1. In dynamic programming, the technique of storing the previously calculated values is called \_\_\_\_\_\_\_\_\_\_\_

a) Saving value property

b) Storing value property

c) Memoization

d) Mapping

Answer: c

Explanation: Memoization is the technique in which previously calculated values are stored, so that, these values can be used to solve other subproblems.

2. When a top-down approach of dynamic programming is applied to a problem, it usually \_\_\_\_\_\_\_\_\_\_\_\_\_

a) Decreases both, the time complexity and the space complexity

b) Decreases the time complexity and increases the space complexity

c) Increases the time complexity and decreases the space complexity

d) Increases both, the time complexity and the space complexity

Answer: b

Explanation: The top-down approach uses the memoization technique which stores the previously calculated values. Due to this, the time complexity is decreased but the space complexity is increased.

3. Which of the following problems should be solved using dynamic programming?

a) Mergesort

b) Binary search

c) Longest common subsequence

d) Quicksort

Answer: c

Explanation: The longest common subsequence problem has both, optimal substructure and overlapping subproblems. Hence, dynamic programming should be used the solve this problem.

4. A greedy algorithm can be used to solve all the dynamic programming problems.

a) True

b) False

Answer: b

Explanation: A greedy algorithm gives optimal solution for all subproblems, but when these locally optimal solutions are combined it may NOT result into a globally optimal solution. Hence, a greedy algorithm CANNOT be used to solve all the dynamic programming problems.

5.Which technique can be used to get the nth fibonacci term?

a) Recursion

b) Dynamic programming

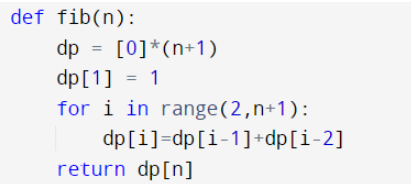
c) A single for loop

d) Recursion, Dynamic Programming, For loops

Answer: d

Explanation: Each of the above mentioned methods can be used to find the nth fibonacci term.

6. What is the time complexity of the following for loop method used to compute the nth fibonacci term?



a) O(1)

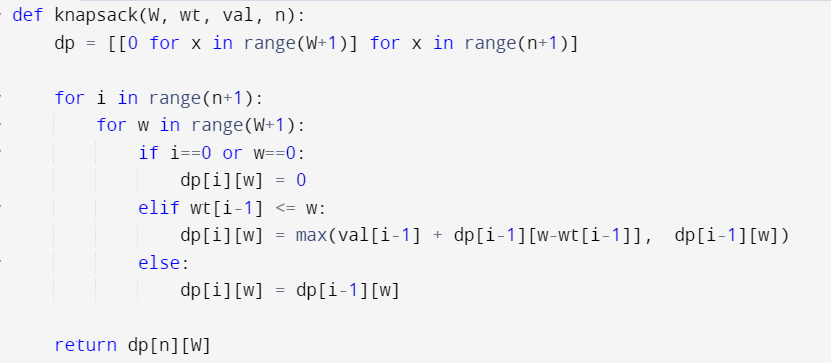
b) O(n)

c) O(n2)

d) Exponential

Answer: b

7.What is the time complexity of the following for loop method used to compute the knapSack term?



a)O(1)

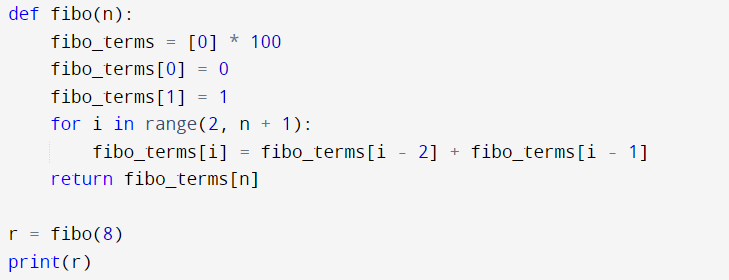
b)O(n)

c)O(W)

d)O(nW)

Answer: d

8. What will be the output when the following code is executed?

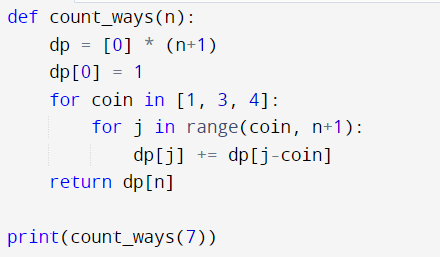


1. 34
2. 55
3. Compile error
4. 21

Answer: d

Explanation: The program prints the 8th fibonacci term, which is 21.

9. You are given infinite coins of denominations 1, 3, 4. What is the total number of ways in which a sum of 7 can be achieved using these coins if the order of the coins is not important?



a) 4

b) 3

c) 5

d) 6

Answer: c

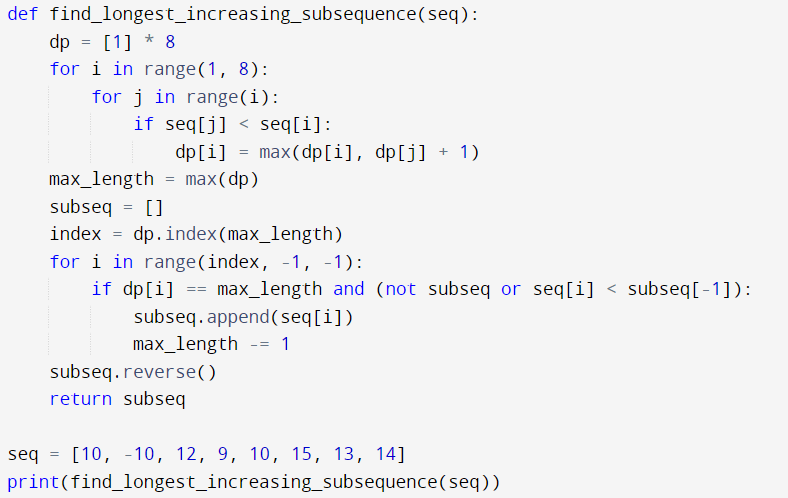
Explanation: A sum of 7 can be achieved in the following ways:

{1,1,1,1,1,1,1}, {1,1,1,1,3}, {1,3,3}, {1,1,1,4}, {3,4}.

Therefore, the sum can be achieved in 5 ways.

10. Find the longest increasing subsequence for the given sequence:

{10, -10, 12, 9, 10, 15, 13, 14}



a) {10, 12, 15}

b) {10, 12, 13, 14}

c) {-10, 12, 13, 14}

d) {-10, 9, 10, 13, 14}

Answer: d

Explanation: The longest increasing subsequence is {-10, 9, 10, 13, 14}.

Code bài 8

def fibo(n):

fibo\_terms = [0, 1] + [0] \* (n - 1)

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

fibo\_terms[i] = fibo\_terms[i - 2] + fibo\_terms[i - 1]

return fibo\_terms[n]

r = fibo(8)

print(r) #21

Code bài 9

def count\_ways(n):

dp = [0] \* (n+1)

dp[0] = 1

for coin in [1, 3, 4]:

for j in range(coin, n+1):

dp[j] += dp[j-coin]

return dp[n]

print(count\_ways(7)) #5

Code bài 10

def find\_longest\_increasing\_subsequence(seq):

n = len(seq)

dp = [1] \* n

for i in range(1, n):

for j in range(i):

if seq[j] < seq[i]:

dp[i] = max(dp[i], dp[j] + 1)

max\_length = max(dp)

subseq = []

index = dp.index(max\_length)

for i in range(index, -1, -1):

if dp[i] == max\_length and (not subseq or seq[i] < subseq[-1]):

subseq.append(seq[i])

max\_length -= 1

subseq.reverse()

return subseq

seq = [10, -10, 12, 9, 10, 15, 13, 14]

print(find\_longest\_increasing\_subsequence(seq)) # Output: [-10,9,10,13,14]