Salesman Problem asks what is the minimum cost Hamiltonian circuit in the graph G

Nearest Neighbor: the greedy approach to the TSP

- Pick a starting vertex v
- Each vertex keeps some state of whether or not that vertex has been visited in the tour
- Look at the neighbors of v and select the neighbor with the lowest cost that has not been visited

Nearest Neighbor: the greedy approach to the TSP

- In all probability, the tour discovered by this algorithm will not be optimal
- But just how far from optimal is it? How good of any approximation do we have?

Shortest edge: another greedy approach to the TSP

- Inspired by Kruskal's algorithm
- Sort the edges in increasing order of weight
- Start with the least cost edge, look at edges 1-by-1 and select an edge only if the edge, together with already selected edges,
  - does not cause a vertex have degree of three or more (# edges)
  - does not form a cycle, unless the number of selected edges equals the number of vertices in the graph

TSP is known to be difficult

- In the category of "NP-hard problems"
- Such problems require us to find approximation method
- In terms of optimization problems, have to decide of a local optimal solution is good enough

4 Key Points

Key Points

- What is a greedy algorithm?
- How we use such algorithms to solve optimization problems?
- The question of global vs. local optimum