2-Pointers

Given an array A and two indexes ‘i’ and ‘j’. you have to tell the sum of numbers from ‘i’ to ‘j’, ex:

[1, 3, -4, 5, 6, 1, 2]
0 1 2 3 4 5 6

for i = 0, j = 2, sub-array = [1, 3, -4] and sum = 0
for i = 4, j = 5, sub-array = [6, 1] and sum = 0

k queries:

1) 0 to 2 = 0
2) 0 to 5 = 12
3) 2 to 3 = 1

For each query, what is the time complexity? ➔ k * O (n). if k value is equal to n, then the Time Complexity would be $O(n^2)$. How to do it in lesser Time Complexity?

If you have any array, that gives some of all the elements i.e., the running-sum,

0	1	2	3	4	5	6	
[1	5	-1	0	4	8	4]	➔ array
[1	6	5	5	9	17	21]	➔ running-sum

For i = 2, j = 5, sum = 11

We can understand that,

P[5] = A[0] + A[1] + A[2] + A[3] + A[4] + A[5]

P[1] = A[0] + A[1]

Any index mentioned in P[j] gives the sum of the value from 0 to j in the array.

If we remove A[0] & A[1] from P[5] we will get the sum of all the numbers from index 2 to index 5. So, for the case,

i = 2, j = 5, sum = P[5] – P[1]

We can say for a given i and j, sum of element from i to j would be,

$$sum\ (i, j) = P[j] - P[i - 1]$$

CODE:

```
def do_precomputation(A):
    running_sum = list()
    running_sum.append(A[0])
    for i in range (1, len(A)):
        running_sum.append(running_sum[i-1] + A[i])
    return running_sum

def solve(running_sum, i, j):
    if i == 0:
        return running_sum[j]
    return running_sum[j] - running_sum[i - 1]

if __name__ == '__main__':
    A = [1, 5, -1, 0, 4, 8, 4]
    running_sum = do_precomputation(A)

    k = int(input("Enter Number of Queries: "))
    while k > 0:
        i, j = map(int, input().split())
        if i >= len(A) or j >=len(A):
            print(-1)
        print(solve(running_sum, i, j))
        k -= 1
```

Given a string, find the largest sub-string with no repeated character.

String = ‘ADOBECODEBANC’

ODEBANC = 7
ADOBEC = 6

For ‘Apple’, length of largest sub-string is 3 (‘ple’).

Implementation:

```
for i in range(0, n):
    for j in range(i, n):
        s = s + r[i:j]
        if (not repeated(s)):
            max_len = max(max_len, len(s))
```

CODE:

```
def is_repeated(s):
    map = dict()
    for ch in s:
        if ch not in map:
            map[ch] = 1
        else:
            return True
    return False

    # for i in range(len(s)):
    #     if s[i] in (s[0:i] + s[i+1:1]):
    #         return True
    # return False

def solve(str):
    n = len(str)
    max_len = 0
    for i in range(n):
        for j in range(i, n):
            s = str[i:j]
            if not is_repeated(s):
                max_len = max(max_len, len(s))
    return max_len
```

```
if __name__ == "__main__":
    print(solve('ADOBECODEBANC'))
```

OUTPUT:

https://algodaily.com/lessons/using-the-two-pointer-technique

https://www.pluralsight.com/guides/algorithm-templates:-two-pointers-part-1

https://leetcode.com/articles/two-pointer-technique/

https://longwayjade.wordpress.com/2015/04/30/leetcode-sort-two-pointer-two-sum/