Review

Friday, October 21, 2022 5:38 PM

HW3

7.1. This exercise is about the *money change* problem stated in Exercise 6.29. Consider a currency system that has the following coins and their values: dollar (100 cents), quarter (25 cents), dime (10 cents), nickel (5 cents), and 1-cent coins. (A unit-value coin is always required). Suppose we want to give a change of value y cents in such a way that the total number of coins n is minimized. Give a greedy algorithm to solve this problem.

Algorith : MChangeGreedy (y)

input : y

output: C100, C25, C10, C5, C1

1. $x = y$
2. $C100 = \lfloor x / 100 \rfloor$
3. $x = x - 100 * C100$
4. $C25 = \lfloor x / 25 \rfloor$
5. $x = x - 25 * C25$
6. $C10 = \lfloor x / 10 \rfloor$
7. $x = x - 10 * C10$
8. $C5 = \lfloor x / 5 \rfloor$
9. $x = x - 5 * C5$
10. $C1 = x$

Return C1, C5, C10, C25, C100

Algorith MChange2

input y , V[1 ... n] // V = [1, 5, 10, 25, 100]

output C[1 ... n]

```
x = y
for i = n down to 1
    C[i] = ⌊x / V[i]⌋
        x = x − C[i] * V[i]
end for
Retur C[1,..n]
```

7.2. Give a counterexample to show that the greedy algorithm obtained in Exercise 7.1 does not always work if we instead use coins of values 1 cent, 5 cents, 7 cents, and 11 cents. Note that in this case dynamic programming can be used to find the minimum number of coins. (See Exercises 6.29 and 6.30).

for $y = 14$

Greedy Alg: $1 \times (11) + 3 \times (1) = 14$, with 4 coins

while : $2 \times (7) = 14$, with 2 coins

$\therefore$ the greedy algorithm is not optimal.

Split.

j

6	7	3	4

i

6	3	7	4

6	3	4	7

4	3	6	7

R

Algorithm 5.5 SPLIT

Input: An array of elements $A[low..high]$.

Output: (1)A with its elements rearranged, if necessary, as described above.
(2) w, the new position of the splitting element $A[low]$.

```
i ← low
x ← A[low]
for j ← low + 1 to high
    if A[j] ≤ x then
        i ← i + 1
        if i ≠ j then interchange A[i] and A[j]
    end if
end for
interchange A[low] and A[i]
w ← i
return A and w
```

j i

8	5	6	7
7	5	6	8

6	3	4	7

Quiz 2

1.

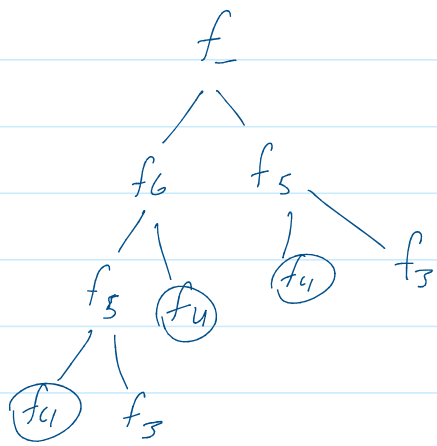
(a) $(5 \times 4) \cdot (4 \times 6)$

$\rightarrow 5 \cdot \quad . \quad = 2$ mult-ops.

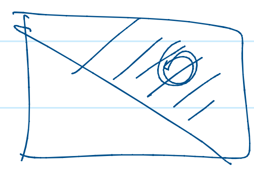
(b) Time complexity:

$\Rightarrow \theta(n^3)$

.



3 times



3.

i					j
3	4	7	5	6	2
3	2	7	5	6	4
2	3	7	5	6	4