1055. Shortest Way to Form String
We will store source’s letter and its associated index store in our memo. Then each time we will use binary search get word’s index and find next index based on last index.
TC is O(nlogm + m)

```
from collections import defaultdict
from bisect import *
class Solution:
    def shortestWay(self, source: str, target: str) -> int:
        memo = defaultdict(list)
        prev = -1
        res = 1
        for i, e in enumerate(source):
            memo[e].append(i)
        for i in target:
            if memo[i]:
                if prev > memo[i][-1]:
                    res += 1
                    prev = -1
                idx = bisect_left(memo[i], prev)
                prev = memo[i][idx] + 1
            else:
                return -1
        return res
```

939. Minimum Area Rectangle

We will iterate all possible all points and calculate all rectangles. In the end, get the minimum one. TC is O(n**2)

```
from collections import defaultdict
class Solution:
    def minAreaRect(self, points: List[List[int]]) -> int:
        memo = defaultdict(set)
        min_area = float('inf')
        for x, y in points:
            memo[x].add(y)
        for x1, y1 in points:
            for x2, y2 in points:
                if x1 == x2 or y1 == y2:
                    continue
                if y2 in memo[x1] and y1 in memo[x2]:
                    min_area = min(min_area, abs(y1 - y2) * abs(x1 - x2))
        return 0 if min_area == float('inf') else min_area
```

947. Most Stones Removed with Same Row or Column
We will union find to union all points with rows or columns are same. And decrease point by 1. TC is O(n*n)

```
class Solution:
    def removeStones(self, stones: List[List[int]]) -> int:
        self.count = len(stones)
        parents = {}
        def find(point):
            p = point
            while point in parents and parents[point] != point:
                point = parents[point]
            parents[p] = point
            return point

        def union(point1, point2):
            p1 = find(point1)
            p2 = find(point2)
            if p1 != p2:
                self.count -= 1
                parents[p1] = p2

        for x1, y1 in stones:
            for x2, y2 in stones:
```

```
            if x1 == x2 or y1 == y2:
                union((x1, y1), (x2, y2))
        return len(stones) - self.count
```

809. Expressive Words

We will compare S and word one letter by one. If they are equal, we will add both index by 1, if not, we will check it’s a more than 3 letters in S, we will check whether two index go to the final step. TC is O(m * n * l)

```
class Solution:
    def expressiveWords(self, S: str, words: List[str]) -> int:
        result = len(words)
        for word in words:
            i, j, m, n = 0, 0, len(S), len(word)
            mark = False
            for i in range(m):
                if j < n and S[i] == word[j]:
                    j += 1
                else:
                    if S[i - 1:i + 2] == S[i] * 3 or S[i - 2:i + 1] == S[i] * 3:
                        continue
                    else:
                        result -= 1
                        mark = True
                        break
            if j != n and not mark:
                result -= 1
        return result
```

271. Encode and Decode Strings

We will iterate through every word and use len/word format to encode all words, also we will decode it in the same format. TC is O(n)