ALGORITHMS

Assignment 2 Report

Group: 6

Student ID: 202020681	name: 성민규
Student ID: 202020673	name: 안관우
Student ID: 201921166	name: 정의철

April 3, 2021

1 Exercise 1

1.1

If m = 5 and n = 4, the worst case is

	[0]	[1]	[2]	[3]	[4]	[5]
[0]	$\leftarrow$	$\leftarrow$				
[1]			$\nwarrow$			
[2]				$\nwarrow$		
[3]					$\nwarrow$	
[4]						$\nwarrow$

PRINT-LCS(b, X, i, j)	times
if i ==0 or j ==0	
return	
if b[i,j] == "$\nwarrow$"	
PRINT-LCS(b, X, i - 1, j - 1)	n
print x_i	n
elseif b[i,j] == "$\uparrow$"	
PRINT-LCS(b, X, i - 1, j)	
else PRINT-LCS(b, X, i, j - 1)	m - n

n + n + m - n = m + n
Thus, run time of PRINT-LCS(b, X, i, j) is O(m+n)
$T(n, m) = T(n-1, m-1) + 1$
O(m+n)

1.2

Algorithm 1 PRINT LCS NEW

```
procedure PRINT_LCS_NEW(c, X, Y, i, j) {X[x_1, x_2, ..., x_m], Y[y_1, y_2, ..., y_n]}
    if i == 0 or j == 0 then
        return
    end if
    if c[i, j]! = c[i, j − 1] and c[i, j]! = c[i − 1, j] then
        PRINT_LCS_NEW(c, X, Y, i − 1, j − 1)
        print x_i
    elseif c[i − 1, j] ≥ c[i, j − 1]
        PRINT_LCS_NEW(c, X, Y, i − 1, j)
    else
        PRINT_LCS_NEW(c, X, Y, i, j − 1)
    end if
```

2 Exercise 2

2.1 i) (5 point)

모든 원소의 값이 같은 배열 A에서 pivot을 가장 오른쪽 값으로 정하고 Quicksort를 시행할 경우, pivot과 다른 모든 원소가 동일하기 때문에 PARTITION(A, p, r)이 작동하면서 pivot의 왼쪽에 모든 원소가 정렬될 것이다.

이는 0개의 원소를 가지는 subarray, n-1개의 원소를 가지는 subarray로 나뉘는 Worst Case의 경우와 유사하고 다음과 같은 점화식으로 표현될 수 있다.

$T(n) = T(n-1) + T(0) + \Theta(n)$

$T(n) = T(n-1) + \Theta(n)$

$T(n) = \Theta(n^2)$

결국 running time은 $\Theta(n^2)$가 될 것이다.

2.2 ii) (5 point)

```
QUICKSORT_NEW(A,p,r) =0
if p < r then
    (q, t) = PARTITION_NEW(A, p, r)
    QUICKSORT_NEW(A, p, q-1)
    QUICKSORT_NEW(A, t+1, r) =0
```

2.3 iii) (7 point)

```
PARTITION_NEW(A,p,r) =0
```

```
x = A[p]
q = p
t = p
for j = p + 1 to r do
   if A[j] < x then
      t = t + 1
      a = A[j]
      A[j] = A[t]
      A[t] = A[q]
      A[q] = a
      q = q + 1
   else if A[j] == x then
      t = t + 1
      exchange A[t] with A[j]
   end for
   return  (q, t)
   =0
```

3 Exercise 3

3.1

Algorithm 2 Recursive insertion sort

procedure INSERTION_SORT_NEW(A, n) $\{A[1...n]\}$
  for $j = 2$ to $A.length$ **do**
    $key \leftarrow A[j]$
    $i \leftarrow j - 1$
    $index \leftarrow$ binary_search(A, 1, i, key)
    while $i \geq index$ **do**
      $A[i + 1] \leftarrow A[i]$
      $i \leftarrow i - 1$
    end while
    $A[i + 1] \leftarrow key$
  end for
  return A
end procedure=0

3.2

이진 탐색을 사용하면, $O(n \log n)$ 복잡도를 갖는다. j번째 iteration에서 이진 탐색 procedure는 $log_2(j-1)$ 만큼의 비교를 진행하기 때문에,

$$\begin{aligned}\sum_{j=2}^{n} \log_2(j-1) &= \sum_{j=1}^{n-1} \log_2(j) \\ &= n\log_2(n+1) + 2^{\log_2(n+1)+1} + 2 \\ &= O(n \log n)\end{aligned} \tag{3.1}$$

으로 정리할 수 있다.

3.3

$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \log(n-1) +$ $c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n}(t_j - 1) + c_7 \sum_{j=2}^{n}(t_j - 1) + c_8(n-1)$
에서 최악의 상황일 때 $t_j = j$가 된다. 따라서
$\sum_{j=2}^{n}(j-1)n(n-1)/2$ 이므로 수행시간은 $an^2 + bn + c + dlogn$로 나타낼 수 있다. 따라서 최악의 상황에서 수행시간은 $O(n^2)$

4 Exercise 4

Algorithm 3 calculate the height of the smallest tree

```
procedure COMPUTE_SMALLEST_H(x)
  if x − 34 ≤ 0 and (x − 11)/2 ≤ 0 then
    return x
  elseif x − 34 ≤ 0
    return COMPUTE_SMALLEST_H((x − 11)/2)
  elseif (x − 11)/2 ≤ 0
    return COMPUTE_SMALLEST_H(x − 34)
  else
    return min (COMPUTE_SMALLEST_H(x − 34) , COMPUTE_SMALLEST_H((x − 11)/2))
  end if
```