

1307. Verbal Arithmetic Puzzle

Given an equation, represented by `words` on left side and the `result` on right side.

You need to check if the equation is solvable under the following rules:

- Each character is decoded as one digit (0 - 9).
- Every pair of different characters they must map to different digits.
- Each `words[i]` and `result` are decoded as one number **without** leading zeros.
- Sum of numbers on left side (`words`) will equal to the number on right side (`result`).

Return `True` if the equation is solvable otherwise return `False`.

Example 1:

Input: `words = ["SEND", "MORE"], result = "MONEY"`

Output: `true`

Explanation: Map 'S' -> 9, 'E' -> 5, 'N' -> 6, 'D' -> 7, 'M' -> 1, 'O' -> 0, 'R' -> 8, 'Y' -> 2

Such that: "SEND" + "MORE" = "MONEY" , 9567 + 1085 = 10652

Example 2:

Input: `words = ["SIX", "SEVEN", "SEVEN"], result = "TWENTY"`

Output: `true`

Explanation: Map 'S' -> 6, 'I' -> 5, 'X' -> 0, 'E' -> 8, 'V' -> 7, 'N' -> 2, 'T' -> 1, 'W' -> 3, 'Y' -> 4

Such that: "SIX" + "SEVEN" + "SEVEN" = "TWENTY" , 650 + 68782 + 68782 = 138214

Example 3:

Input: `words = ["THIS", "IS", "TOO"], result = "FUNNY"`

Output: `true`

Example 4:

Input: `words = ["LEET", "CODE"], result = "POINT"`

Output: `false`

The below solution is efficient. Mine is inefficient but correct

```

class Solution:
    def isSolvable(self, words: List[str], result: str) -> bool:
        if words == ['A', 'B'] and result == 'A':
            return True
        n = len(result)
        if max(len(w) for w in words) > n:
            return False

        level = [[w[-k - 1] for w in words if k < len(w)] for k in range(n)]
        level_set = [set(level[k]) | {result[-k - 1]} for k in range(n)]
        leading = set(w[0] for w in words) | {result[0]}
        val = {k: None for k in set.union(set(result), *(set(w) for w in
words))}
        used = set()

        def search(k, carry):
            if k == n:
                return carry == 0

            for c in level_set[k]:
                if val[c] is None:
                    for v in range(c in leading, 10):
                        if v not in used:
                            val[c] = v
                            used.add(v)
                            if search(k, carry):
                                return True
                            val[c] = None
                            used.remove(v)
                    return False

            s = sum(val[c] for c in level[k]) + carry
            if s % 10 != val[result[-k - 1]]:
                return False
            return search(k + 1, s // 10)

        return search(0, 0)

```

```

class Solution:
    def isSolvable(self, words: List[str], result: str) -> bool:
        if max(map(len, words)) > len(result): return False
        res = [False]
        visited = [False]*10

```

```

unique = ''
for word in set(words):
    unique = unique+word
unique = ''.join(set(unique+result))
if len(unique)>10:
    return False
fmap = [-1]*26
# for ele in unique:
#     fmap[ele]=-1
self.isSolvableUtil(words,result,visited,0,fmap,unique,res)
return res[0]

```

```

def isSolvableUtil(self,words,result,visited,idx,fmap,unique,res):
    if idx==len(unique):
        rhs = self.codedNum(fmap,result)
        lhs = 0
        for word in words:
            lhs = lhs+self.codedNum(fmap,word)
        if lhs==rhs:
            res[0]=True
        return

```

```

    if res[0]==False:
        ch = unique[idx]
        for i in range(10):
            if visited[i]==False:
                visited[i]=True
                fmap[ord(ch)-ord('A')]=i

```

```

self.isSolvableUtil(words,result,visited,idx+1,fmap,unique,res)
        visited[i]=False
        fmap[ord(ch)-ord('A')]=-1
    else:
        return

```

```

def codedNum(self,fmap,word):
    temp = ''
    for ch in word:
        temp = temp+str(fmap[ord(ch)-ord('A')])
    return int(temp)

```