

## Homework 4

Posted: Thursday, 10/30/2025

Rithik Chowdam

Solutions: 11:59PM, Friday, 11/07/2025

1. Solve the Exercise 14.1-3 on page 373 from CLRS. (20 pt)

**ROD\_CUTTING\_DP(p, n, c)**

$R = [0 \dots n]$

$R[0] = 0$

For  $j = 1$  to  $n$

$Q = -\infty$

For  $i = 1$  to  $j$

$Q = \max(Q, (p[i] + r[j-i]) - c)$

$R[j] = Q$

Return  $r[n]$

**End Procedure**

2. Choose a recurrence relation which is denoted by  $T(n)$ . Design recursive and a bottom-up dynamic programming algorithms for  $T(n)$ . Bottom-up DP solution should be faster than the recursive solution. (20 pt)

Such a problem can be a house robber problem where a robber tries to steal as much money from a row of houses without robbing houses that are next to each other. The input is an array denoting the stealable money from each house and the output is a maximized amount of money a robber can steal given the preceding rule.

$T(n)$  for this problem:

$$T(n) = T(n-1) + T(n-2) + c$$

Recursive:

**RECURSIVE-ROBBER(m, n)**

If  $n == 0$

Return 0

If  $n == 1$

Return  $m[0]$

```
First_choice = m[n] + RECURSIVE_ROBBER(m, n-2)
```

```
Second_choice = RECURSIVE_ROBBER(m, n-1)
```

```
Return max(first_choice, second_choice)
```

**End Procedure**

DP:

**DP\_ROBBER(m, n)**

```
Table = [0...n+1]
```

```
Table[0] = 0
```

```
Table[1] = m[0]
```

For I = 2 to n

```
Table[i] = max(m[I - 1] + table[I - 2], table[I - 1])
```

```
Return table[n]
```

**End Procedure**

3. Solve Exercise 14.3-3 on page 393 in CLRS (**20 pt**)

Yes, optimal substructure is still maintained. If there is a midpoint k in the current parenthesization I -> j where  $A_{ij} = A_{ik} * A_{(k+1)j}$  and the optimal solution for the chosen parenthesization is the cost of  $A_{ik}$  + the cost of  $A_{(k+1)j}$  + the cost of multiplying  $A_{ik}$  and  $A_{(k+1)j}$ , then the optimal solutions for the subproblems would be the max for the subproblem. Then these subproblems would be combined to form the current optimal solution. Thus, this variant of matrix multiplication maintains optimal substructure.

4. Solve Problem 14-2 on page 407 in CLRS (**20 pt**)

**LONGEST\_PALINDROME\_SUBSEQUENCE(str)**

```
Str_reverse = str.reverse()
```

```
Table = [0...str.length() + 1][0...str.length() + 1]
```

```
For I .. str.length() + 1
```

```

Table[0][i] = 0
Table[i][0] = 0

For I = 1 ... str.length() +1, j = 1 ... str.length() + 1
    If str[I - 1] == str_reverse[j - 1]
        Table[i][j] = 1 + table[i-1][j-1]
    Else
        Table[i][j] = max(table[i-1][j], table[i][j - 1])

Result = ""
I = str.length() -1
J = str.length() - 1

While I > 0 and j > 0
    If str[i-1] == str_reverse[j - 1]
        Result += str[I - 1]
        J -= 1
        I -= 1
    Else
        If table[i-1][j] > table[i][j-1]
            I -= 1
        Else
            J -= 1

Return result
End Procedure

```

5. Solve Exercise 15.1-3 on page 425 in CLRS (20 pt) 1

