Name : Kang Chin Shen
Matriculation ID : 0141921.

Question 1

Given a string x : _____ (n characters).

- Find the minimum number of characters to add to the string so that it becomes a palindrome.

Part 1a - Let D[i,j] be the minimum number of characters that need to be inserted to make the string x[i] ... x[j] a palindrome.

- i is pointing to first character of the string; j is pointing to the last character of the string.

∴ Recurrence Formulation:

$$D[i,j] = \begin{cases} D[i+1, j-1] & \text{if } x[i] == x[j] \\ MIN(D[i+1,j], D[i,j-1]) + 1 & \text{if } x[i] != x[j]. \end{cases}$$

Base cases:

$$D[i,j] = \begin{cases} 0 & \text{if } i == j. \\ (0 \text{ if } x[i] == x[j] \text{ else } 1) & \text{if } i == j-1. \\ INTEGER\_MAX & \text{if } i > j. \end{cases}$$

Part 1b.

- Defining ① base case: if i==j, 1 character by itself is a palindrome, return 0.

- Suppose we have a non-palindrome string, and to break this into smaller problem, we have the following cases:

① a b c a — x[i] == x[j]

- This implies that D[i,j] == D[i+1, j-1] (a smaller problem.)

② a b c d — x[i] != x[j]

- Then, we have 2 ways of breaking the problem:

add d in front: d a b c d

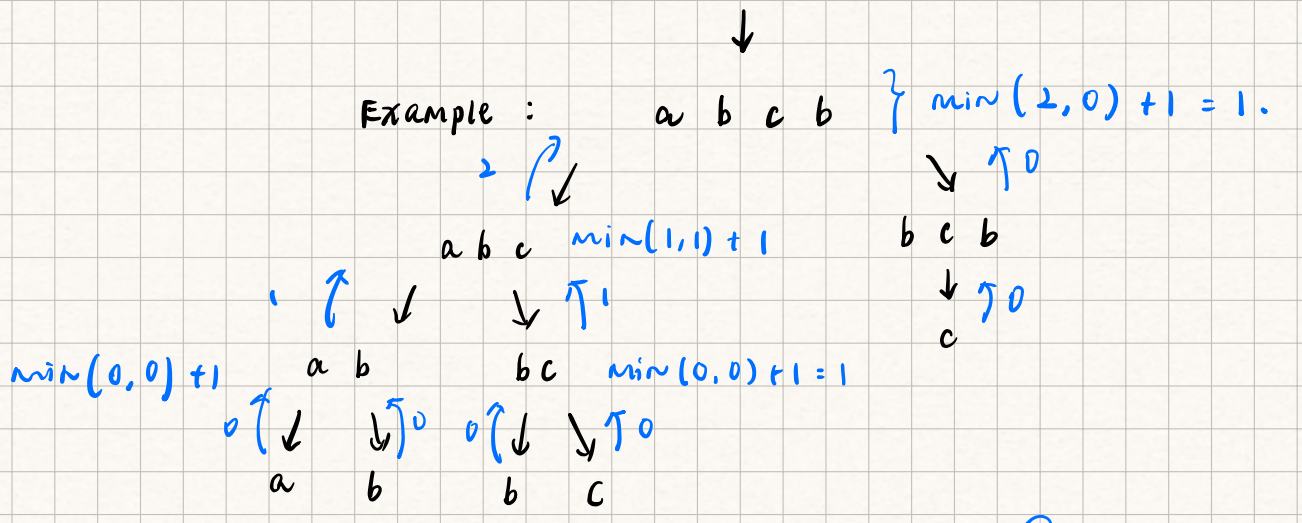
To define a smaller problem for this case, we take D[i, j-1]

add a at the end: a b c d a

To define a smaller problem for this case, we take D[i+1, j]

Since there are two ways of breaking the big problem, and we are trying to find the minimum number of characters to be added:

return MIN(D[i, j-1], D[i+1, j]) + 1 .



∴ Hence, the recurrence formulation is combining ① and ② :

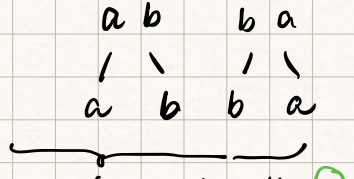
$$D[i,j] = \begin{cases} D[i+1, j-1] & \text{if } x[i] == x[j] \\ MIN(D[i+1, j], D[i, j-1]) + 1 & \text{if } x[i] != x[j]. \end{cases}$$

For base cases, we have to consider the following cases:

i == 0 a a j == 1

since x[i] == x[j],

- it will enter ①, finding D[1, 0], in which i > j, this should return 0 instead.



works well with ① base case.

↓

② base case:
- Go one level above,
- if (i == j-1), return 0 if (x[i] == x[j]) else 1 .

③ base case:
- if (i > j) return INTEGER_MAX } incorrect input.

∴ Hence, the three base cases are:

$$D[i,j] = \begin{cases} 0 & \text{if } i == j. \\ (0 \text{ if } x[i] == x[j] \text{ else } 1) & \text{if } i == j-1. \\ INTEGER\_MAX & \text{if } i > j. \end{cases}$$

Part 1C - Refer Python code.

Question 2

Length	1	2	3	4	5	6	7	8	9	10
Price	1	5	7	9	10	17	17	20	23	30
index	0	1	2	3	4	5	6	7	8	9

*Assume index starts from 0

Part 2a *Assume price is not negative.

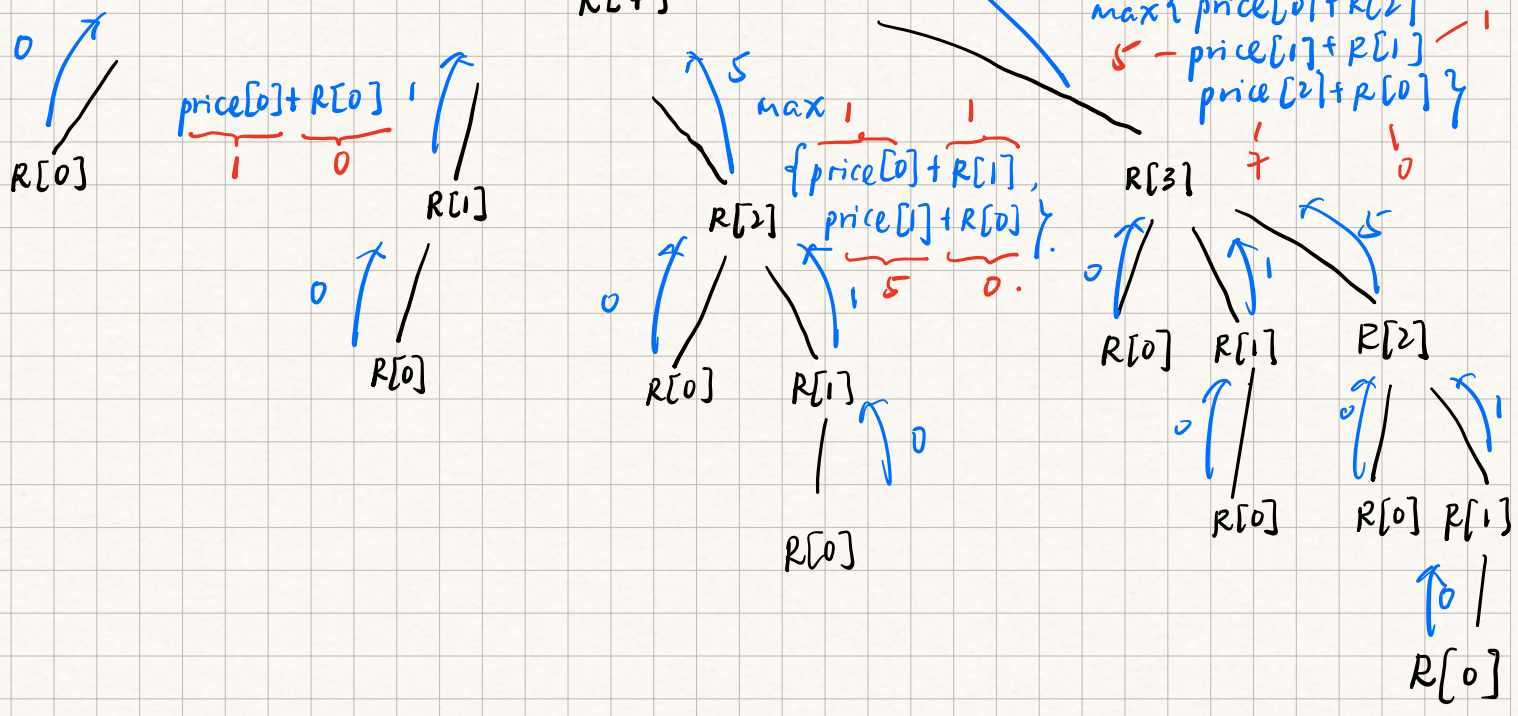
Recursive Formulation:

$$R[n] = \begin{cases} 0 \text{ if } n = 0 \\ \max\{ price[i] + R[n-i-1] \} \text{ for all } i \in \{0, \ldots, n-1\}. \end{cases}$$

max{
1 price[0] + R[3] 7
5 price[1] + R[2] 5
7 price[2] + R[1] 1
9 price[3] + R[0] } 0

10.

R[4]



Part 2b.

Iterative Pseudocode (Dynamic Programming): *Assuming price is not negative.

```
PROCEDURE FIND-MAX-REVENUE( PRICE, N )

    SET values to an array of 0's of size N+1
    SET values[0] to 0

    FOR i = 1 TO N
        SET max_value TO -1
        FOR j = 1 TO i
            SET max_value to MAX( max_value, price[j] + val[i-j-1] )
            SET values[i] = max_val.

    RETURN values[N]
```

Part 2c.

- Initial recurrence formulation assumed that sub-problems do not share resources.
- Violation of optimal substructure property as additional contraint is imposed.
- This implies sub-problems can no longer be solved independently.
