Searching in an array where adjacent differ by at most k

A step array is an array of integers where each element has a difference of at most k with its neighbor. Given a key x, we need to find the index value of x if multiple-element exist to return the first occurrence of the key.

Input : arr[] = {4, 5, 6, 7, 6}
k = 1
x = 6
Output : 2
The first index of 6 is 2.

Input : arr[] = {20, 40, 50, 70, 70, 60}
k = 20
x = 60
Output : 5
The index of 60 is 5
This problem is mainly an extension of
Search an element in an array where difference between adjacent elements is 1.
A **Simple Approach** is to traverse the given array one by one and compare every element with the given element 'x'.
If matches, then return index.
The above solution can be **Optimized** using
the fact that the difference between all adjacent elements is at most k. The idea is to start comparing from the leftmost element and find the difference between the current array element and x. Let this difference be 'diff'. From the given property of the array, we always know that x must be at least 'diff/k' away, so instead of searching one by one, we jump 'diff/k'.
Below is the implementation of the above idea.

```
def search(arr, n, x, k):

    # Traverse the given array starting from
    # leftmost element
    i = 0
    while (i < n):

        # If x is found at index i
```

```
        if (arr[i] == x):
            return i

        # Jump the difference between current
        # array element and x divided by k
        # We use max here to make sure that i
        # moves at-least one step ahead.
        i = i + max(1, int(abs(arr[i] - x) / k))


    print("number is not present!")
    return -1


# Driver program to test above function
arr = [2, 4, 5, 7, 7, 6]
x = 6
k = 2
n = len(arr)
print("Element", x, "is present at index",search(arr, n, x, k))

# This code is contributed
# by Smitha Dinesh Semwal
```