HW05 for ECE 9343

Tongda XU, N18100977

December 5, 2018

1 Question 1: CLRS Exercise 15.1-3

BOTTOM-UP-CUT-ROD(p, n, c)

```
1  r[] = c
2  for j = 1 to n
3      for i = 1 to j
4          r[i] ← max(p[i] + r[j − i] − c)
5  return r[n]
```

2 Question 2: CLRS Exercise 15.1-4

MEMOIZED-CUT-ROD(p, n, m, s)

```
1  if m[n] > −1
2      return m[n]
3  else
4      for i ← 1 to n
5          m[n] ← max(p[i] + r[n − i])
6          s[n] ← i
7      return m[n]
```

3 Question 3: CLRS Exercise 15.4-3

LCS(X, Y)

```
1  DP ← [][]
2  return LSC-Aid(X.length, Y.length)
```

```
LSC-Aid(i, j)
 1  if i = 0 or j = 0
 2       DP[i][j] ← 0
 3  else
 4       if X[i] = Y[j]
 5            if DP[i − 1][j − 1] = NIL
 6                 DP[i − 1][j − 1] = LSC-Aid(i − 1, j − 1)
 7            DP[i][j] ← DP[i − 1][j − 1] + 1
 8       else
 9            if DP[i − 1][j] = NIL
10                 DP[i − 1][j] = LSC-Aid(i − 1, j)
11            if DP[i][j − 1] = NIL
12                 DP[i][j − 1] = LSC-Aid(i, j − 1)
13            DP[i][j] = max{DP[i][j − 1], DP[i − 1][j]}
14  return DP[i][j]
```

4 Question 4: CLRS Exercise 15.4-5

This is easy to construct from bottom to top, and straightforward to see a time complexity of $\Theta(n^2)$:

```
Longest-Mono-Increase(s)
1  DP ← [][]
2  DP[1] ← s[1]
3  for i ← 1 to n
4       for j ← i − 1 downto 1
5            if DP[j].end < s[i]
6                 DP[i] ← DP[i].length < DP[j].length + 1?DP[j] + s[i] : DP[i]
7            else DP[i] ← DP[i].length < DP[j].length?DP[j] : DP[i]
8  return DP[s.length]
```

5 Question 5: CLRS Exercise 15.1

It is easy to implement a memorized recursive algorithm, but very hard to build from down to top:

```
Longest-simple-path(s, t)
1  DP[] ← −1
2  return Longest-simple-path-aid(s, t)
```

LONGEST-SIMPLE-PATH-AID(s, t)

1 **if** $s \neq t$
2 **if** $DP[s] = -1$
3 $DP[s] \leftarrow \max_{v \in s.adjList} \{\text{WEIGHT}(s, v) + \text{LONGEST-SIMPLE-PATH-AID}(v, t)\}$
4 **return** $DP[s]$
5 **else return** 0

the $DP[s]$ is a array with length V, all overlapping subproblem is solved by memory, so $DP[s]$ cost $\Theta(V)$ time to construct. In each query, it cost $s.adjList.length()$ time, and in total it cost $O(E)$ time. So Longest-simple-path cost $O(E + V)$ time to compute.

6 Question 6: CLRS Exercise 16.1-1

This process fill a grid of $\frac{1}{2}n^2$ and take space and time of $\Theta(n^2)$. Greedy is one-pass and take only $\Theta(n)$.

AS-ADI(a)

1 $DP = [][]$
2 **return** AS-ADI$(0, a.length)$

AS-ADI(i, j)

1 **for** $m \leftarrow j - 1$ **downto** $i + 1$
2 **if** $a[m].f \leq a[j].s$ and $a[m].s \geq a[i].f$
3 $S[i][j].push(a[m])$
4 **if** $S[i][j] = \emptyset$
5 $DP[i][j] \leftarrow 0$
6 **else**
7 **if** $DP[i][j] = NIL$
8 $DP[i][j] \leftarrow \max_{a[k] \in S[i][j]} \{\text{AS-ADI}(i, k) + 1 + \text{AS-ADI}(k, j)\}$
9 **return** $DP[i][j]$