'''

Problem:

Ugly numbers are numbers whose only prime factors are 2, 3 or 5.

By convention, 1 is included.

'''

# for DP solution, rather than loop through all natural number

# and check whether it is ugly number, we plan to construct the ugly number one by one

# precisely, we should maintain 3 pointers {i2, i3, i5} and 3 next values{next\_mulp\_2,

# next\_mulp\_3, next\_mulp\_5}. each time we select the minimum from these 3 nxt values and

# updata the corresponding pointer.

# next\_mulp\_i = ugly\_data[pointer\_i] \* i

# pointer += 1

def ugly\_number\_DP(n):

ugly\_data = [0] \* (n)

ugly\_data[0] = 1

i2 = i3 = i5 = 0

next\_mulp\_2 = 2

next\_mulp\_3 = 3

next\_mulp\_5 = 5

# notice:

# for each i, we should use 3 separated if, rather than if--elif--else

# because there may exists equal numbers {next\_mulp\_2, next\_mulp\_3, next\_mulp\_5}

for i in range(1, n):

ugly\_data[i] = min(next\_mulp\_2, next\_mulp\_3, next\_mulp\_5)

if ugly\_data[i] == next\_mulp\_2:

i2 += 1

next\_mulp\_2 = ugly\_data[i2] \* 2

if ugly\_data[i] == next\_mulp\_3:

i3 += 1

next\_mulp\_3 = ugly\_data[i3] \* 3

if ugly\_data[i] == next\_mulp\_5:

i5 += 1

next\_mulp\_5 = ugly\_data[i5] \* 5

return ugly\_data

print(ugly\_number\_DP(150))

'''

A dynamic programming problem contains two properties:

(1) Overlapping subproblems, we can store the results for each subproblem,

so as to reduce calculation

(2) Optimal substructure

'''

def fib(n, lookup):

# base case

if n == 0 or n == 1:

return n

if lookup[n] != None:

return lookup[n]

return fib(n-1, lookup) + fib(n-2, lookup)

def fib\_tabulation(n, table):

table[0] = 0

table[1] = 1

if n <= 1:

return table[n]

for i in range(2, n + 1):

table[i] = table[i - 1] + table[i - 2]

return table[n]

n = 10

lookup = [None] \* (n + 1)

result = fib(n, lookup)

print(result)

result = fib\_tabulation(n, lookup)

print(result)