PROGRAM-8

Python3 program for the above approach

# Function to check if the

# assignment of digits to

# characters is possible

def isSolvable(words, result):

# Stores the value

# assigned to alphabets

mp = [-1]\*(26)

# Stores if a number

# is assigned to any

# character or not

used = [0]\*(10)

# Stores the sum of position

# value of a character

# in every string

Hash = [0]\*(26)

# Stores if a character

# is at index 0 of any

# string

CharAtfront = [0]\*(26)

# Stores the string formed

# by concatenating every

# occurred character only

# once

uniq = ""

# Iterator over the array,

# words

for word in range(len(words)):

# Iterate over the string,

# word

for i in range(len(words[word])):

# Stores the character

# at ith position

ch = words[word][i]

# Update Hash[ch-'A]

Hash[ord(ch) - ord('A')] += pow(10, len(words[word]) - i - 1)

# If mp[ch-'A'] is -1

if mp[ord(ch) - ord('A')] == -1:

mp[ord(ch) - ord('A')] = 0

uniq += str(ch)

# If i is 0 and word

# length is greater

# than 1

if i == 0 and len(words[word]) > 1:

CharAtfront[ord(ch) - ord('A')] = 1

# Iterate over the string result

for i in range(len(result)):

ch = result[i]

Hash[ord(ch) - ord('A')] -= pow(10, len(result) - i - 1)

# If mp[ch-'A] is -1

if mp[ord(ch) - ord('A')] == -1:

mp[ord(ch) - ord('A')] = 0

uniq += str(ch)

# If i is 0 and length of

# result is greater than 1

if i == 0 and len(result) > 1:

CharAtfront[ord(ch) - ord('A')] = 1

mp = [-1]\*(26)

# Recursive call of the function

return True

# Auxiliary Recursive function

# to perform backtracking

def solve(words, i, S, mp, used, Hash, CharAtfront):

# If i is word.length

if i == len(words):

# Return true if S is 0

return S == 0

# Stores the character at

# index i

ch = words[i]

# Stores the mapped value

# of ch

val = mp[ord(words[i]) - ord('A')]

# If val is -1

if val != -1:

# Recursion

return solve(words, i + 1, S + val \* Hash[ord(ch) - ord('A')], mp, used, Hash, CharAtfront)

# Stores if there is any

# possible solution

x = False

# Iterate over the range

for l in range(10):

# If CharAtfront[ch-'A']

# is true and l is 0

if CharAtfront[ord(ch) - ord('A')] == 1 and l == 0:

continue

# If used[l] is true

if used[l] == 1:

continue

# Assign l to ch

mp[ord(ch) - ord('A')] = l

# Marked l as used

used[l] = 1

# Recursive function call

x |= solve(words, i + 1, S + l \* Hash[ord(ch) - ord('A')], mp, used, Hash, CharAtfront)

# Backtrack

mp[ord(ch) - ord('A')] = -1

# Unset used[l]

used[l] = 0

# Return the value of x;

return x

arr = [ "SIX", "SEVEN", "SEVEN" ]

S = "TWENTY"

# Function Call

if isSolvable(arr, S):

print("Yes")

else:

print("No")

OUTPUT:

