*Your report must be typed and submitted online. Each team member’s name must be listed as well as any resources used to finish the project.*

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

Given coins of denominations (value) 1 = v1 < v2< … < vn, we wish to make change for an amount A using as few coins as possible. Assume that vi’s and A are integers. All values of A will have a solution since v1= 1.

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

* An array V where V[i] is the value of the coin of the ith denomination.
* A value A which is the amount of change we are asked to make

The algorithm should return an array C where C[i] is the number of coins of value V[i] to return as change and m the minimum number of coins it took. You must return exact change so

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

# Implementation:

You may use any language you choose to implement your algorithms. You will implement three algorithms for this problem. The algorithms may be combined in one program or implemented as separate programs. The algorithms are to be based on these ideas:

**1. Implement a Brute Force or Divide and Conquer Algorithm:**

It is your choice of which algorithm to implement. This implementation will be called **changeslow** since either a brute force or recursive divide and conquer algorithm will be very slow. A brute force algorithm would try all possible combinations of coins that could add to A cents and keep track of the minimum number of coins used. A divide and conquer algorithm can be based on the following idea.

To make change for A cents, start by letting K = A

* If there is a K-cent coin, then that one coin is the minimum
* Otherwise, for each value i < K,
  1. Find the minimum number of coins needed to make i cents
  2. Find the minimum number of coins needed to make K - i cents
* Choose the i that minimizes the sum of 1) and 2)

Another variation of a recursive algorithm would let MC(K) rep­re­sents the min­i­mum num­ber of coins required to make change of amount K and then the options are:

* Select 1st coin (value = v1), Now Smaller prob­lem is min­i­mum num­ber of coins required to make change of amount( K-v1),so call  MC(K-v1)
* Select 2st coin (value = v2), Now Smaller prob­lem is min­i­mum num­ber of coins required to make change of amount( K-v2), so call MC(K-v2)
* Like­wise to up to n
* Select nth coin (value = vn), Now Smaller prob­lem is min­i­mum num­ber of coins required to make change of amount( K-vn),  so call MC(k-vn).

We need to find the min­i­mum num­ber of coins required to make change for **K** amount. So we will select the min­i­mum of all the smaller prob­lems and add 1 to it because we have select one coin. Now smaller prob­lems will be solved recursively.

**2. Greedy Algorithm:**

Another approach to coin change problem is the greedy approach. This implementation is called **changegreedy**. This is also “naive” since it may not be optimal.

*•* Use the largest value coin possible.

*•* Subtract the value of this coin from the amount of change to be made.

*•* Repeat.

**3. Dynamic Programming:**

One dynamic programming approach uses table *T* indexed by values of change 0*,* 1*,* 2*, . . . , A* where *T* [*v*] is the minimum number of coins needed to make change for v.

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 *C*[*i*]?) This implementation is called **changedp**. Note: there are other versions of the DP algorithm you may use one but need to explain in your report.

The execution of the program(s) should be as follows:

* User runs the programs on the command-line, specifying a file ([input filename].txt) in which the first line contains the array V, formatted as 1 5 10 25, denominations in increasing order.
* The next line contains one integer value for which we must make change.

Program output should be written to a file named [input filename]change.txt where [input filename].txt is the input filename, and should be formatted with the name of the algorithm used, the denominations, the change result and the minimum number of coins m, per line. For example

Amount.txt:

1 2 5

10

1 3 7 26

22

Amountchange.txt:

Algorithm changeslow:

1 2 5

0 0 2

2

1 3 7 26

1 0 3 0

4

**Testing for correctness.** Above all else your algorithm should be correct. You can test your algorithms on the following:

* + - * Suppose V= [1,2,4,8] and A= 15. All algorithms should return C=[1,1,1,1] and m = 4.
      * Suppose V= [1,3,7,12] and A= 29. The **changegreedy** should return C= [2,1,0,2] with m = 5 and **changedp and slowchange** should return C= [0,1,2,1] with m = 4. The minimum number of coins is four. The greedy algorithm is suboptimal.
      * If A is changed above to 31, all algorithms should return C = [0, 0, 1, 2], with m = 3.

# Project Report

Your team’s report should include the following:

1. Give the pseudocode and theoretical asymptotic running time for each algorithm.

2. Describe, in words, how you fill in the dynamic programming table in changedp**.** Justify why is this a valid way to fill the table?

3. Suppose V1= [1, 2, 6, 12, 24, 48, 60], V2 = [1, 5, 10, 25, 50] and V3 = [1, 6, 13, 37, 150], for each integer value of *A* in [1, 2, 3, …, 50] determine the number of coins that changeslow. changegreedy and changedp requires for each denomination set. For each integer value of *A* in [2000, 2001, 2002, …, 2200] determine the number of coins that changegreedy and changedp requires for each denomination set (you can attempt to run changeslow but it will probably be too slow). Plot the **number of coins as a function of *A***for each algorithm. How do the approaches compare?

4. For each of the algorithms collect experimental running time data. Plot the **running time as a function of A** and fit curves to the data. Compare the experimental running times to the theoretical running times.

5. Create a single log-log plot that contains the running time data from all three algorithms. Fit trend lines to each data set. Compare the running times of the different algorithms.

6. Suppose you are living in a country where coins have values V = [1, 3, 9, 27]. How do you think the dynamic programming and greedy approaches would compare? Explain.

7. Give at least three examples of denominations sets V for which the greedy method is optimal. Why does the greedy method produce optimal values in these cases?

# What to Submit

Your elected submitter must upload

* To TEACH: a ZIP file containing (1) Project Report PDF, (2) README, (3) CODE
* To Canvas: (1) Project Report PDF