Problem 1 (Problem 16.1-3)

Solution: 1) Let $(s_1,f_1) = (3,5)$, $(s_2,f_2) = (1,4)$, $(s_3,f_3) = (4,7)$. The greedy solution is {1}, but the optimal solution is {2,3}

2) Let $(s_1,f_1) = (3,5)$, $(s_2,f_2) = (0,2)$, $(s_3,f_3) = (6,8)$, $(s_4,f_4) = (2,4)$, $(s_5,f_5) = (4,6)$, $(s_6,f_6) = (1,3)$, $(s_7,f_7) = (1,3)$, $(s_8,f_8) = (5,7)$, $(s_9,f_9) = (5,7)$. The greedy solution is {1,2,3}, but the optimal solution is {2,3,4,5}

3) Let $(s_1,f_1) = (1,5)$, $(s_2,f_2) = (2,3)$, $(s_3,f_3) = (4,5)$. The greedy solution is {1}, but the optimal solution is {2,3}

Problem 2 (Problem 16.1)

Solution

a) Given n cents, let $n_q = [n/25]$, $r_q=n-25n_q$, $n_d = [r_d/10]$, $r_d=r_q-10n_d$, $n_k = [r_d/5]$, $r_k = r_d – 5n_k$, $n_p = r_k$. Thus, we can make change for n dollars by obtaining n_q quarters, n_d dimes, n_k nickels and n_p pennies. This greedy algorithm requires O(1) time.

Next, we will prove the correctness.

We prove it by induction, First the Greedy algorithm produces optimal solution for arbitrary n if there are only nickels and pennies, and let's demote this greedy algorithm by A2. Assume that the optimal solution is x_k nickels and x_p pennies. If $x_p>=5$. Then it's not optimal because $x_k'=x_k+[x_p/5]$, , gives fewer number of coins, contradiction.

Next, we prove that the greedy algorithm also works if there are only dimes, nickels and pennies, and we denote this greedy algorithm by A3. Assume the optimal solution is x_d, x_k, x_p. If $x_p<5$, we can employ A2 to $5x_k+x_p$ to get a better solution. If $5x_k+x_p<10$, then (x_d, x_k, x_p) is also the greedy solution to A3 and is optimal. If $5x_k+x_p>=10$, which implies $5x_k<=10$ since $x_p<5$. Then we can get a better solution $(x_d+[x_k/2], x_k-2[x_k/2], x_p)$. Therefore, A3 produces optimal solution.

Finally, we prove that the greedy algorithm is correct is there are quarters, dimes, nickels, and pennies. And we denote this algorithm by A4. Otherwise, assume the optimal solution is x_q,x_d,x_k,x_p. We know that $x_k<2$ and $x_p <5$; otherwise, we can use algorithm A3 to get a better solution on the input $10x_d+5x_k+x_p$. Let $n_q=[n/25]$. If $x_q=n_q$, then this optimal solution is also the solution to algorithm A4. If $x_q<n_q$, then by A3, each quarter will lead to 1 more coin, which means that (x_q,x_d,x_k,x_p) is not optimal.

Therefore, the greedy algorithm gives the optimal solution

b) given an optimal solution $(x_0,x_1,\ldots x_k)$, where x_i indicates the number of coins of denomination c_i. First, $x_i<c$ for every i=1, 2,…,k-1. Suppose we have some $x_i>=c$, then we could decrease x_i by c and increase x_i+1 by 1. This connection of coins has the same value and has c-1 fewer coins, so the original solution must be non-optimal. This configuration of coins is exactly the same as you would get if you kept greedily picking the largest coin possible. Let $S_i = \sum_{i=1}^{i} c^j(c-1)$. Then $S_i<c_i+1$ for i = 0,1,2…k. This $(x_0,x_1,\ldots x_k)$ is the only solution that satisfies the property $x_i<c$ (i=1,2,…,k-1). Therefore, the greedy algorithm always yields an optimal solution.

c) Let the coin denominations be {1,3,4}, and the value to make change for be 6. The greedy algorithm would result in the collection of coins {1,1,4}, but the optimal solution is {3,3}

d) Let n_coins[i] be the fewest number of coins to make change for I cents and S be the collection of k distinct coin denominations. The n_coins[0]=0, the recursive part is: $n\_coins[i] = min_{i\geq c, c\in S} n\_coins[i-c]+1$

```
Input: S,n
Output: n_coins[n] and coins set {}
Let n_coins[0] =0
For i from 1 to n do
    Let n_coins[i] = +∞
    For c in S do
        If c<= i and n_coins[i] > n_coins[i-c]+1 then
            n_coins[i] = 1 + n_coins[i-c]
        End if
    End for
End for
iter = n
Let coins={}
While iter > 0 do
    For c in S do
        If c <= iter and n_coins[i] == n_coins[i-c]+1 then
            Add c to coins
            iter = iter -c
        End if
    End for
End while
```

Time complexity is $O(nk)$, where $k = |S|$