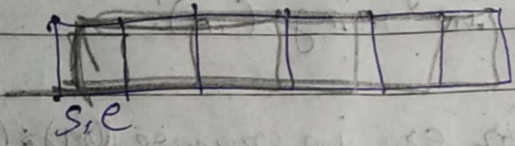
560) Subarray Sum equals K.

Input: [1, 1, 1] , K = 2.   → {1, 1} ⇒ 1+1=2
                              {1, 1} ⇒ 1+1=2
Output: 2

Input: [1, -1, 0] , K = 0
Output: 3.   ⇒ [1, -1, 0] = 0
                [1, -1] = 0
                [0] = 0.

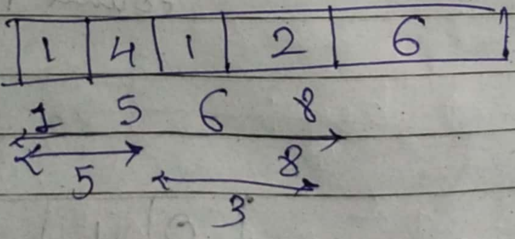
* Brute force:

(1) Find all possible subarrays.
(2) Find sum & check == target.

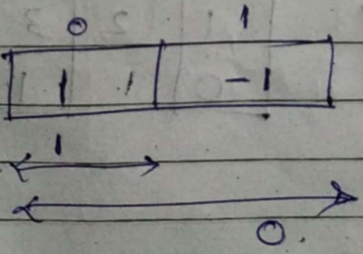


```
for s in range(n):
    for e in range(s, n):
        sum = 0
        for i in range(s, e+1):
            sum += num[i]
        if sum == k
            count ++
```

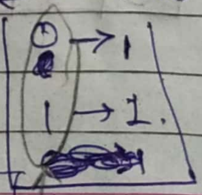
* Optimal:

eg. 

⟶ store previous sum in map ⇒ {} with their count.

eg. 

, K = 0. ⟸ this Base case.



⟶ Initialize with $\boxed{d = \{0 : 1\}}$. (cumulative Sum.

* Prefix-Sum technique / Cumulative Sum

① → subarray sum equals k
② → make sum divisible by p
③ → continuous subarray sum
④ → contiguous array
⑤ → maximum size subarray sum equals k

①
```
ans = 0
prefix-sum = 0
d = {0:1}  ──→  if k-k = 0

for num in nums:                    ⇒ nums = [1, 2, 3]
    prefix_sum += num
    key = prefix-sum - K            ⇒ K = 3
                                      K = prefix-sum - key
    if key in d:                      K = 1 - (-2) = 3 ✓
        ans += d[key]

    d[prefix-sum] = d.get(prefix-sum, 0) + 1
```

```
    d = {0:1}                           ③ prefix-sum = 6
①  prefix-sum = 1                         key = 6 - 3 = 3
    key = 1 - 3 = -2                       if key in d :→ ✓
    key in d → ✗                               ans += d[3]
    d = {0:1, 1:1}                             ans += 1 ⇒ [ans=2]
           ↑    ↑
                                              ↓
②  prefix sum = 3                      arr:    [1, 2, 3]
    key = 3 - 3 = 0                     prefix
  if key in d :→ ✓                      sum:   [1, 3, 6]
        ans += d[0] ⇒ [ans=1]
  d = {0:1, 1:1, 3:1}
```

1074) Number of Submatrices Thats Sum to Target :

Date
Page

$$\text{e.g.}\quad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 0 & 1 & 0 \end{array}, \quad \text{target} = 2$$

output: 4

sr = start row
sc = start column
er = end row
ec = end column

* Brute Force :

same for previous but this is 2D matrix

```
for sr in range(n):
    for sc in range(m):

        for er in range(sr, n)
            for ec in range(sc, m)

                sum = 0
                for i in range (sr, er+1):
                    for j in range (sc, ec+1):
                        sum += mat[i][j]

                if sum == target:
                    count ++
```

* Optimal :

$$\begin{array}{c|ccc} & \text{sc} & & \\ \hline \text{sr} & 0 & 1 & 0 \\ & 1 & 1 & 1 \\ & 0 & 1 & 0 \end{array} \Longrightarrow \begin{array}{ccc} & \overset{\text{sum}}{3} & \\ 0 & 1 & 1 \\ 1 & 2 & 3 \\ 0 & 1 & 1 \\ & 4 & \end{array} \Bigg\} \text{column wise sum}$$

d[CumSum] = d.get(CumSum, 0)+1

Date ____
Page ____

* **Optimal :**

$$\begin{array}{c|ccc} & \overset{sc}{\downarrow j} & & \\ \text{row} & 0 & 1 & 1 \\ \text{row} & 1 & 2 & 3 \\ \text{row} & 0 & 1 & 1 \end{array}$$, K=2

initlize

0	1

map

0	~~1~~2
1	~~1~~2

cumSum += matrix[row][j]

cumSum - K ⇒ 0 - 2 = -2 ×

⇒ 1 - 2 = -1 ×

⇒ 1 - 2 = -1 ×

re-initlize map.

0	1

map

0	1
1	1
3	1
4	1

$$\begin{array}{c|ccc} & 0 & \overset{sc}{\underset{j}{1}} & 2 \\ \hline \text{~~row~~}\ 0 & 0 & 1 & 1 \\ \text{~~row~~}\ 1 & 1 & 2 & 3 \\ \text{row}\ 2 & 0 & 1 & 1 \end{array}$$

CumSum += matrix[row][j]

cumSum - K ⇒ 1-2 = -1 ×

⇒ (1+2)-2 = 1 ✓

result = 1

⇒ 4-2 = 2 ×

$$\begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \begin{matrix} \}1 \\ \}2 \end{matrix} 3 \Rightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1 & 2 \\ \hline \end{array} \leftarrow 3$$

map

0	1
1	1
4	1
5	1

$$\begin{array}{c|ccc} & 0 & 1 & \overset{sc}{\underset{j}{2}} \\ \hline \text{~~row~~}\ 0 & 0 & 1 & 1 \\ 1 & 1 & 2 & 3 \\ 2 & 0 & 1 & 1 \end{array}$$

CumSum += matrix[row][j]

CumSum - K ⇒ 1 - 2 = -1 ×

⇒ 4 - 2 = 2 ×

⇒ 5 - 2 = 3 ×

map

0	1
1	1
2	1

$$\begin{array}{c|ccc} & 0 & \overset{se}{\overset{\downarrow}{j}}\ 1 & 2 \\ \hline row\ 0 & 0 & 1 & 1 \\ 1 & 1 & 2 & 3 \\ 2 & 0 & 1 & 1 \end{array}$$

CumSum += mat[row][j] ~~- mat[row][j-1]~~

CumSum-K ⇒ 1 - 2 - 0 = -1 ✗

Yes '0' in map ⇒ [(1+2) - 1] - 2 = 0

result = 1

Yes '1' in map ⇒ [(1+2+1) - 1] - 2 = 1

result = $\boxed{2}$

map

0	1
1	1
3	1

$$\begin{array}{c|ccc} & 0 & 1 & \overset{se}{\overset{\downarrow}{j}}\ 2 \\ \hline row\ 0 & 0 & 1 & 1 \\ 1 & 1 & 2 & 3 \\ 2 & 0 & 1 & 1 \end{array}$$

((1+3)-2) - k

CumSum += mat[row][j]
- mat[row][j-1]

CumSum - K = [(1 - 1) - 2] = -2 ✗

Yes '0' in map = [(1+3) - 2] - 2 = 0 ✓

result = $\boxed{1}$

= [(1+3+1) - 1] - 2 = 2 ✗

$\boxed{\text{result} = 4}$

Date ________
Page ________

```
def numSubmatrixSum Target( matrix , target):
        rows = len(matrix)
        ~~cols~~ = len (matrix [0])

        # first take the cumulative sum row-wise
        for r in range(rows):
            for c in range(1, cols):
                matrix[r][c] += matrix[r][c-1]

        # Now, you need to find the 'No. of
        # Subarrays with sum k" in
        # downward direction.

        result =0
        for startCol in range (cols):
            for currCol in range (starCol, cols):
                # find all sub matrices sum
                # concept of 'No. of subarrays ---"
                mp = {0: 1}
                cumSum =0
                # Go downwards row wise
                for r in range (rows):
                    cumSum = matrix[r][currCol]
                     - (matrix [r][starCol -1]
                       if startCol >0 else 0)

                    if cumSum - target in mp:
                        result += mp [cumSum -
                                     target]
                    mp [cumSum] = mp.get(cumSum,0)
    return result.                                    +1
```