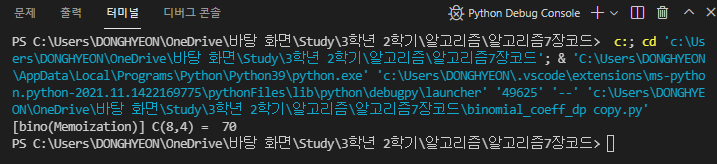
**10주차 과제**

1. **이항계수 알고리즘을 메모이제이션 기법으로 다시 구현하세요.**

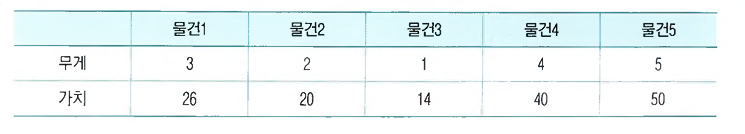
**# 소스**

|  |
| --- |
| def bino\_mem(n, k):      C = [[-1 for \_ in range(k+1)] for \_ in range(n+1)]      if k == 0 or k == n:          return 1      if C[n][k] != -1:          return C[n][k]      else:          C[n][k] = bino\_mem(n-1, k-1) + bino\_mem(n-1, k)      return C[n][k]                  # 최종 결과 |

**# 실행 결과**



**2. 알고리즘 7.6을 이용하여 다음의 0-1 배낭 채우기 문제를 해결하세요. 테이블을 만들고 모든 항목을 채워서 답을 구하는 과정을 보이세요. 배낭의 용량은 6입니다.**



**배낭 용량**

0 1 2 3 4 5 6

**물** 0 0 0 0 0 0 0 0

**건** 1 0 0 0 26 26 26 26

**의** 2 0 0 20 26 26 46 46

3 0 14 20 34 40 46 60

**수** 4 0 14 20 34 40 54 60

5 0 14 20 34 40 54 64

**3. 두 문자열 “DATA STRUCTURE”와 “PYTHON ALGORITHM”에 대해 LCS 테이블이 만들어지는 과정을 해당 프로그램을 실행하여 결과를 쓰세요.**

X = DATA STRUCTURE

Y = PYTHON ALGORITHM

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]

[0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2]

[0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4]

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4]

LCS = TART

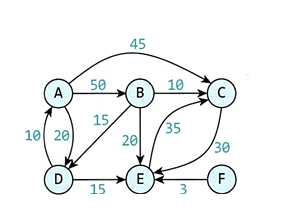
LCS(동적 계획) 4

**4. 위 문제에서 알고리즘 7.9를 이용하여 LCS를 추적하는 과정을 그리세요.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | P | Y | T | H | O | N | ‘ ‘ | A | L | G | O | R | I | T | H | M |
|  | ↑0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D | ↑0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | ↑0 | ←0 | ←0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| T | 0 | 0 | 0 | ↖1 | ←1 | ←1 | ←1 | ←1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |
| A | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | ↖2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| ‘ ‘ | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | ↑2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| S | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | ↑2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| T | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | ↑2 | ←2 | ←2 | ←2 | 2 | 2 | 3 | 3 | 3 |
| R | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | ↖3 | 3 | 3 | 3 | 3 |
| U | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | ↑3 | 3 | 3 | 3 | 3 |
| C | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | ↑3 | ←3 | 3 | 3 | 3 |
| T | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | ↖4 | 4 | 4 |
| U | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | ↑4 | 4 | 4 |
| R | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | ↑4 | 4 | 4 |
| E | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | ↑4 | ←4 | ←4 |

←↑↖

**5. 다음의 방향 그래프에 대해 Floyd 알고리즘을 이용하여 모든 정점간의 최단경로를 구하세요.**



**(1) 인접행렬을 정의하세요. (이때 방향이 존재하므로 인접행렬은 대각선상의 대칭이 아님)**

vertex = [ 'A', 'B', 'C', 'D', 'E', 'F']

# 두줄은 아래와 같음 나머지를 완성하세요.

weight = [ [ 0, 50, 45, 20, INF, INF ],

[ INF, 0, 10, 15, 20, INF ],

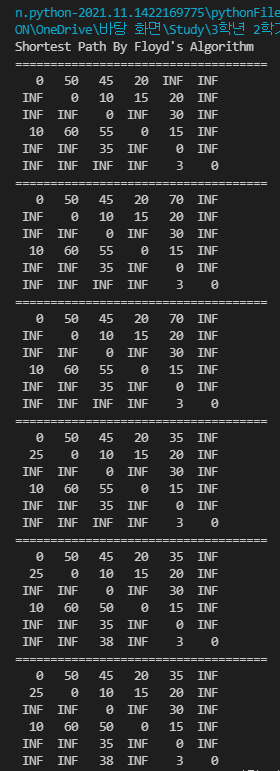
[ INF, INF, 0, INF, 30, INF ],

[ 10, INF, INF, 0, 15, INF ],

[ INF, INF, 35, INF, 0, INF ],

[ INF, INF, INF, INF, 3, 0 ] ]

**(2) 프로그램을 실행하여 결과를 출력하세요.**



**6. “HELLO WORLD”를 편집하여 “GAME OVER”로 수정하려고 합니다. 사용할 수 있는 연산은 삽입,삭제,대체입니다. 편집거리 프로그램을 수정하여 실행하고 그 결과를 출력하세요.**

