CS325: Coin change project

Stephen Redfield

Due Sunday, end of Week 6

Your report must be typeset *PDF document. Each team member's name must be listed as well as the Group Number and any resources used to finish the project.*

Coin Change Problem

For this project, you will investigate the **coin-change** problem:

> Given a cash register, how does one make change such that the fewest coins possible are returned to the customer? In this assignment, we explore two algorithmic solutions to this problem: a greedy algorithm and a dynamic programming algorithm.

Formally, an algorithm for this problem should take as input:

- A vector V where $V[i]$ is the value of the coin of the i^{th} denomination.
- A value C which is the amount of change we are asked to make.

The algorithm should return a vector O where $O[i]$ is the number of coins of value $V[i]$ to return. Note that it is required that your algorithm output O satisfying:

$$\sum_i V[i] \cdot O[i] = C$$

The objective is to minimize the number of coins returned or:

$$\sum_i O[i]$$

You will implement a greedy algorithm and a dynamic programming algorithm for this. The execution of the program should be as follows

- User runs program (either changegreedy or changedp) on the command-line, specifying a file in which the first line contains the array V, formatted as "[1,5,10,25]" without the quotes.
- Each subsequent line contains one decimal value for which we must make change.

Program output should be to a file named [input_filename]_change.txt where [input_filename].txt is the input filename, and should be formatted with one change result, e.g. "[1,0,1,1]", without the quotes, per line.

Greedy Algorithm

One approach to coin change problem is a greedy approach:

- Return the largest value coin possible.
- Subtract the value of this coin from the amount of change to be made.
- Repeat.

This implementation is called `changegreedy`.

Dynamic Programming

We can use a dynamic programming table T indexed by values of change $0, 1, 2, \ldots, C$ where $T[v]$ is the minimum number of coins needed to make change for v

$$T[v] = \min_{i:V[i] \leq v} \{T[v - V[i]] + 1\}$$

We initialize $T[0] = 0$. How do you store and return the number of each type of coin to return? (That is, how do you build $O[i]$?) This implementation is called `changedp`.

Example

Suppose $V = [1, 10, 25, 50]$ and $C = 40$. `changegreedy` should return $O = [5, 1, 1, 0]$ and `changedp` should return $O = [0, 4, 0, 0]$. If C is changed to 76, both algorithms should return $O = [1, 0, 1, 1]$. For $C = 40$, the dynamic programming solution outperforms the greedy solution: it finds a solution requiring fewer coins.

Your report

Your team's report should include answers to the following questions.

1. Describe, in words, how you fill in the dynamic programming table in `changedp`. Justify why is this a valid way to fill the table?
2. Give pseudocode for each algorithm, `changegreedy` and `changedp`.
3. Prove that the dynamic programming approach is correct by induction. That is, prove that $T[v] = \min_{i:V[i] \leq v}\{T[v - V[i]] + 1\}$, $T[0] = 0$ is the minimum number of coins possible to make change for value v.
4. Suppose $V = [1\ 10\ 25\ 50]$. For each integer value of C in and [2000 2300] determine the number of coins that `changegreedy` and `changedp` requires. Plot this number of coins (two lines, one for each `changegreedy` and `changedp`) as a function of C. How do the two approaches compare?
5. For the above situation, determine (experimentally) the running time required by both approaches, using the methods suggested for the first two projects. How do the running times of the two approaches compare? What is the *asymptotic* running time in terms of the number of denominations and the value C for each approach?
6. Suppose $V = [1\ 5\ 10\ 25\ 50]$. For each integer value of C in and [2000 2300] determine the number of coins that `changegreedy` and `changedp` requires. Plot this number of coins (two lines, one for each `changegreedy` and `changedp`) as a function of C. How do the two approaches compare?
7. Suppose you are living in a country where coins have values that are powers of p, i.e. $V = [p^0, p^1, p^2, p^3, \cdots]$. How do you think the dynamic programming and greedy approaches would compare? Explain.

8. Is there a set of coins for which the greedy algorithm does not simply find a solution requiring more coins than the dynamic programming algorithm, but all out fails? That is, is there a set of coins such that the greedy algorithm will get stuck and not be able to find a set of coins whose value adds up to the required amount C, but that the dynamic programming algorithm will not get stuck? If no, explain. If yes, give an example.

CODE: Upload a ZIP (actually .zip) containing (1) your Report PDF, (2) your README, and (3) your code to Blackboard and T.E.A.C.H. Only one student from each group should do this.