```
ums[0]

, current_sum + num)
, current_sum)


1, -5, 4]
```

6

```
                                                    [[1, 1, 2], [1, 2, 1], [2, 1, 1]]
  :


ms)):
[i] == nums[i - 1]:

] + nums[i + 1:], path + [nums[i]])
```

```
s, target):

, target):


    len(candidates)):
candidates[i] == candidates[i - 1]:

path + [candidates[i]], target - candidates[i])


  1, 5]

tes, target))
```

[[1, 1, 6], [1, 2, 5], [1, 7], [2, 6]]

```
s, target):


, target):




 len(candidates)):
candidates[i] == candidates[i - 1]:

path + [candidates[i]], target - candidates[i])




 1, 5]

tes, target))
```

[[1, 1, 6], [1, 2, 5], [1, 7], [2, 6]]

```
:

dSay(n - 1)


(1, len(prev)):
 == prev[i - 1]:
+= 1


 += str(count) + prev[i - 1]
= 1
count) + prev[-1]



4))
```

1211

```
['5', '3', '4', '6', '7', '8', '9', '1', '2']
['6', '7', '2', '1', '9', '5', '3', '4', '8']
['1', '9', '8', '3', '4', '2', '5', '6', '7']
['8', '5', '9', '7', '6', '1', '4', '2', '3']
['4', '2', '6', '8', '5', '3', '7', '9', '1']
['7', '1', '3', '9', '2', '4', '8', '5', '6']
['9', '6', '1', '5', '3', '7', '2', '8', '4']
['2', '8', '7', '4', '1', '9', '6', '3', '5']
['3', '4', '5', '2', '8', '6', '1', '7', '9']
```

```
[i][y] for i in range(9)) and \
[x][j] for j in range(9)) and \
[x // 3 * 3 + i][y // 3 * 3 + j] for i in range(3) for j in
```

```
:
] == '.':
 '123456789':
Valid(i, j, c):
oard[i][j] = c
f solve():
    return True
oard[i][j] = '.'
lse
```

```
".", ".", ".", "."],
"5", ".", ".", "."],
".", ".", "6", "."],
".", ".", ".", "3"],
"3", ".", ".", "1"],
".", ".", ".", "6"],
".", "2", "8", "."],
"9", ".", ".", "5"],
".", ".", "7", "9"]
```

```
or seen.add(x) for i, row in enumerate(board)
n enumerate(row) if c != '.' if not seen.add(
add((i, c)) and not seen.add((i // 3, j // 3, c)))
```

```
".", ".", ".", "."],
"5", ".", ".", "."],
".", ".", "6", "."],
".", ".", ".", "3"],
"3", ".", ".", "1"],
".", ".", ".", "6"],
".", "2", "8", "."],
"9", ".", ".", "5"],
".", ".", "7", "9"]
```

```
s if x != val]



))
```

2