1) **Sequential Search**

def findKthPositive(arr, k):

missing\_count = 0

current = 1

index = 0

while missing\_count < k:

if index < len(arr) and arr[index] == current:

index += 1

else:

missing\_count += 1

if missing\_count == k:

return current

current += 1

print(findKthPositive([2, 3, 4, 7, 11], 5))

print(findKthPositive([1, 2, 3, 4], 2))

Input: arr = [2,3,4,7,11], k = 5

Output: 9

Input: arr = [1,2,3,4], k = 2

Output: 6

2) **Sequential Search(An algorithm that runs in O(log n) time)**

def find\_peak(nums):

n = len(nums)

if n == 0:

return -1

if n == 1:

return 0

if nums[0] > nums[1]:

return 0

if nums[n - 1] > nums[n - 2]:

return n - 1

left, right = 1, n - 2

while left <= right:

mid = (left + right) // 2

if nums[mid] > nums[mid - 1] and nums[mid] > nums[mid + 1]:

return mid

elif nums[mid] < nums[mid + 1]:

left = mid + 1

else:

right = mid - 1

return -1

print(find\_peak([1, 2, 3, 1]))

print(find\_peak([1, 2, 1, 3, 5, 6, 4]))

Input: nums = [1,2,3,1]

Output: 2

Input: nums = [1,2,1,3,5,6,4]

Output: 5

3) **Brute-Force String Matching**

def brute\_force\_match(haystack: str, needle: str) -> int:

if not needle:

return 0

haystack\_length = len(haystack)

needle\_length = len(needle)

for i in range(haystack\_length - needle\_length + 1):

if haystack[i:i + needle\_length] == needle:

return i

return -1

print(brute\_force\_match("sadbutsad", "sad"))

print(brute\_force\_match("leetcode", "leeto"))

Input: haystack = "sadbutsad", needle = "sad"

Output: 0

Input: haystack = "leetcode", needle = "leeto"

Output: -1

4) **Brute-Force String Matching(**Given an array of string words, return all strings in words that is a substring of another word. You can return the answer in any order.A substring is a contiguous sequence of characters within a string**)**

def find\_substrings(words):

result = []

for i in range(len(words)):

for j in range(len(words)):

if i != j and words[i] in words[j]:

result.append(words[i])

break

return list(set(result))

words1 = ["mass", "as", "hero", "superhero"]

print(find\_substrings(words1)) # Output: ["as", "hero"]

words2 = ["leetcode", "et", "code"]

print(find\_substrings(words2)) # Output: ["et", "code"]

words3 = ["blue", "green", "bu"]

print(find\_substrings(words3))

Input: words = ["mass","as","hero","superhero"]

Output: ["as","hero"]

Input: words = ["leetcode","et","code"]

Output: ["et","code"]

Input: words = ["blue","green","bu"]

Output: []

5) **Word Break Problem**

def word\_break\_dp(s, wordDict):

word\_set = set(wordDict)

dp = [False] \* (len(s) + 1)

dp[0] = True

for i in range(1, len(s) + 1):

for j in range(i):

if dp[j] and s[j:i] in word\_set:

dp[i] = True

break

return dp[len(s)]

print(word\_break\_dp("leetcode", ["leet", "code"]))

print(word\_break\_dp("applepenapple", ["apple", "pen"]))

print(word\_break\_dp("catsandog", ["cats", "dog", "sand", "and", "cat"]))

Input: s = "leetcode", wordDict = ["leet","code"]

Output: true

Input: s = "applepenapple", wordDict = ["apple","pen"]

Output: true

Input: s = "catsandog", wordDict = ["cats","dog","sand","and","cat"]

Output: false