Assignment 2.

Name: Guangfeng Lin
Student #: 301312131

Question 1:

a) LCS (min1[1...m], min2[1...n]){
    C = array(0...m)(0...n); Build C table as a m x n matrix   O(1)
    for i = 0 to m {   O(m) iterations
        C[i,0] = 0;   O(1) per iteration
    }
    for j = 0 to n {   O(n) iterations.
        C[0,j] = 0;   O(1) per iteration
    }
    for i = 0 to m {   } O(mn) iterations.
        for j = 0 to n {
            if min1[i] == min2[j]
                C[i,j] = (C[i-1][j-1]) + 1;
            else
                C[i,j] = max(C[i, j-1], C[i-1, j]);   O(1) work per iteration.
        }
    }
    return C[m,n];   O(1)
}

O(m) + O(n) + O(mn) + O(1)

= O(mn)

Scanned with CamScanner

```
b) LCS (min[1...m]){
       C = array(0... m);
       for i = 0 to m{                                O(m) iterations
           C[i,0] = 0;                                    O(1) per iteration
       }
       target-array;                        give a target array.
       for i=0 to size of(target-array){.        O(size of(target-array)) iterations.
           if min[i] = target-array[i];
               C[i] = C[i-1] + 1;                  O(1)
           else
               C[i] = max(C[i, i-1]);              O(1)
       }
       return C[size of (target-array)];           O(1)
   }
```

$O(m) + O(\text{size of (target-array)}) + O(1)$

$= O(\text{size of target-array}).$

Question 2:

a) C_i = cost of ith operation

$C_i = i$ if i is an exact power of 2

$C_i = 1$ otherwise.

$\hat{C_i}$ = Amortized cost of ith operation and charge every operation 3.

operation	actual cost	amortized cost.
1	1	3
2	2	3
3	1	3
4	4	3
⋮	⋮	⋮

$\sum_{i=1}^{n} \hat{C_i} = 3n$

$\sum_{i=1}^{n} C_i \leq \sum_{i=1}^{n} \hat{C_i}$, since $\sum_{i=1}^{n} C_i \leq n + \sum_{j=0}^{\lfloor \log n \rfloor} 2^j \leq n+2n \leq 3n$

Therefore, the Amortized cost operation is less than 3.

b)

$\hat{C}_i = C_i + \Phi(D_i) - \Phi(D_{i-1})$

$\Phi(D_0) = 0.$

$\Phi(D_i) \geq 0$

$\Phi(D_i) = 2i - 2^{\lfloor \log_2 i \rfloor} + 1$ for $i > 0$.

For i, which is exact power of 2; potential difference is

$$\Phi(D_i) - \Phi(D_{i-1}) = 2i - 2i + 1 - (2(i-1) - i + 1)$$
$$= 2i - 2i + 1 - (2i - 2 - i + 1)$$
$$= 2i - 2i + 1 - 2i + 2 + i - 1$$
$$= 2 - i.$$

For i, which is not exact power of 2, potential different is

$$\Phi(D_i) - \Phi(D_{i-1}) = 2i - i + 1 - (2(i-1) - i + 1)$$
$$= 2i - i + 1 - 2i + 2 + i - 1$$
$$= 2.$$

Thus, $\hat{C}_i = i + 2 - i = 2$, if $i = 1^{st}$ operation

$\hat{C}_i = 1 + 2 = 3$, if $i = i^{th}$ operation

Therefore, the Amortized cost operation is less than or equal 3.

Question 3 :

a) With the greedy algorithm, people can have the highest denomination coin until the amount of remaining change change to 0. For example if the value you want is 103 then, the algorithm will return 1 one hundred dollar denomination and 3 one dollar denomination. This is the fastest way to get the particular amounts that user needs.

b) let D_i indicates the number of coins of denomination C_i. The optimal solution is $(D_0, D_1, \dots, D_k)$. First, we need to show that $D_i < C$ for every $i < k$, if $D_i \geq C$ then we can decrease D_i by C and increase D_{i+1} by 1. With this solution, the output value will be the same and will have $C-1$ fewer amount number of coins. Since everytime is picking up the largest denomination if possible so it will output the fewest number of denominations.

c) let's consider the set $\{1, 3, 4\}$, and the asking for a change of 6 cents. The greedy algorithm will pick two 1s and one 4. which is $2 \times 1 + 1 \times 4 = 6$ cents. And it have 3 coins for the solution. However, the best solution for the change of 6 cents is picking up two 3s which is $2 \times 3 = 6$ cents, and it is end up picking up 2 coins.

d) Greedy 2.0(int amount, coins[...]) {

```
Greedy 2.0( int amount, coins[...]) {
    if amount == 0
        return 0;

    for i= size of coins to i>0 , i-- {
        coin = coins [i - 1];
        if amount >= coin {
            result = 1+ Greedy 2.0 (amount - coin, coins [...]);
            return result;
        }
    }
    return 0;
}
```

Scanned with CamScanner