

def alg(m, n, x, y):

dp = [[0 for i in range(n + 1)]

for j in range(m + 1)]

for i in range(m + 1):

for j in range(n + 1):

if i == 0:

dp[i][j] = j

elif j == 0:

dp[i][j] = i

elif x[i - 1] == y[j - 1]:

dp[i][j] = 1 + dp[i - 1][j - 1]

else:

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

string = ""

//bottom right corner

i = m

j = n

while i \* j > 0:

# If current character in X and Y are same,

# then current character is part of

# shortest supersequence

if x[i - 1] == y[j - 1]:

string = x[i - 1] + string

i -= 1

j -= 1

# If current character in X and Y are different

elif dp[i - 1][j] > dp[i][j - 1]:

# Put current character of Y in result

string = y[j - 1] + string

# reduce values of j and index

j -= 1

else:

# Put current character of X in result

string = x[i - 1] + string

# reduce values of i and index

i -= 1

while i > 0:

string = x[i - 1] + string

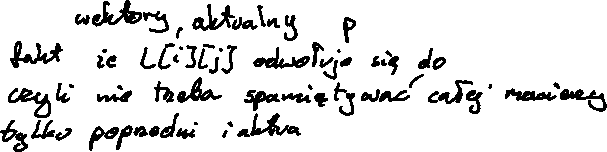
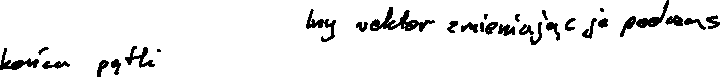
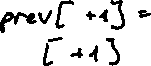
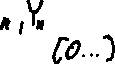
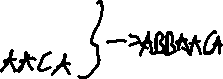
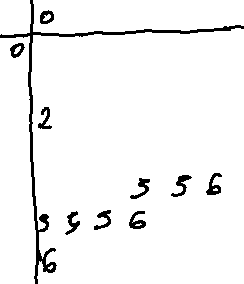
i -= 1

while j > 0:

string = y[j - 1] + string

j -= 1

return string



Utwórz mapę, aby zapamiętać wyniki

Recur, aby rozwiązać problem dla węzła root

Jeśli węzeł główny ma wartość NULL, zwróć 0 (przypadek podstawowy)

Jeśli odpowiedź na ten podproblem jest już zapisana w mapie, zwróć ją.

Teraz albo uwzględnij bieżący węzeł i wykonaj rekurencję dla jego wnuków

lub nie uwzględniaj bieżącego węzła i wyszukaj jego lewe i prawe dziecko.

Zapisz odpowiedź na ten problem równą maksimum z powyższych dwóch przypadków.

map = {}

def maxSum(root):

iV=[]

eV=[]

if (root == None):

return None

if (root in map):

return map[root]

//inc = root.data

iV.append(root)

if (root.left):

//inc += maxSum(root.left.left) + maxSum(root.left.right)

iV.append(left)

iV.append(right)

if (root.right):

inc += maxSum(root.right.left) + maxSum(root.right.right)

iV.append(left)

iV.append(right)

//exc = maxSum(root.left) + maxSum(root.right)

eV.append(^)

// map[root] = max(inc, exc)

// return max(inc, exc)

if sum(iV)> sum(eV):

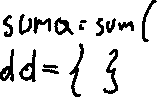
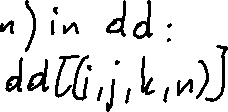
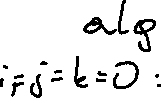
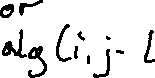
map[root]=iV

return iV

else:

map[root]=eV

return eV



(<https://medium.com/quick-code/longest-common-increasing-subsequence-lcis-1d2fc2713cd6> )

def LCIS(arr1, n, arr2, m):

table = [0] \* m

parent = [-1] \* m

for i in range(n):

current = 0

last = -1

for j in range(m):

if arr1[i] == arr2[j] and current + 1 > table[j]:

table[j] = current + 1

parent[j] = last

if arr1[i] > arr2[j] and table[j] > current:

current = table[j]

last = j

result = 0

pos = 0

for i in range(m):

if table[i] > result:

result = table[i]

pos = i

lcis = []

while pos != -1:

lcis.append(arr2[pos])

pos = parent[pos]

lcis.reverse()

return result, lcis

