Bryon Burleigh

Jennifer Erland

Jilian LaFerte

CS 325 Project 2: Coin Change

1. The array is filled by starting with the base case of 0, then the coins needed to make that change is none. Then fill the array as the change needed is increased by appending the coin value to the current coins needed according to the following scheme. Ignore any coins that are larger than the current change value. Then check that [current minimum minus the current coin value + 1] is greater than the current minimum. If so, the current minimum change set is now the change set of [current minimum minus the current coin value] plus the current coin.

2.

**Pseudo code for changeslow:**

**Pseudo code for changedp:**

for i from 1 to change needed

for coin in coin list

if coin > i, skip to next coin

else

if change list or (number of coins needed for (i-coin) +1) > number needed

if change list(i-coin) is not null

change list(i) = append coin to change list(i-coin)

**Pseudo code for changegreedy:**

for i = 0 to the number of denominations

append 0 to A (the list of coins needed)

set i to the length of V -1 (so that it is set to the largest denomination)

while the change value is greater than 0

if V[i] is less than the current change value

subtract that denomination (V[i]) from the current change calue

increase the minimum number of coins by one

add one to A[i]

else

subtract one from i

3.

Let T(v) be the minimum number of coins needed to make change for v. Let V[n] be the nth coin in the coin list.

Base step:

When v = 0, T(0) =0.

Inductive step:

Assume T(k) is correct for some positive integer k. Show that T(k+1) is also correct.

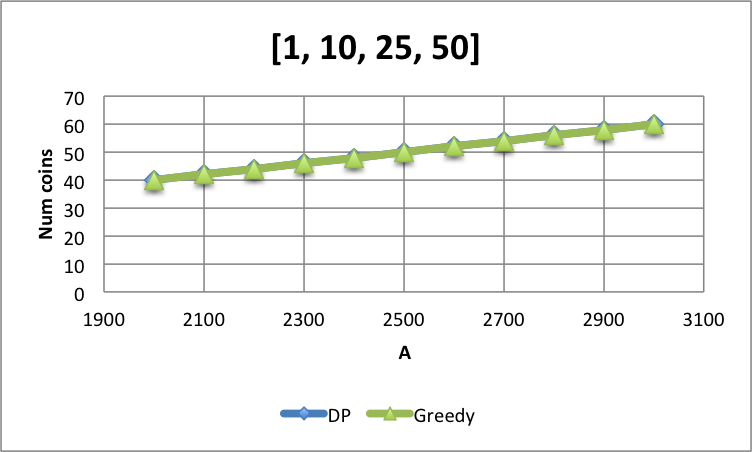
Case 1: T(k+1) = min {T(k-V[i]) + 1} +1 occurs when the addition of exactly 1 coin is needed to make change for k+1.

Case 2: T(k+1) < min {T(k-V[i] + 1} occurs when the addition of a lower value coin in the list would enable a number of lower value coins in change to be exchanged for a higher value coin, thus lowering the minimum number of coins needed.

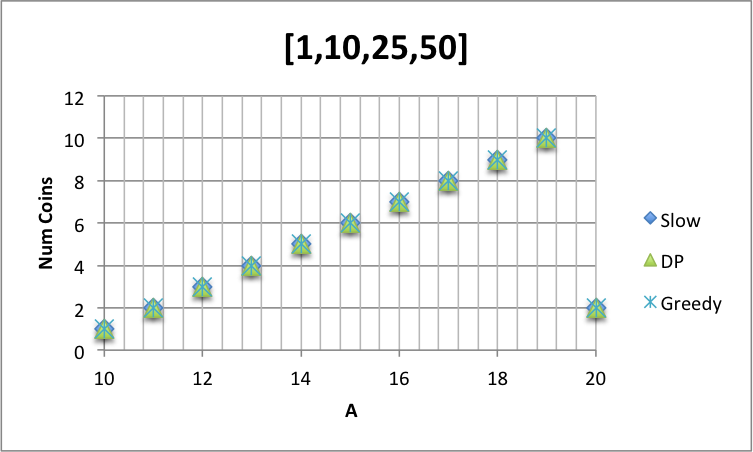
Note, the algorithm does not permit for T(k+1) > min {T(k-V[i] + 1}.

Hence, T(k+1) is correct in all possible cases. Thus, the algorithm is correct.

4.

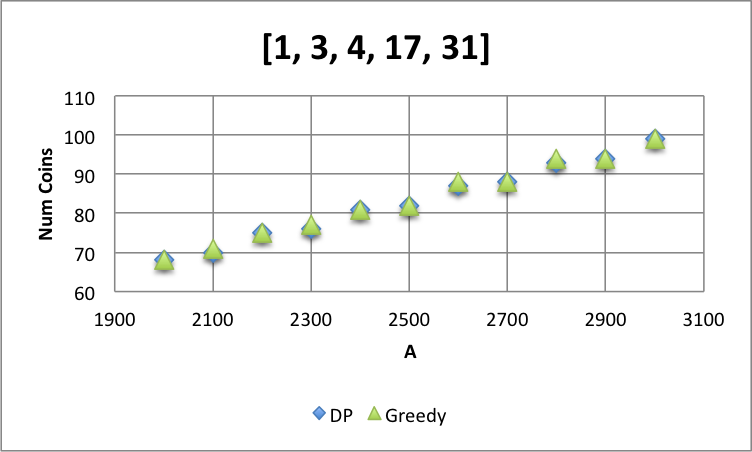


This is the graph of the values given in the assignment. Here, we left out the slow algorithm because it would take an unrealistic amount of time to run. You can’t see the DP line here because the number of coins needed is exactly the same for both DP and Greedy algorithms.

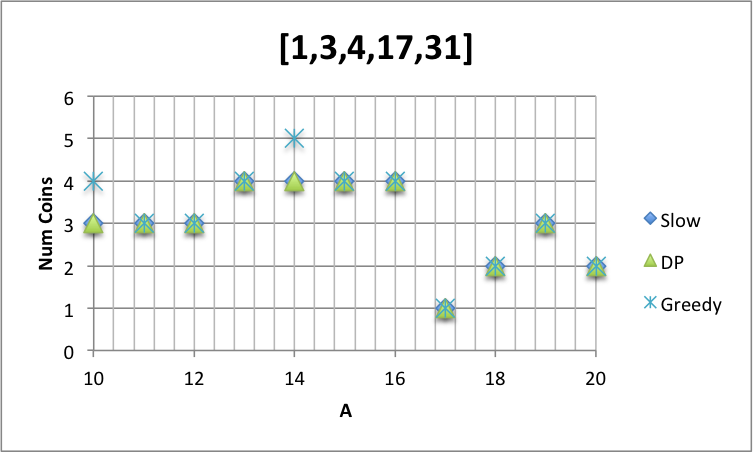


This is the graph from the tests we ran with all three algorithms. We used the values 10 through 20 so that we could run all three without waiting a lifetime. Again, all algorithms used the same number of coins.

5.



This is the graph for the values given in the assignment. Again, we did not include the slow algorithm because it takes too long to run. The two algorithms only deviate in a dew places, but when they do, the greedy algorithm uses one more coin than the DP algorithm.



We again, wanted to compare the slow algorithm with the others, so we used the values 10 through 20 because it was more manageable time-wise. Like with the larger values of A, the greedy is the only one that is different – but only sometimes. It is easier to see in this graph that when it does deviate, it uses one more coin than the other algorithms.

6.

The running times for each algorithm varies greatly. The changeslow algorithm is FAR slower than the other two. The changedp and changegreedy algorithms are closer to each other in terms of time to run, but changegreedy is still notably faster than changedp.

7.