CMPS 6610 Problem Set 04

In this assignment we'll look at the greedy and dynamic programming paradigms.

To make grading easier, please place all written solutions directly in `answers.md`, rather than scanning in handwritten work or editing this file.

All coding portions should go in `main.py` as usual.

Part 1: Fixed-Length vs. Variable-Length Codes

You saw the Huffman coding algorithm for data compresssion in our course materials. Let's implement the algorithm and look at its empirical performance on a dataset of 5 text files.

1a) Already implemented is a means to compute character frequencies in a text file with the function `get_frequencies` in `main.py`. Compute cost for a fixed length encoding for each text file.

1b) Complete the implementation of Huffman coding in `make_huffman_tree`. Note that we manipulate binary trees in the priority queue using the object `TreeNode`. Moreover, once the tree is constructed, we must compute the actual encodings by traversing the Huffman tree that has been constructed. To do this, complete the implementation of `get_code`, which is a typical recursive binary tree traversal. That is, given a tree node, we recursively visit the left and right subtrees, appending a `0` or `1` to the encoding in each direction as appropriate. If we visit a leaf of the tree (which represents a character in the alphabet) we store the collected encoding for that character in `code`.

1c) Now implement `huffman_cost` to compute the cost of a Huffman encoding for a character set with given frequencies.

1d) Test your implementation of Huffman coding on the 5 given text files, and fill out a table of the encoding cost of each file for fixed-length and Huffman. Fill out a final column which gives the ratio of Huffman coding cost to fixed-length coding cost. Do you see a consistent trend? If so, what is it?

enter answer in `answers.md`

1d) Suppose that we used Huffman coding on a document with alphabet Σ in which every character had the same frequency. What is the expected cost of a Huffman encoding for the document? Is it consistent across documents?

enter answer in `answers.md`

Part 2: Binary Heaps

2a) Give a method to construct a binary min-heap in $O(n)$ work. Hint: Given an array A of elements, consider the implicit representation as an almost-complete binary tree and show how to achieve the heap property for this tree with $O(n)$ work.

2b) What is the span of your approach?

Part 3: Making Change

After completing a tortuous semester of Algorithms, you decide to take a much needed vacation. You arrive in a city called Geometrica, and head to the bank to exchange N dollars for local currency. In Geometrica they have a currency that is 1-1 with U.S. Dollars, but they only have coins. Moreover the coins are in denominations of powers of 2 (e.g., k denominations of values 2^0, 2^1, $\ldots$, 2^k). You wonder why they have such strange denominations. You think about it a while, and because you had such a good Algorithms instructor, you realize that there is a very clever reason.

3a) Given a N dollars, state a greedy algorithm for producing as few coins as possible that sum to N.

enter answer in `answers.md`

3b) Prove that this algorithm is optimal by proving the greedy choice and optimal substructure properties.

enter answer in `answers.md`

3c) What is the work and span of your algorithm?

enter answer in `answers.md`

Part 4: Making Change Again

You get tired of Geometrica and travel to the nearby town of Fortuito. While Fortuito also has a 1-1 exchange rate to the US Dollar, it has an even stranger system of currency where any given bank has a completely arbitrary set of denominations (k denominations of values $D_0, D_2, \ldots, D_k$). There is no guarantee that you can even make change. So you wonder, given N dollars is it possible to even make change? If so, how can it be done with as few coins as possible?

4a) You realize the greedy algorithm you devised above doesn't work in Fortuito. Give a simple counterexample that shows that the greedy algorithm does not produce the fewest number of coins.

enter answer in `answers.md`

4b) Since you paid attention in Algorithms class, you realize that while this problem does not have the greedy choice property it does have an optimal substructure property. State and prove this property.

enter answer in `answers.md`

4c) Use this optimal substructure property to design a dynamic programming algorithm for this problem. If you used top-down or bottom-up memoization to avoid recomputing solutions to subproblems, what is the work and span of your approach?

enter answer in `answers.md`

Part 5: Weighted Task Selection

In class we gave a greedy algorithm for task scheduling. In that problem, all tasks had equal value and the goal was to simply maximize the total number of non-overlapping tasks. Suppose now that we consider tasks $A = \{a_0, \ldots, a_{n-1}\}$ where each task a_i has (s_i, f_i, v_i), with start and finish times as well as a value for completion. The goal now is to identify a set of tasks with maximum value.

5a) Does the optimal substructure property hold for weighted task selection? If so, prove it. If not, give a counterexample.

enter answer in `answers.md`

5b) Does the greedy choice property hold for this problem? If so, define a greedy criterion and prove that it satisfies the greedy choice property. If you cannot, find at least two counterexamples of greedy criteria that fail to achieve an optimal solution.

enter answer in `answers.md`

5c) Use the optimal substructure property of this problem to design a dynamic programming algorithm for this problem. Derive the work and span of your approach.

enter answer in `answers.md`