CS 102: SOLUTIONS TO GREEDY ALGORITHMS
FALL 2001: ASSIGNMENT 6

Part A:

(a). All the three greedy strategies proposed for 0-1 knapsack problem fail. The first two strategies proposed for continuous knapscak problem fail. Counterexamples are simple to construct and have been discussed in the class. The third strategy gives the correct solution.
(b). Counterexample is simple to construct and has been discussed in the class.
(c). Counterexamples are simple to construct and have been discussed in the class.
(d). The first three greedy strategies proposed for activity scheduling problem fail. Counterexamples are simple to construct and have been discussed in the class. The final strategy gives the correct solution.
(e). Solution to this problem is described in the textbook.

Part B:

(a). Consider the problem of making change for n cents using the least number of coins.

(a) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

A greedy algorithm for the problem is as follows:

```
Make-Change(x,c,num_coins)
  for i := 1 to num_coins do
    num[i] := 0
  for i := num_coins downto 1 do
    while x >= c[i] do
      num[i] := num[i] + 1
      x := x - c[i]
return num
```

The array `c` constains the denominations of the coins, and `num` returns the number of each coin taken in order to sum to `x`. Note that this algorithm isn't specific to any coin denominations. Now we need to prove that this algorithm returns an optimal result when the denominations 1, 5, 10, and 25 are used. Let G_1 be the number of pennies taken by the greedy algorithm, G_2 the number of nickels, G_3 the number of dimes, and G_4 the number of quarters. And in a similar way, let O_1 through O_4 be the number of coins taken by an optimal solution. Let n be the value to make change from. Now we'll prove the greedy solution is optimal coin by coin.

First we do pennies. We know that $n = 1G_1 + 5G_2 + 10G_3 + 25G_4 = 1O_1 + 5O_2 + 10O_3 + 25O_4$ which means that $n - G_1 \mod 5 = 0$ and $n - O_1 \mod 5 = 0$ and so $O_1 - G_1 \mod 5 = 0$. From the algorithm above we know that $0 \leq G_1 < 5$, and for an optional solution $O_1 < 5$

otherwise the solution could have taken a nickel to use less coins. That means $O_1 - G_1 = 0$ or $O_1 = G_1$ so the two solutions use the same number of pennies.
Next we take care of the nickels. As above, we note that $0 \leq G_2 < 2$ and $0 \leq O_2 < 2$ otherwise the respective greedy and optimal solutions would have taken a dime instead. Suppose that $O_2 \neq G_2$ or that $O_2 - G_2 \neq 0$. Then $O_2 - G_2 \geq -1$. Also we know $G_4 \geq O_4$ (the greedy algorithm takes the maximum number of quarters possible) so we can set $k = G_4 - O_4$ and know that $k \geq 0$. Then we have

$$\begin{aligned}
10(O_3 - G_3) + 25(O_4 - G_4) &= -5(O_2 - G_2) \\
10(O_3 - G_3) + 25(-k) &= -5(O_2 - G_2) \\
O_3 - G_3 &= \frac{25k - 5(O_2 - G_2)}{10}
\end{aligned}$$

But $O_3 - G_3$ must be an integer which means k can't be zero. So $k \geq 1$. That means

$$\begin{aligned}
\sum_{i=1}^{4}(O_i - G_i) &= 0 + (O_2 - G_2) + \frac{25k - 5(O_2 - G_2)}{10} - k \\
&= (O_2 - G_2) + 2.5k - 0.5(O_2 - G_2) - k \\
&= 0.5(O_2 - G_2) + 1.5k
\end{aligned}$$

and since $O_2 - G_2 \geq -1$ and $k \geq 1$

$$\begin{aligned}
&> -0.5 + 1.5 \\
&> 0
\end{aligned}$$

which is a contradiction since the optimal solution has to use the same or fewer coins that the greedy solution. Hence $G_2 = O_2$.
Finally we take on the dimes and the quarters. We now have that $10O_3 + 25O_3 = 10G_3 + 25G_3$ which means $O_3 - G_3 = 2.5(G_4 - O_4)$. But we still know that $G_4 \geq O_4$ so if $k = G_4 - O_4$ then $O_3 - G_3 = 2.5k$ and

$$\sum_{i=1}^{4} O_i - G_i = 0 + 0 + 2.5k - k = 1.5k \geq 0.$$

Since O is an optimal solution, the only way for this to happen is if $k = 0$. That means that $G_4 = O_4$ and $G_3 = O_3$.
Since the greedy solution uses the same number of coins as an optimal solution, it too must be optimal.