139. Word Break

We will use dp to traverse all previous substring whether there is possibility the point could be reached. Then we will set current point to True. TC is O(n * n)

```
class Solution:
    def wordBreak(self, s: str, wordDict: List[str]) -> bool:
        dp = [False] * (len(s) + 1)
        dp[0] = True
        wordDictSet = set(wordDict)
        for i in range(1, len(s) + 1):
            for j in range(i):
                if dp[j] and s[j:i] in wordDictSet:
                    dp[i] = True
                    break

        return dp[-1]
```

2. Random pick users

We will always pick an element from a team of that elements length is the maximum. Then we will pick any element from others. So we could confirm we could create the most pairs.

```
from random import *
from collections import defaultdict
def getRandomUsers(users, counter):
 print(counter)
 if len(counter.keys()) < 2:
   return False
 max_team = max(map(lambda a: (len(a[1]), a[0]), counter.items()))[1]
 another_team = max_team
 while another_team == max_team:
   another_team = choice(counter.keys())
 user1 = choice(counter[max_team])
 counter[max_team].remove(user1)
 if not counter[max_team]:
   del counter[max_team]
 user2 = choice(counter[another_team])
 counter[another_team].remove(user2)
 if not counter[another_team]:
   del counter[another_team]
 return user1, user2

def matchUsers(users):
```

```
 result = []
 counter = defaultdict(list)
 for user in users:
   counter[user['team']].append(user['id'])

 for _ in range(len(users) // 2):
   pair_users = getRandomUsers(users, counter)
   if pair_users:
     result.append(pair_users)
   else:
     break
 return result

print(matchUsers(people1))
```

386. Lexicographical Numbers

We will take advantage of list and set key as str to sort it in lexicographical order. TC is O(nlogn), SC is O(1)

```
class Solution:
    def lexicalOrder(self, n: int) -> List[int]:
        return sorted(range(1, n + 1), key=str)
```

984. String Without AAA or BBB

When A < B, we will swap A and B. When A is larger than 2 * B, we will return aab and append other as. Or we will return (A-B) aab and append (a + b) * (B * 2 - A). TC is O(1)

```
class Solution:
    def strWithout3a3b(self, A, B, a = "a", b = "b"):
        if B > A:
            return self.strWithout3a3b(B, A, b, a)
        if A >= B * 2:
            return (a + a + b) * B + a * (A - B * 2)
        return (a + a + b) * (A - B) + (a + b) * (B * 2 - A)
```

387. First Unique Character in a String

We will count all element's number. Then we iterate through all characters in string. If its number is 1, we will return index. Or in the end, we will return -1.

```
from collections import Counter
class Solution:
    def firstUniqChar(self, s: str) -> int:
        memo = Counter(s)
        for i, c in enumerate(s):
            if memo[c] == 1:
```

```
        return i
return -1
```