15.2-2

There are 2 functions needed to make it recusive. Matrix-chain-multiply(A, s, i, j) and Matrix-multiply(A, B)

```
Matrix-multiply(A, B):
    if len(A[0]) != len(B):
        return non-compatible error.

    for i = 1 to len(A):
        for j = 1 to len(B[0]):
            C[i,j] = 0
            for k = 1 to len(A[0]):
                C[i,j] = C[i,j] + A[i,k] + B[k,j]

    return C
```

```
Matrix-chain-multiply(A, s, i, j):
    if i = j:
        return A[i]

    left = Matrix-chain-multiply(A, s, i, s[i,j])
    right = Matrix-chain-multiply(A, s, s[i,j]+1, j)

    return matrix-multiply(left, Right)
```

15.2-4

The vertice are ordered pairs built from $\{A_1, A_2 \ldots A_n\}$, So the number of vertice is $\sum_{i=1}^{n} \sum_{j=1}^{n} 1 = \frac{n(n+1)}{2} = \Theta(n^2)$

The edges are the operation needed to solve matrix multiplication of $\{A_1 \ldots A_n\}$

This is the upper triangle of S matrix multiplied by n, that is

$$\sum_{i=1}^{n} \sum_{j=i}^{n} (j-i) \Rightarrow \Theta(n^3)$$

So there are n^2 vertices

and n^3 edges.

15.2-6

For a full parenthesization of n-element express, there must be a k in A that we can devide in to $B(A_1, A_2 \cdots A_k)$ and $C(A_{k+1} \cdots A_n)$

Then we have 2 matrix B, C

Multiply them takes X -1 parenthesis

X is 2. If we recur this down

B has k-1 parenthesis, C has n-k+1-1 parenthesis. In total there are k-1 + n-k+1-1 ~~Par~~parenthesis that is n-1

15.3-3

To prove that it exhibit optimal substructure, we need to prove that it can be splited into sub problems and subproblems have optimal solution.

To maximize scalar. we can split A to $\{A_1, A_2 \cdots A_k\} \cdot \{A_{k+1} \cdots A_n\}$ where $1 \le k \le n$

Assume we have a solution to $\{A_1, A_2 \cdots A_k\}$ that does not maximize $\{A_1, A_2 \cdots A_k\}$ we will always have another solution maximize $\{A_1, A_2 \cdots A_k\}$ So, the sub problem has optimal solution and therefore, maximizing the number of scalar exhibit optimal substructure.

15.3-6

Let's say the c_k is 0.

when we exchange currency 1 to currency n, there may exist a currency k that $d\,r_{1k} \cdot r_{kn} > d\,r_{1n}$

Then we can divide the problem into 2 subproblems, exchanging currency 1 to k then from k to n. The sub problems are optimal from the inference above and final solution is the combination of subproblems. So it exhibit optimal sub structure, and can be solved recursively.

the second case is when C_k is arbitury, even if there is a k that $dr_{ik} r_{kn} > dr_{1n}$, It might not be optimal, so it does not exhibit optimal sub structure.

15.4-2

```
Print_LCS (C, A, P, q):
    if p == 0 or q == 0:
        return

    if c[p, q] = c[p-1, q-1]:
        print A[p]

    else if c[p-1, q] >= c[p, q-1]:
        print_LCS(C, A, p-1, q)

    else:
        print_LCS(C, A, p, q-1)
```

15.4-5

```
Longest-sequence (L)
    For i in range L:
        Build Points (A, i)

    Sort points based on y value
    Build a table m with n*n entries
    Set all values in m be 0.
    Num = 1

    for x, y in Points:
        if m[x][num+1] == 0
            break

    for i=1 to n:
        if m[i][0] != 0:
            value = 1
        m[i][0] = m[i-1][0] + value
```

```
for i = 1 to n:
    for j = 1 to n:
        if m[i][j] != 0:
            value = 1
            m[i][j] = max(m[i-1][j], m[i][j-1])
                      + value

Sequence = []
pointer = m[n][n]
if pointer == φ:
    sequence = sequence + pointer

return sequence.
```

16.1-2

The difference between selecting the first activity to finish and the last activity to start is that it is the reversed version of selecting the first activity to finish.

It selects activity to start in descending order. This results in that the algorithm selecting the optimal solution on each stage in descending order. Therefore, this new approach results in optimal solution

16.1-3

	1	2	3
Start	2	3	5
finish	5	4	7
Duration	3	1	2

least duration will select $\{2\}$

But optimal solution is $\{1, 3\}$

	1	2	3	4	5
start	1	3	4	6	3
finish	3	4	6	7	10
over lap	0	1	2	2	3

least overlap will choose $\{1, 2, 3, 4\}$

but optimal solution is $\{1, 5\}$

	1	2	3
start	1	3	5
finish	3	6	10

Earliest will choose $\{1, 2\} \Rightarrow$ 5hrs

optimal should be $\{1, 3\} \Rightarrow$ 7hrs.