Dayou Du
N13825075
dd2645@nyu.edu

Homework 6-SA
Fundamental Algorithms

2017-10-26

Ex 11.20

a)

0	0	0	0	0
20000	20000	20000	10000	
24000	120000	11000		
60000	16000			
26000				

I got CORRECT on this question.

b)

$$mults(i,j) = \begin{cases} 0 & ,i = j \\ min_{i \le k < j}\{mults(i,k) + D[i]D[k+1]D[j+1] + mults(k+1,j)\} & ,i < j \end{cases} \quad (1)$$

I got CORRECT on this question.

c)

Algorithm 1 Ex 11.20c

```
function LeastChainMM(D[1..(n+1)]; ;)
    Create Empty table Look[1..n, 1..n] where we will store Look[i, j] = mults(i, j)
    for i ∈ [1, n] do
        Look[i][i] = 0
    end for
    for step ∈ [1, n − 1] do
        for i ∈ [1, n − step] do
            j ← i + step
            result ← +∞
            for k ∈ [i, j − 1] do
                result ← min{result, Look[i, k] + D[i]D[k + 1]D[j + 1] + Look[k + 1][j]}
            end for
            Look[i][j] ← result
        end for
    end for
    return Look[1][n]
end function
```

I got CORRECT on this question.

Ex 11.23

$$F(i,j) = \begin{cases} 1 & ,i = 0 \; and \; j > 0 \\ 0 & ,i > 0 \; and \; j = 0 \\ 0.5F(i-1,j) + 0.5F(i,j-1) & ,i > 0 \; and \; j > 0 \end{cases} \tag{2}$$

Algorithm 2 Ex 11.23

function $WinProbDriver(n)$
    Create an empty table $Look[1..n, 1..n]$ with all elements initialized to -1.
    return $WinProb(n, n, Look)$
end function
function $WinProb(i, j;; Look[1..n, 1..n])$
    if $i = 0$ **then**
        return 1
    end if
    if $j = 0$ **then**
        return 0
    end if
    if $Look[i][j] = -1$ **then**
        $Look[i][j] \leftarrow 0.5WinProb(i-1, j, Look) + 0.5WinProb(i, j-1, Look)$
    end if
    return $Look[i][j]$
end function

At beginning we call $WinProbDriver(10)$
I got CORRECT on this question.

a)

For every $1 \le i \le n$ and $1 \le j \le n$, $WinProb(i,j)$ will call 2 recursion calls in the first time, and 0 recursion calls in the next time. Thus totally we have $2n^2$ recursive calls.
For $n = 10$, we have 200 recursive calls.
I got CORRECT on this question.

b)

Algorithm 3 Ex 11.23b

function $WinProb(n)$
    Create an empty table $Look[0..n, 0..n]$
    for $k \in [1, n]$ **do**
        $Look[0][k] \leftarrow 1$
        $Look[k][0] \leftarrow 0$
    end for
    for $i \in [1, n]$ **do**
        for $j \in [1, n]$ **do**
            $Look[i][j] = 0.5Look[i-1][j] + 0.5Look[i][j-1]$
        end for
    end for
    return $Look[n][n]$
end function

At beginning we call $WinProb(10)$
I got CORRECT on this question.

Ex 11.27

Create array $R_{value}[0..n]$ where $R_{value}[j]$be the greatest $R\text{-}value$ that can result from a partitioning of the first j buildings.

$$R_{value}[j] = \begin{cases} 0 & ,j = 0 \\ max_{0 \le k < j}\{R_{value}[k] + Sign(k+1, j)Weight(k+1, j)\} & ,0 < j \le n \end{cases} \tag{3}$$

The answer is $R_{value}[n]$

I got CORRECT on this question.

Ex 11.28

$$Sum[i] = \begin{cases} 0 & ,i = 0 \\ Sum[i-1] + S[i] & ,i > 0 \end{cases} \tag{4}$$

Let $MinCost[i][j]$ be the minimum total cost for creating one restaurant from restaurant $i, i+1, ..., j$

$$MinCost[i][j] = \begin{cases} 0 & ,i = j \\ min_{i \le k < j}\{MinCost[i][k] + MinCost[k+1][j] + max\{Sum[k] - Sum[i], Sum[j] - Sum[k+1]\}\} & ,i < j \end{cases} \tag{5}$$

The answer is $MinCost[1][n]$

The recursion is CORRECT. However,the index for calculation of the size is not totally correct. The lesson from this is that, when we have two or more recursive equations, we have to make clear about the meaning for each index. e.g. Included?Excluded?

The answer after fix is listed below:

$$MinCost[i][j] = \begin{cases} 0 & ,i = j \\ min_{i \le k < j}\{MinCost[i][k] + MinCost[k+1][j] + \\ \qquad max\{Sum[k] - Sum[i-1], Sum[j] - Sum[k]\}\} & ,i < j \end{cases} \tag{6}$$

Ex 11.29

Note: $Val(char)$ is defined as a real valued function, thus the value of a char *could be negative*

$$weightedGCS[i][j] = \begin{cases} 0 & ,i = 0 \ or \ j = 0 \\ weightedGCS[i-1][j-1] + max\{Val(Astring[i]), 0\} & ,Astring[i] = Bstring[j] \\ max\{weightedGCS[i-1][j], weightedGCS[i][j-1]\} & ,Astring[i] \neq Bstring[j] \end{cases} \tag{7}$$

There's actually a small *trick* behind the second equation(when $Astring[i] = Bstring[j]$):
The real situation when $Val(Astring[i]) < 0$ is a little bit more complicated here, the equation should be $max\{weightedGCS[i-1][j-1], weightedGCS[i-1][j], weightedGCS[i][j-1]\}$. However,we already know that the value of the last char($Astring[i] and Bstring[j]$) is negative. Thus ultimately we will *throw them away* at some time, then we don't have to consider the other two situations.

I got CORRECT on this question.

Node: The handout solution for this question did NOT consider the situation when $Val(char) < 0$

Ex 11.30

$$Best[i] = \begin{cases} Val[0][1] & ,i = 1 \\ max\{Val[0][i], max_{0 < k < i}\{Best[k] + Val[k][i] - 1\}\} & ,1 < i \le n \end{cases} \tag{8}$$

I got CORRECT on this question.

Ex 11.35

a)

In fact, the problem is : find the GCS of S and $S_{reverse}$.
Let $GCS(i,j)$ be the length of the longest common subsequence of $S[1..i]$ and $S_{reverse}[1..j]$
Note that we have $S[k] = S_{reverse}[n+1-k]$

$$GCS(i,j) = \begin{cases} 0 & ,i = 0 \; or \; j = 0 \\ 1 + GCS(i-1,j-1) & ,S[i] = S[n+1-j] \\ max\{GCS(i-1,j), GCS(i,j-1)\} & ,S[i] \neq S[n+1-j] \end{cases} \tag{9}$$

Then the answer of the original question is $GCS(n,n)$
I got CORRECT on this question.
Note: In this question I reversed j for a better illustration of the relation between origin GCS problem.

b)

Let $GPS(i,j)$ be the length of longest palindromes subsequence in $S[i..j]$,$1 \leq i \leq j \leq n$

$$GPS(i,j) = \begin{cases} 2 + GPS(i+1,j-1) & ,i > j \; and \; S[i] = S[j] \\ max\{GPS(i+1,j), GPS(i,j-1)\} & ,i > j \; and \; S[i] \neq S[j] \\ 1 & ,i = j \\ 0 & ,i > j \end{cases} \tag{10}$$

Then the answer of the original question is $GPS(1,n)$
I got CORRECT on this question.

c)

If, in question a), required that the two subsequence have to be the same, then it would be equivalent to question b).
In other words, (a) differed from (b) that (a) doesn't require the two subsequence to be the same.
I got CORRECT on this question.

d)

$acbac$
I got CORRECT on this question.

e)

The answer(greatest length) of the two problem are the same.
i.e. The length of the GCS of S and $S_{reverse}$ is the same as the length of greatest palindrome subsequence in S
I got CORRECT on this question.

11.theDogAteMyDecisionsTable

Algorithm 4 Ex 11.theDogAteMyDecisionsTable

function $PrintGCS(;; A[1..m], B[1..n], Look[1..m, 1..n])$
    $gcsLen \leftarrow Look[m][n]$
    Create empty array $GCS[1..gcsLen]$ to store the answer
    $i \leftarrow m$, $j \leftarrow n$, $k \leftarrow gcsLen$
    while $i \neq 0$ and $j \neq 0$ **do**
        if $A[i] = B[j]$ and $Look[i][j] = Look[i-1][j-1] + 1$ **then**
            $GCS[k] \leftarrow A[i]$
            $i \leftarrow i - 1$, $j \leftarrow j - 1$, $k \leftarrow k - 1$
        else
            if $Look[i][j] = Look[i-1][j]$ **then**
                $i \leftarrow i - 1$
            else
                $j \leftarrow j - 1$
            end if
        end if
    end while
    $Print(GCS)$
end function

I got CORRECT on this question.
Note: Here we use a iterative solution, not a recursive one. The reason is we already know the length of the answer, thus we don't have to use a postorder printing to create GCD from front to back.

11.zzzw

Let $Foodness(i, j, left, right)$ be the greatest possible value of $Foodness[R]$ we can get from applying $A[i..j]$ to a subtree rooted at R, where $left$ is the number of leftward descending edges on the path from the root to R, and $right$ is the number of rightward descending edges on the path.

$Foodness(i, j) =$

$$\begin{cases} 0 & , i > j \\ max_{i \leq k \leq j}\{2Foodness(i, k-1) + A[k] + 3Foodness(k+1, j)\} & , i \leq j \end{cases} \tag{11}$$

The solution to the problem is $Foodness(1, n)$
I got CORRECT on this question.