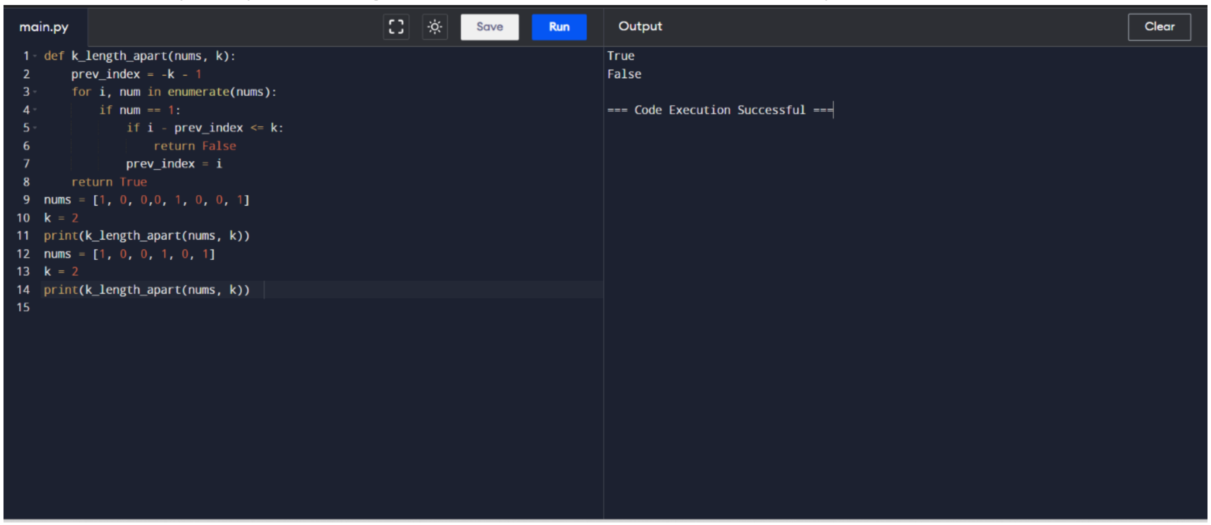
1. Given an binary arrayof an integer k return true if or once or at last k places..

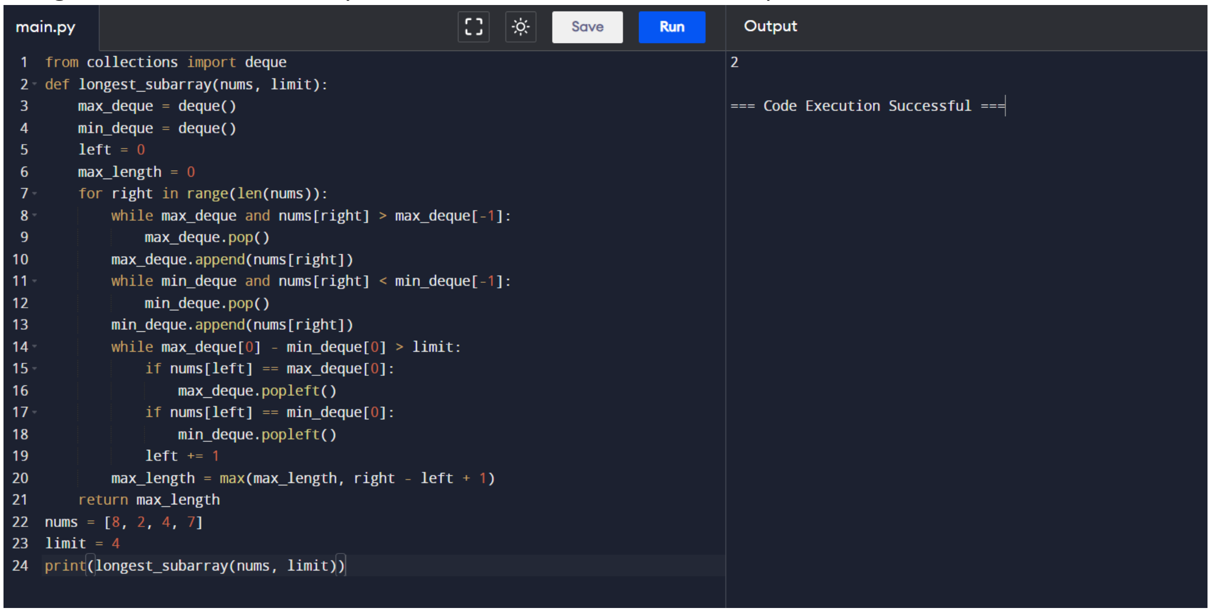


```python
def k_length_apart(nums, k):
    prev_index = -k - 1
    for i, num in enumerate(nums):
        if num == 1:
            if i - prev_index <= k:
                return False
            prev_index = i
    return True
nums = [1, 0, 0,0, 1, 0, 0, 1]
k = 2
print(k_length_apart(nums, k))
nums = [1, 0, 0, 1, 0, 1]
k = 2
print(k_length_apart(nums, k))
```

```
True
False

=== Code Execution Successful ===
```

2. Longest Continuous Subarray With Absolute Diff Less Than or Equal to Limit

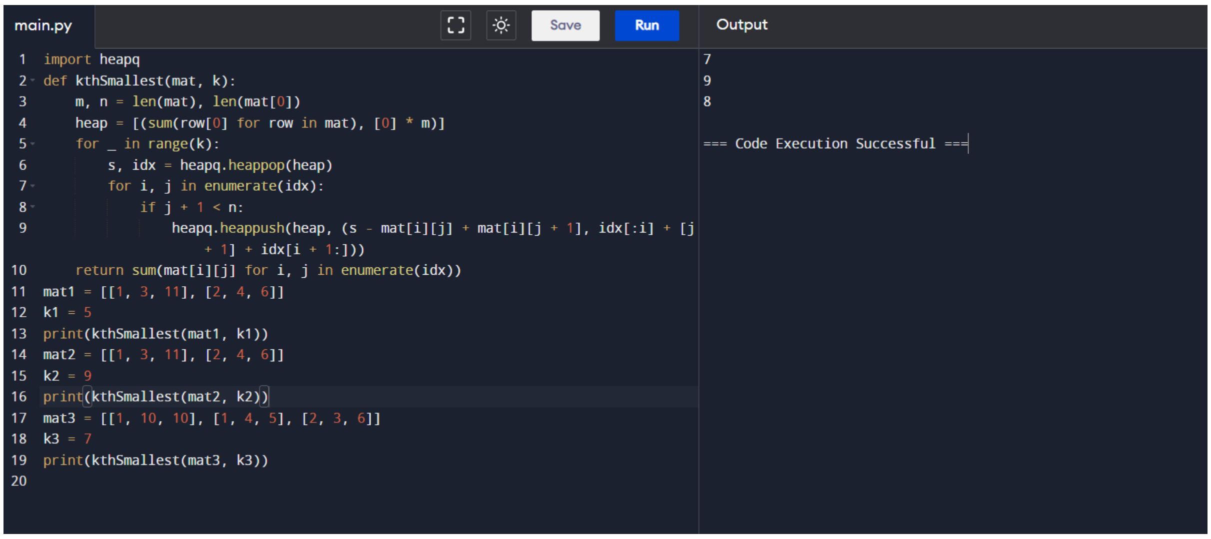


```python
from collections import deque
def longest_subarray(nums, limit):
    max_deque = deque()
    min_deque = deque()
    left = 0
    max_length = 0
    for right in range(len(nums)):
        while max_deque and nums[right] > max_deque[-1]:
            max_deque.pop()
        max_deque.append(nums[right])
        while min_deque and nums[right] < min_deque[-1]:
            min_deque.pop()
        min_deque.append(nums[right])
        while max_deque[0] - min_deque[0] > limit:
            if nums[left] == max_deque[0]:
                max_deque.popleft()
            if nums[left] == min_deque[0]:
                min_deque.popleft()
            left += 1
        max_length = max(max_length, right - left + 1)
    return max_length
nums = [8, 2, 4, 7]
limit = 4
print(longest_subarray(nums, limit))
```

```
2

=== Code Execution Successful ===
```

3. Find the Kth Smallest Sum of a Matrix With Sorted Rows

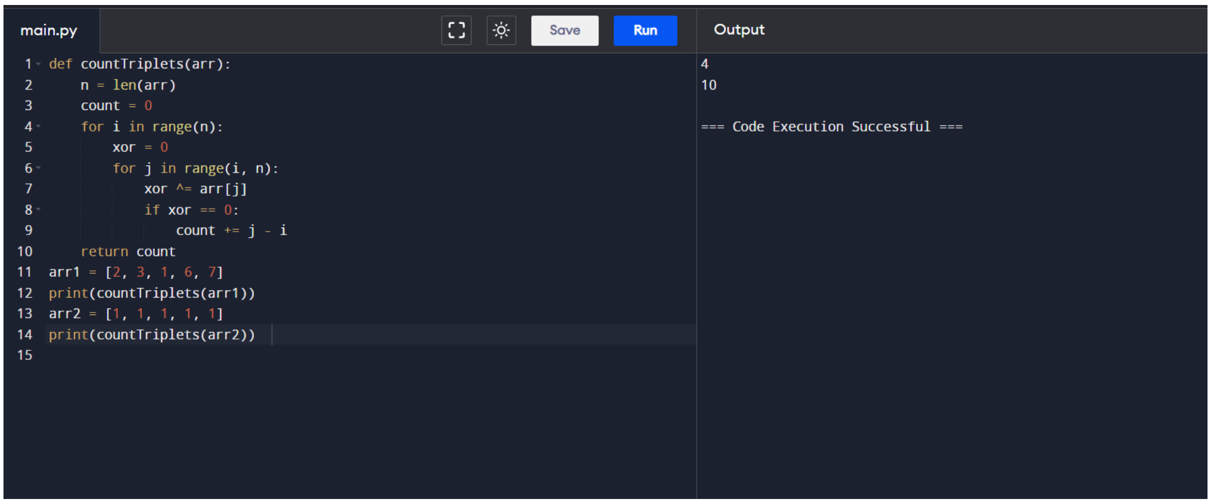


```python
import heapq
def kthSmallest(mat, k):
    m, n = len(mat), len(mat[0])
    heap = [(sum(row[0] for row in mat), [0] * m)]
    for _ in range(k):
        s, idx = heapq.heappop(heap)
        for i, j in enumerate(idx):
            if j + 1 < n:
                heapq.heappush(heap, (s - mat[i][j] + mat[i][j + 1], idx[:i] + [j
                    + 1] + idx[i + 1:]))
    return sum(mat[i][j] for i, j in enumerate(idx))
mat1 = [[1, 3, 11], [2, 4, 6]]
k1 = 5
print(kthSmallest(mat1, k1))
mat2 = [[1, 3, 11], [2, 4, 6]]
k2 = 9
print(kthSmallest(mat2, k2))
mat3 = [[1, 10, 10], [1, 4, 5], [2, 3, 6]]
k3 = 7
print(kthSmallest(mat3, k3))
```

```
7
9
8

=== Code Execution Successful ===
```

4. Count Triplets That Can Form Two Arrays of Equal XOR



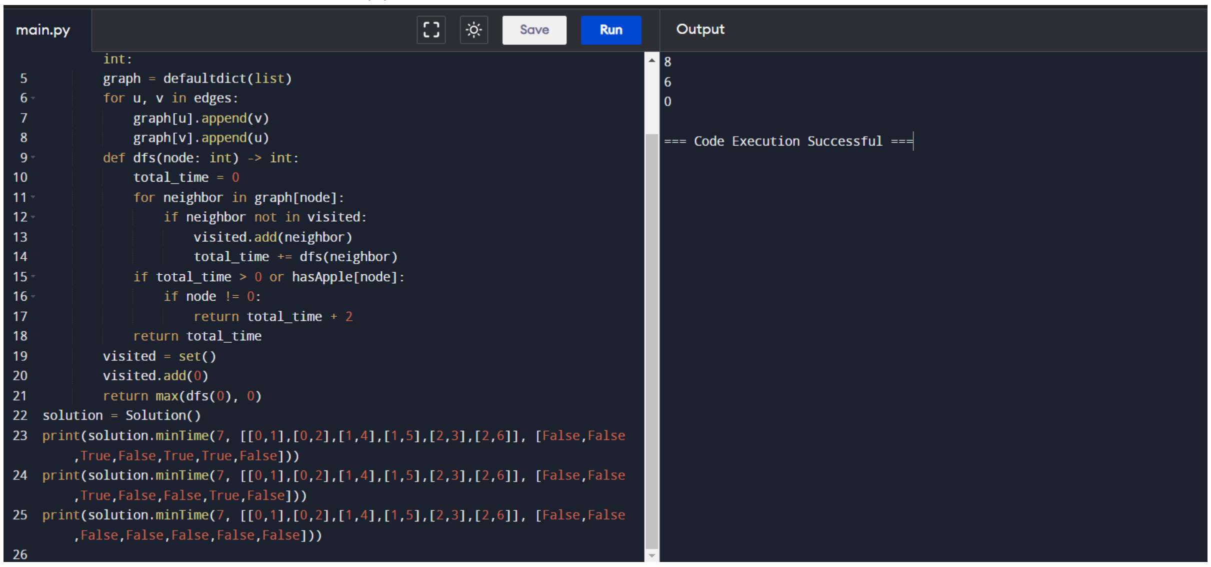
```python
def countTriplets(arr):
    n = len(arr)
    count = 0
    for i in range(n):
        xor = 0
        for j in range(i, n):
            xor ^= arr[j]
            if xor == 0:
                count += j - i
    return count
arr1 = [2, 3, 1, 6, 7]
print(countTriplets(arr1))
arr2 = [1, 1, 1, 1, 1]
print(countTriplets(arr2))

```

```
4
10

=== Code Execution Successful ===
```

5. Minimum Time to Collect All Apples in a Tree



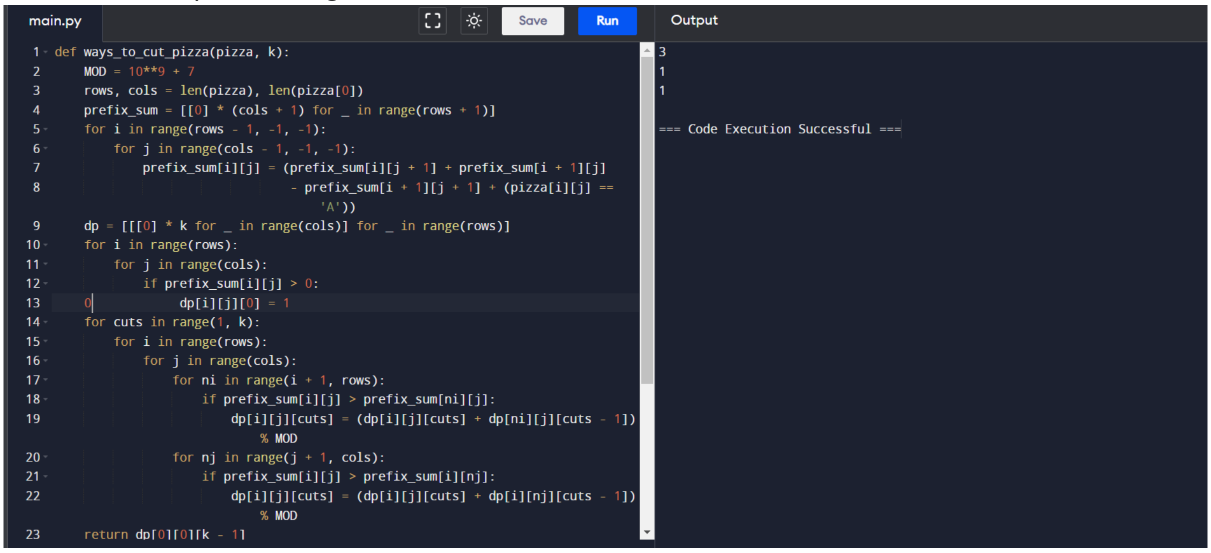
```python
        int:
        graph = defaultdict(list)
        for u, v in edges:
            graph[u].append(v)
            graph[v].append(u)
        def dfs(node: int) -> int:
            total_time = 0
            for neighbor in graph[node]:
                if neighbor not in visited:
                    visited.add(neighbor)
                    total_time += dfs(neighbor)
            if total_time > 0 or hasApple[node]:
                if node != 0:
                    return total_time + 2
            return total_time
        visited = set()
        visited.add(0)
        return max(dfs(0), 0)
solution = Solution()
print(solution.minTime(7, [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], [False,False
    ,True,False,True,True,False]))
print(solution.minTime(7, [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], [False,False
    ,True,False,False,True,False]))
print(solution.minTime(7, [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], [False,False
    ,False,False,False,False,False]))

```

```
8
6
0

=== Code Execution Successful ===
```

6. . Number of Ways of Cutting a Pizza



```python
def ways_to_cut_pizza(pizza, k):
    MOD = 10**9 + 7
    rows, cols = len(pizza), len(pizza[0])
    prefix_sum = [[0] * (cols + 1) for _ in range(rows + 1)]
    for i in range(rows - 1, -1, -1):
        for j in range(cols - 1, -1, -1):
            prefix_sum[i][j] = (prefix_sum[i][j + 1] + prefix_sum[i + 1][j]
                                - prefix_sum[i + 1][j + 1] + (pizza[i][j] ==
                                  'A'))
    dp = [[[0] * k for _ in range(cols)] for _ in range(rows)]
    for i in range(rows):
        for j in range(cols):
            if prefix_sum[i][j] > 0:
    0            dp[i][j][0] = 1
    for cuts in range(1, k):
        for i in range(rows):
            for j in range(cols):
                for ni in range(i + 1, rows):
                    if prefix_sum[i][j] > prefix_sum[ni][j]:
                        dp[i][j][cuts] = (dp[i][j][cuts] + dp[ni][j][cuts - 1])
                            % MOD
                for nj in range(j + 1, cols):
                    if prefix_sum[i][j] > prefix_sum[i][nj]:
                        dp[i][j][cuts] = (dp[i][j][cuts] + dp[i][nj][cuts - 1])
                            % MOD
    return dp[0][0][k - 1]
```

```
3
1
1

=== Code Execution Successful ===
```