CS 4720/5720 Design and Analysis of Algorithms
Homework #4
Daniel Frey

Answers to homework problems:

1. ***Edit Distance Problem***

   (a) **Algorithm:**

   Loop through each character in the first string while looping through each character in the second string. If either of the strings are empty, then set the value to the opposite loop counter to indicate the number of operations up to that point. If the previous characters were the same, then use previously generated result. If previous characters were different, then determine if change, insert, or delete.

   Pseudocode:

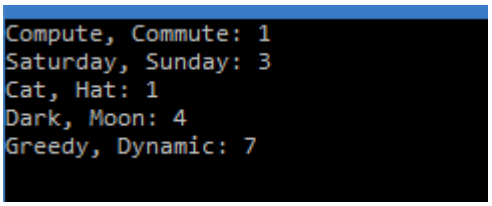
```
EditDistanceDP(string1, string2)
    m = size string1, n = size string2
    results[m+1, n+1]
    for i = 0 to m
        for j = 0 to n
            if i == 0
                results[i, j] = j
            else if j == 0
                results[i, j] = i
            else if string1[i-1] == string2[j-1]
                results[i, j] = results[i-1, j-1]
            else
                results[i, j] = 1 + min(results[i-1, j-1], results[i, j-1], results[i-1, j])
    return results[m, n]
```

   (b) **Time Complexity:**

   Worst-case: (Two for loops)
   $\sum_{i=0}^{m} \sum_{j=0}^{n} 1 = (m+1)(n+1) \in \Theta(mn)$

   Best-case = Worst-case $\in \Theta(mn)$: must iterate through entirety of each for loop.

   (c) ***Code***



2. ***Coin Change Problem***

   (a) ***DP Code***

   (b) *Greedy Coin Change*
   (Assumption: Denominations D[1...m] in ascending order.)

(i) **Algorithm:**

Loop through each coin denomination from largest to smallest. Figure out how many of current largest coin can be used for current value n, and subtract that from the current value n.

Pseudocode:

```
CoinChangeGreedy(D[1...m], n)
    numCoins = 0
    for i = m to 1
        if n/D[i] ≥ 1
            n = n - [(n/D[i]) * D[i]]
            numCoins = numCoins + (n/D[i])
    return numCoins
```

(ii) **Time complexity:**

Worst-case: (For loop dependent on m.)
$\sum_{m}^{1} 1 \in \Theta(m)$

Best-case: = Worst-case $\in \Theta(m)$: must iterate through entirety of for loop.

(iii) ***Code***

(c) **Comparison:**

(i) **Non-optimal Greedy Denominations:**

Minimum Number Coins			
Denomination	n	DP	Greedy
{1, 5, 6}	10	2	5
{1, 10, 15}	20	2	6
{1, 30, 50}	60	2	11

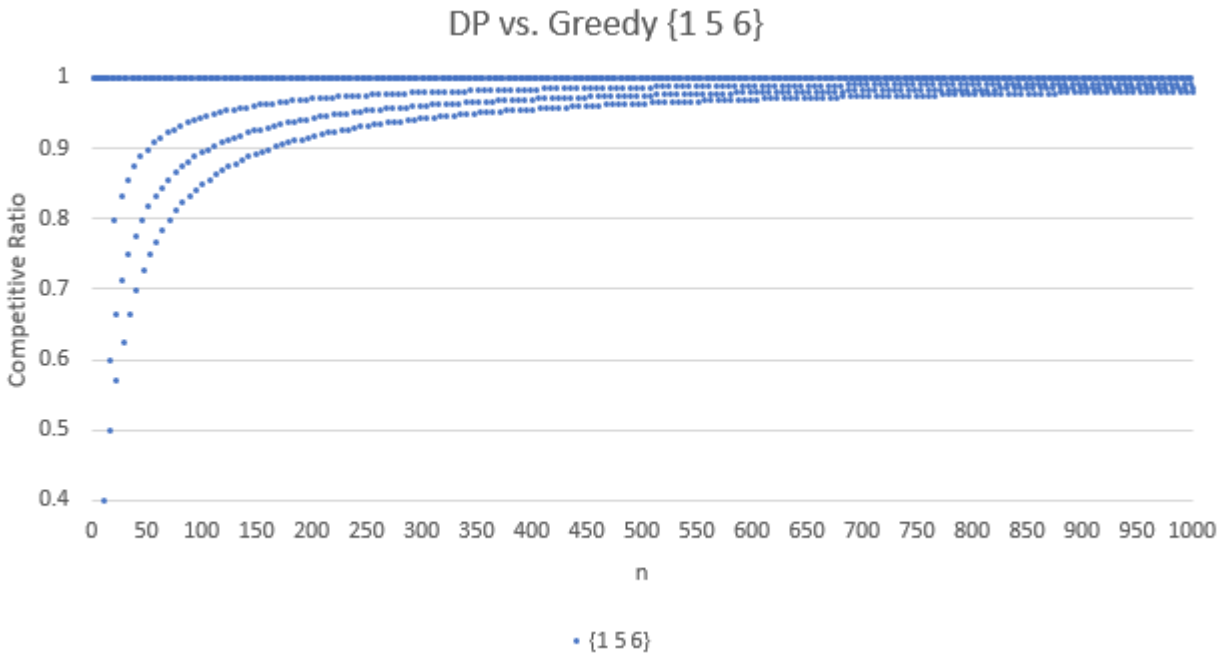
(ii) **Sample Comparisons:**

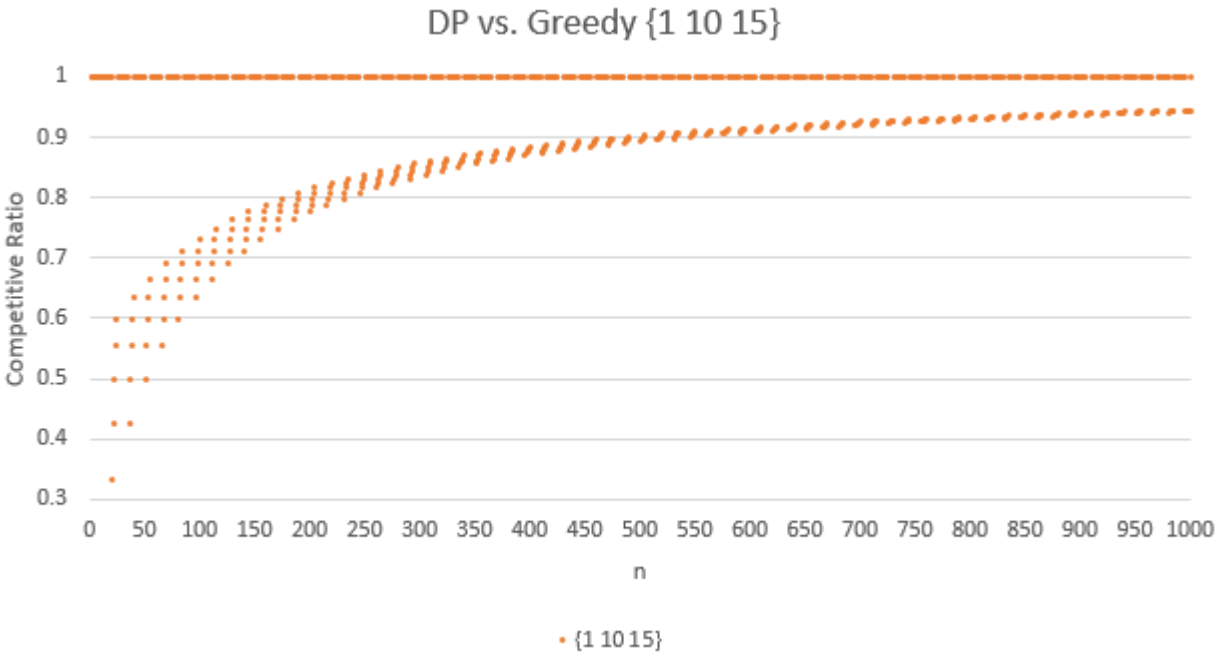
Denomination {1, 5, 6}		
n	DP	Greedy
250	42	45
500	84	85
750	125	125
1000	167	170

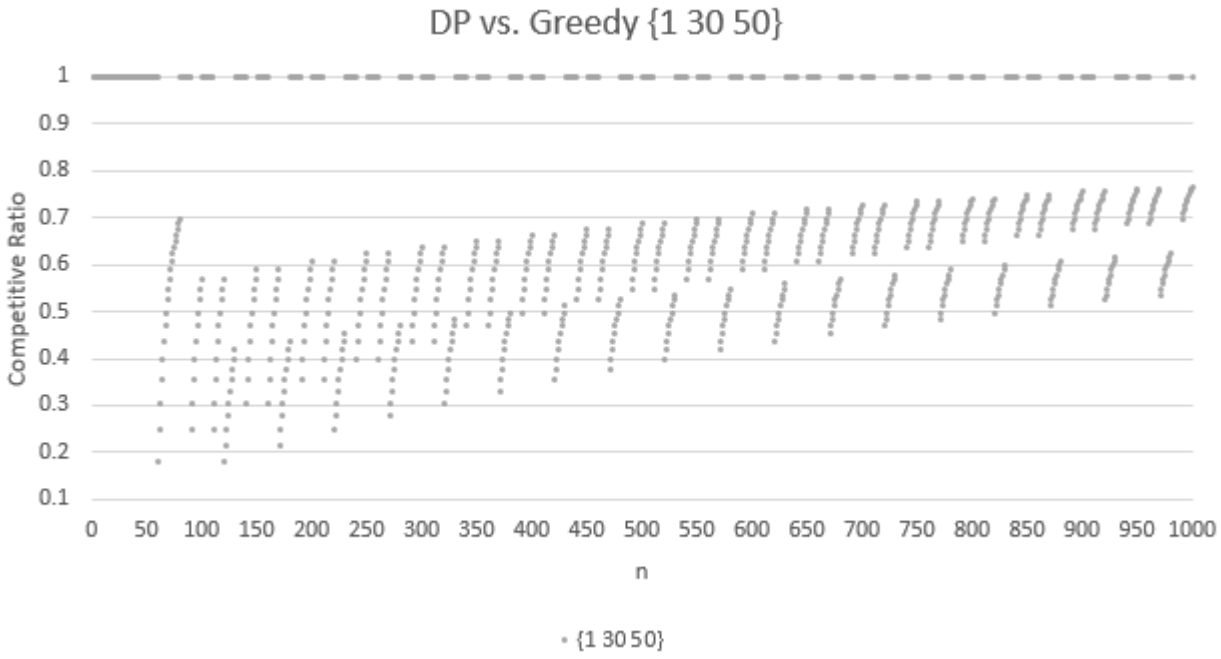
Denomination {1, 10, 15}		
n	DP	Greedy
250	17	17
500	34	38
750	50	50
1000	67	67

Denomination {1, 30, 50}		
n	DP	Greedy
250	5	5
500	10	10
750	15	15
1000	20	20

(iii) **Plot:**


DP vs. Greedy {1 5 6}
Competitive Ratio
1
0.9
0.8
0.7
0.6
0.5
0.4
0 50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950 1000
n
{1 5 6}


DP vs. Greedy {1 10 15}
Competitive Ratio
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0 50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950 1000
n
{1 10 15}



(d) (i) **Optimal Greedy Proof for US Coin Denominations:** {*1, 5, 10, 25, 50*}

For $1 \leq k \leq m$ prove that $D_k + D_{k-1} - 1$ yields the fewest coins for $\{1...D_k\} and \{1..., D_{k-2}, D_{k-1}\}$. If $\{1..., D_{k-2}, D_{k-1}\}$ yields fewer number of coins, then greedy fails.

Using {1, 5, 10, 25, 50}:
Highest value = 50 + 25 - 1 = 74 →7 coins
Removing highest denomination: {1, 5, 10, 25}
74 →8 coins
Highest value = 25 + 10 - 1 = 34 →6 coins
Removing highest denomination: {1, 5, 10}
34 →7 coins
Highest value = 10 + 5 - 1 = 14 →5 coins
Remove highest denomination: {1, 5}
14 →6 coins
Highest value = 5 + 1 - 1 = 5 →1
Remove highest denomination: {1}
5 →6 coins

∴ Greedy minimizes number of coins for {1, 5, 10, 25, 50} since greedy picks the fewest number of coins for the denominations.

(ii) **Greedy 2x, 10x vs. DP**

Minimum Number Coins				
Denomination	n	DP	Greedy	Greedy/DP
{1, 10, 25}	30	3	6	$\frac{6}{3} = 2$
{1, 43, 100}	129	3	30	$\frac{30}{3} = 10$