Dayou Du
N13825075
dd2645@nyu.edu

Homework 5 - SA
Fundamental Algorithms

2017-10-19

Ex 2.21

Algorithm 1 Ex 2.21

Input: S, a set of pair (l, c) as described
Output: print all the strings that can be constructed from S

```
procedure MississippiDriver(S; ;)
    k ← S.size()
    Use S to create an array LetterCount(1..k) of pair(l, c).
        ▷ LetterCount[i].l stores the letter, and LetterCount[i].c stores how many this kind of letter we have
    Create an empty array Answer(1..n)
    Mississippi(k, n, LetterCount, Answer)
end procedure
procedure Mississippi(k, n; ; LetterCount, Answer)
    if n == 0 then
        Print(Answer)
        return
    end if
    for i ∈ [1..k] do
        Answer[n] ← LetterCount[i].l
        LetterCount[i].c ← LetterCount[i].c − 1
        if LetterCount[i].c = 0 then
            Swap(LetterCount[i], LetterCount[k])
            Mississippi(k − 1, n − 1, LetterCount, Answer)
            Swap(LetterCount[i], LetterCount[k])
        else
            Mississippi(k, n − 1, LtterCount, Answer)
        end if
        LetterCount[i].c ← LetterCount[i].c + 1
    end for
end procedure
```

At beginning we call $MississippiDriver(S)$
I got this question CORRECT.

Ex 3.29

a)

$A(1) = 1$
$A(n) = n + \frac{10}{9}A(\frac{9n}{10}), n > 1$
So the solution is $A(n) = n + n + n + ... + n$ where there are $1 + log_{\frac{10}{9}n}$ terms in the sum.
Thus $A(n) = \Theta(nlogn)$
I got this question CORRECT.

b)

$B(1) = 1$
$B(n) = n + B(0.7n) + B(0.3n), n > 1$
So the solution is $A(n) = n + n + n + ... + n$ where there are no more than $1 + log_{\frac{10}{7}n}$ terms in the sum.
Thus $A(n) = \Theta(nlogn)$
I got this question CORRECT.

c)

$C(1) = 1$
$C(n) = n + C(0.9n), n > 1$
So the solution is $C(n) = 0.9^0 n + 0.9^1 n + ... + 0.9^{log_{\frac{10}{9}} n} n$
Thus $C(n) = \Theta(n)$
I got this question CORRECT.

d)

$D(1) = 1$
$D(n) = n + D(0.7n) + D(0.2n), n > 1$
So the solution of $D(n)$ is no more than $0.9^0 n + 0.9^1 n + ... + 0.9^{log_{\frac{10}{7}} n} n$ and no less than $0.9^0 n + 0.9^1 n + ... + 0.9^{log_5 n}$
Thus $D(n) = \Theta(n)$
I got this question CORRECT.

e)

$E(1) = 1$
$E(n) = n^2 + D(0.7n) + D(0.3n), n > 1$
So the solution of $E(n)$ is no more than $(0.7^2 + 0.3^2)^0 n^2 + (0.7^2 + 0.3^2)^1 n^2 + ... + (0.7^2 + 0.3^2)^{log_{\frac{10}{7}} n} n^2$ and no less than $(0.7^2 + 0.3^2)^0 n^2 + (0.7^2 + 0.3^2)^1 n^2 + ... + (0.7^2 + 0.3^2)^{log_{\frac{10}{3}} n} n^2$
Thus $E(n) = \Theta(n^2)$
I got this question CORRECT.

f)

$F(1) = 1$
$F(n) = n^2 + F(\frac{3n}{5}) + F(\frac{4n}{5}), n > 1$
So the solution of $F(n)$ is no more than $(0.6^2 + 0.8^2)^0 n^2 + (0.6^2 + 0.8^2)^1 n^2 + ... + (0.6^2 + 0.8^2)^{log_{\frac{5}{4}} n} n^2$ and no less than $(0.6^2 + 0.8^2)^0 n^2 + (0.6^2 + 0.8^2)^1 n^2 + ... + (0.6^2 + 0.8^2)^{log_{\frac{5}{3}} n} n^2$
Thus $F(n) = \Theta(n^2 logn)$
I got this question CORRECT.

Ex 3.30

a)

$T(1) = 1$
$T(n) = 2T(n-1) + 1, n > 1$
I got this question CORRECT.

b)

$U(1) = 1$
$U(n) = U(n-1) + T(n-1) + U(n-1), n > 1$
Since we have: $T(n) = 2^n - 1, n \geq 1$
Then $U(n) = 2U(n-1) + 2^{n-1} - 1, n > 1$
I got this question CORRECT.

c)

$W(1) = 1$
$W(n) = W(n-1) + W(n-1) + W(n-1) = 3W(n-1), n > 1$
I got this question CORRECT.

Extra Question 1

a)

Given string $s1[1..l_1]$ and $s2[1..l_2]$, find the largest k that exist series $1 \leq a_1 < a_2 < a_3 < ..a_k \leq l_1$ and $1 \leq b_1 < b_2 < b_3 < ..b_k \leq l_2$ that for $i \in [1, k]$, $s1[a_i] = s2[b_i]$
I got this question CORRECT.

b)

Let $GCS(i, j)$ be the length of the greatest common string for $s1[1..i]$ and $s2[1..j]$

$$GCS(i,j) = \begin{cases} 0 & , i = 0 \; or \; j = 0 \\ 1 + GCS(i-1, j-1) & , s1[i] = s2[j] \\ max\{GCS(i-1, j), GCS(i, j-1)\} & , s1[i] \neq s2[j] \end{cases} \quad (1)$$

So the solution of the original problem is $GCS(l1, l2)$
I got this question CORRECT.

Extra Question 2

a)

Given an integer n and a 2-D array $Val[i, j], 1 \leq i < j \leq n$, find an integer m and indices $0 < i_1 < i_2 < .. < i_m < n$ that maximize the sum $Val[0, i_1] + Val[i_1, i_2] + ... + Val[i_m, n]$
I got this question CORRECT.

b)

Let $Best(l)$ be the largest sum that described above for cutting up $Wood[1..l]$.
i.e maximum of the sum $Val[0, i_1] + Val[i_1, i_2] + ... + Val[i_k, l]$, where $0 < i_1 < i_2 < .. < i_k < l < n$

$$Best(l) = \begin{cases} 0 & , l = 0 \\ \max_{0 \leq k < l}\{Best(k) + Val[k, l]\} & , l > 0 \end{cases} \quad (2)$$

So the solution of the original problem is $Best(n)$
I got this question CORRECT.

Extra Question 3

a)

At beginning we call $GCS(l1, l2, s1, s2)$

Algorithm 2 Extra 3a

function $GCS(i, j; ; s1, s2)$
  if $i = 0$ *or* $j = 0$ **then**
    return 0
  end if
  if $s1[i] = s2[j]$ **then**
    return $GCS(i-1, j-1, s1, s2) + 1$
  else
    return $max\{GCS(i-1, j, s1, s2), GCS(i, j-1, s1, s2)\}$
  end if
end function

I got this question CORRECT.

b)

At beginning we call $GCSDriver(l1, l2, s1, s2)$

Algorithm 3 Extra 3b

function $GCSDriver(l1, l2; ; s1, s2)$
  Create look-up table $Look[1..l1, 1..l2]$ to store the result of subproblems. All elements initialized to -1.
  return $GCS(l1, l2, s1, s2, Look)$
end function
function $GCS(i, j; ; s1, s2, Look[1..l1, 1..l2])$
  if $i = 0$ *or* $j = 0$ **then**
    return 0
  end if
  $CurrAns \leftarrow Look[i][j]$
  if $CurrAns \neq -1$ **then**
    return $CurrAns$
  end if
  if $s1[i] = s2[j]$ **then**
    $CurrAns \leftarrow GCS(i-1, j-1, s1, s2, Look) + 1$
  else
    $CurrAns \leftarrow max\{GCS(i-1, j, s1, s2, Look), GCS(i, j-1, s1, s2, Look)\}$
  end if
  $Look[i][j] \leftarrow CurrAns$
  return $CurrAns$
end function

I got this question CORRECT.

Extra Question 4

At beginning we call $GCSDriver(l1, l2, s1, s2)$

Algorithm 4 Extra 4

1: **function** $GCSDriver(l1, l2; ; s1, s2)$
2: Create look-up table $Look[1..l1, 1..l2]$ to store the result of subproblems. All elements initialized to -1.
3: Create empty table $Path[1..l1, 1..l2]$ of $pair(i : int, j : int)$ to records the solution path.
4: $len \leftarrow GCS(l1, l2, s1, s2, Look, Path)$
5: $PrintPath(l1, l2, Path, s1)$
6: **return** len
7: **end function**
8: **function** $GCS(i, j; ; s1, s2, Look[1..l1, 1..l2], Path[1..l1, 1..l2])$
9: **if** $i = 0$ or $j = 0$ **then**
10: **return** 0
11: **end if**
12: **if** $Look[i][j] \neq -1$ **then**
13: **return** $Look[i][j]$
14: **end if**
15: **if** $s1[i] = s2[j]$ **then**
16: $Look[i][j] \leftarrow GCS(i-1, j-1, s1, s2) + 1$
17: $Path[i][j] \leftarrow (i-1, j-1)$
18: **else**
19: **if** $GCS(i-1, j, s1, s2, Look, Path) > GCS(i, j-1, s1, s2, Look, Path)$ **then**
20: $Look[i][j] = GCS(i-1, j, s1, s2, Look, Path)$
21: $Path[i][j] \leftarrow (i-1, j)$
22: **else**
23: $Look[i][j] = GCS(i, j-1, s1, s2, Look, Path)$
24: $Path[i][j] \leftarrow (i, j-1)$
25: **end if**
26: **end if**
27: **return** $Look[i][j]$
28: **end function**
29: **procedure** $PrintPath(i, j; ; Path[1..l1, 1..l2], s1)$
30: **if** $i > 0$ and $j > 0$ **then**
31: $(r, s) \leftarrow Path[i][j]$
32: $PrintPath(r, s, Path, s1)$
33: **if** $r = i - 1$ and $s = j - 1$ **then**
34: $Print(s1[i])$
35: **end if**
36: **end if**
37: **end procedure**

I got this question CORRECT.
Note: We can compute and store $GCS(i-1, j, s1, s2, Look, Path)$ and $GCS(i, j-1, s1, s2, Look, Path)$ into temporal variables before line 19. Doing this could potentially save one function calls for each item.(Although the function call will just look up the table and return the value).