Python算法模板

代码来自*B站up主*：灵茶山艾府

***GitHub*主页：**

https://github.com/liupengsay

GitHub源代码主页：

https://github.com/liupengsay/PyIsTheBestLang

CodeForces主页：

https://codeforces.com/profile/liupengsay

洛谷主页：

https://www.luogu.com.cn/user/739032

~~（由于24年5月25，26号就是昆明邀请赛了，所以就先只整理一下string的模板和与之相关的模板）~~

> ***版本号：***V1.0.0.24.05.10
>
> ***版本介绍：***只整理了string的template以及peoblem代码，相关的暂未整理
>
> ***与上一版变化：***~~不是第一版能有啥变化，不过好像确实就是和网上说的一样10min足够学完基本的md语法然后把这个py模板整理出来了~~

> ***简介：***string共包含8个模块，按以下顺序排列：**kmp，manacher_palindrome，palindrome_num，expression，lyndon_decomposition，string_hash，suffix_array，automaton**

> ***补充：***以下代码块中的fastio以及其他数据结构之类的模板暂时并未纳入，想先把第一版打印之后再补充，还有有些模板里面的题目不全或者太少这个坑也暂时空着。

一.string

1.kmp

1.1 template:

```
class KMP:
    def __init__(self):
        return

    @classmethod
    def prefix_function(cls, s):
        """calculate the longest common true prefix and true suffix for s
[:i+1] and s [:i+1]"""
        n = len(s)  # fail tree
        pi = [0] * n
        for i in range(1, n):
            j = pi[i - 1]
            while j > 0 and s[i] != s[j]:
                j = pi[j - 1]
            if s[i] == s[j]:  # all pi[i] pi[pi[i]] ... are border
                j += 1   # all i+1-pi[i] pi[i]+1-pi[pi[i]] ... are
circular_section
            pi[i] = j  # pi[i] <= i also known as next
        # pi[0] = 0
        return pi    # longest common true prefix_suffix / i+1-nex[i] is
shortest circular_section

    @staticmethod
    def z_function(s):
        """calculate the longest common prefix between s[i:] and s"""
        n = len(s)
        z = [0] * n
        left, r = 0, 0
        for i in range(1, n):
            if i <= r and z[i - left] < r - i + 1:
                z[i] = z[i - left]
            else:
                z[i] = max(0, r - i + 1)
                while i + z[i] < n and s[z[i]] == s[i + z[i]]:
                    z[i] += 1
            if i + z[i] - 1 > r:
                left = i
                r = i + z[i] - 1
        # z[0] = 0
        return z

    def prefix_function_reverse(self, s):
        n = len(s)
        nxt = [0] + self.prefix_function(s)
        nxt[1] = 0
        for i in range(2, n + 1):
            j = i
```

```
            while nxt[j]:
                j = nxt[j]
            if nxt[i]:
                nxt[i] = j
        return nxt[1:]  # shortest common true prefix_suffix / i+1-nex[i]
is longest circular_section

    def find(self, s1, s2):
        """find the index position of s2 in s1"""
        n, m = len(s1), len(s2)
        pi = self.prefix_function(s2 + "#" + s1)
        ans = []
        for i in range(m + 1, m + n + 1):
            if pi[i] == m:
                ans.append(i - m - m)
        return ans

    def find_lst(self, s1, s2, tag=-1):
        """find the index position of s2 in s1"""
        n, m = len(s1), len(s2)
        pi = self.prefix_function(s2 + [tag] + s1)
        ans = []
        for i in range(m + 1, m + n + 1):
            if pi[i] == m:
                ans.append(i - m - m)
        return ans

    def find_longest_palindrome(self, s, pos="prefix") -> int:
        """calculate the longest prefix and longest suffix palindrome
substring"""
        if pos == "prefix":
            return self.prefix_function(s + "#" + s[::-1])[-1]
        return self.prefix_function(s[::-1] + "#" + s)[-1]

    @staticmethod
    def kmp_automaton(s, m=26):
        n = len(s)
        nxt = [0] * m * (n + 1)
        j = 0
        for i in range(1, n + 1):
            j = nxt[j * m + s[i - 1]]
            nxt[(i - 1) * m + s[i - 1]] = i
            for k in range(m):
                nxt[i * m + k] = nxt[j * m + k]
        return nxt

    @classmethod
    def merge_b_from_a(cls, a, b):
        c = b + "#" + a
        f = cls.prefix_function(c)
        m = len(b)
        if max(f[m:]) == m:
            return a
        x = f[-1]
        return a + b[x:]
```

1.2problem:

```
# Algorithm：kmp|find|z-function|circular_section
# Description：string|prefix_suffix
'''
==================================LuoGu==============================
P3375（https://www.luogu.com.cn/problem/P3375）longest_prefix_suffix|fin
P4391（https://www.luogu.com.cn/problem/P4391）brain_teaser|kmp|n-pi[n-1]
P3435（https://www.luogu.com.cn/problem/P3435）
kmp|longest_circular_section|prefix_function_reverse|classical
P4824（https://www.luogu.com.cn/problem/P4824）
P2375（https://www.luogu.com.cn/problem/P2375）kmp|z-function|diff_array
P7114（https://www.luogu.com.cn/problem/P7114）
P3426（https://www.luogu.com.cn/problem/P3426）
P3193（https://www.luogu.com.cn/problem/P3193）
kmp_automaton|matrix_fast_power|matrix_dp
P4036（https://www.luogu.com.cn/problem/P4036）kmp|z-function
P5410（https://www.luogu.com.cn/problem/P5410）kmp|z-function
P1368（https://www.luogu.com.cn/problem/P1368）
P3121（https://www.luogu.com.cn/problem/P3121）
P5829（https://www.luogu.com.cn/problem/P5829）kmp|z-
function|fail_tree|classical|border|longest_common_border|tree_lca
P8112（https://www.luogu.com.cn/problem/P8112）
z_function|point_set|range_min|classical
==================================CodeForces==========================
1326D2（https://codeforces.com/problemset/problem/1326/D2）
manacher|greedy|prefix_suffix|longest_prefix_suffix|palindrome_substring
432D（https://codeforces.com/contest/432/problem/D）kmp|z-function|sorted_list
25E（https://codeforces.com/contest/25/problem/E）
kmp|prefix_suffix|greedy|longest_common_prefix_suffix
126B（https://codeforces.com/contest/126/problem/B）kmp|z-
function|classical|brute_force
471D（https://codeforces.com/contest/471/problem/D）
kmp|brain_teaser|classical|diff_array
346B（https://codeforces.com/contest/346/problem/B）kmp|lcs|matrix_dp
494B（https://codeforces.com/contest/494/problem/B）kmp|linear_dp|prefix_sum
1200E（https://codeforces.com/problemset/problem/1200/E）string_hash|kmp
615C（https://codeforces.com/contest/615/problem/C）kmp|linear_dp|specific_plan
1163D（https://codeforces.com/problemset/problem/1163/D）
kmp|matrix_dp|kmp_automaton
526D（https://codeforces.com/contest/526/problem/D）
brain_teaser|classical|kmp|circular_section
954I（https://codeforces.com/problemset/problem/954/I）
808G（https://codeforces.com/contest/808/problem/G）kmp|kmp_automaton|z-
function|matrix_dp
182D（https://codeforces.com/problemset/problem/182/D）
kmp|circular_section|num_factor
535D（https://codeforces.com/problemset/problem/535/D）kmp|z-function|union_find
1051E（https://codeforces.com/contest/1051/problem/E）kmp|z-function|linear_dp
1015F（https://codeforces.com/contest/1015/problem/F）kmp_automaton|matrix_dp
1690F（https://codeforces.com/contest/1690/problem/F）
permutation_circle|kmp|circle_section
```

```
1968G2（https://codeforces.com/contest/1968/problem/G2）
z_algorithm|offline_query|binary_search|brute_force|preprocess
================================================================================
'''
```

```python
import bisect
import math
from collections import Counter
from functools import lru_cache
from itertools import permutations
from typing import List

from src.data_structure.segment_tree.template import PointSetRangeMin
from src.data_structure.sorted_list.template import SortedList
from src.graph.tree_lca.template import TreeAncestor
from src.graph.union_find.template import UnionFind
from src.mathmatics.fast_power.template import MatrixFastPower
from src.mathmatics.number_theory.template import NumFactor
from src.strings.kmp.template import KMP
from src.utils.fast_io import FastIO, inf

class Solution:
    def __init__(self):
        return

    def lg_p3375(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P3375
        tag: longest_prefix_suffix|find
        """
        s1 = ac.read_str()
        s2 = ac.read_str()
        m, n = len(s1), len(s2)
        pi = KMP().prefix_function(s2 + "@" + s1)
        for i in range(n, m + n + 1):
            if pi[i] == n:
                ac.st(i - n + 1 - n)
        ac.lst(pi[:n])
        return

    def lg_p4391(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P4391
        tag: brain_teaser|kmp|n-pi[n-1]|classical
        """
        n = ac.read_int()
        s = ac.read_str()
        pi = KMP().prefix_function(s)
        ac.st(n - pi[-1])
        return

    def lg_p3435(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P3435
        tag: kmp|longest_circular_section|prefix_function_reverse|classical
```

```
        """
        n = ac.read_int()
        s = ac.read_str()
        nxt = KMP().prefix_function_reverse(s)
        ans = sum(i + 1 - nxt[i] for i in range(1, n) if nxt[i])
        ac.st(ans)
        return

    def lg_p2375(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P2375
        tag: kmp|z-function|diff_array
        """
        mod = 10 ** 9 + 7
        for _ in range(ac.read_int()):
            s = ac.read_str()
            n = len(s)
            z = KMP().z_function(s)
            diff = [0] * n
            for i in range(1, n):
                if z[i]:
                    x = ac.min(z[i], i)
                    diff[i] += 1
                    if i + x < n:
                        diff[i + x] -= 1
            ans = 1
            for i in range(1, n):
                diff[i] += diff[i - 1]
                ans *= (diff[i] + 1)
                ans %= mod
            ac.st(ans)
        return

    def lg_p3193(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P3193
        tag: kmp_automaton|matrix_fast_power|matrix_dp
        """
        n, m, k = ac.read_list_ints()
        lst = [int(w) for w in ac.read_str()]
        nxt = KMP().kmp_automaton(lst, 10)
        grid = [[0] * (m + 9) for _ in range(m + 9)]
        for i in range(10):
            for j in range(m):
                ind = i if j == 0 else j + 9
                for x in range(10):
                    y = nxt[j * 10 + x]
                    if y == 0:
                        grid[x][ind] = 1
                    elif y < m:
                        grid[y + 9][ind] = 1

        initial = [0] * (m + 9)
        for x in range(10):
            if x == lst[0] and m > 1:
                initial[10] = 1
```

```
            elif x != lst[0]:
                initial[x] = 1
        mat = MatrixFastPower().matrix_pow(grid, n - 1, k)
        ans = 0
        for i in range(m + 9):
            ans += sum(mat[i][j] * initial[j] for j in range(m + 9))
        ac.st(ans % k)
        return

    def lg_p4036(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P4036
        tag: kmp|z-function
        """
        lst = [ord(w) - ord("a") for w in ac.read_str()]
        for _ in range(ac.read_int()):
            cur = ac.read_list_strs()
            if cur[0] == "Q":
                x, y = [int(w) - 1 for w in cur[1:]]
                n = len(lst)
                ans = KMP().z_function(lst[x:] + [-1] + lst[y:])[n - x + 1]
                ac.st(ans)
            elif cur[0] == "R":
                x = int(cur[1]) - 1
                w = ord(cur[2]) - ord("a")
                lst[x] = w
            else:
                x = int(cur[1])
                w = ord(cur[2]) - ord("a")
                lst.insert(x, w)
        return

    def lg_p5410(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P5410
        tag: kmp|z-function
        """
        a = ac.read_str()
        b = ac.read_str()
        m, n = len(a), len(b)
        z = KMP().z_function(b + "#" + a)
        z[0] = n
        ans = 0
        for i in range(n):
            ans ^= (i + 1) * (z[i] + 1)
        ac.st(ans)

        ans = 0
        for i in range(n + 1, n + m + 1):
            ans ^= (i - n) * (z[i] + 1)
        ac.st(ans)
        return

    def lg_p5829(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P5829
```

```
        tag: kmp|z-
function|fail_tree|classical|border|longest_common_border|tree_lca
        """
        lst = [ord(w) - ord("a") for w in ac.read_str()]
        n = len(lst)
        pi = [0] + KMP().prefix_function(lst)

        edges = [[] for _ in range(n + 1)]
        for i in range(1, n + 1):
            if pi[i]:
                edges[pi[i]].append(i)
            else:
                edges[0].append(i)
        tree = TreeAncestor(edges, 0)
        for _ in range(ac.read_int()):
            p, q = ac.read_list_ints()
            ac.st(tree.get_lca(pi[p], pi[q]))
        return

    def lg_p8112(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P8112
        tag: z_function|point_set|range_min|classical
        """
        n, m = ac.read_list_ints()  # TLE
        s = ac.read_str()
        t = ac.read_str()
        z = KMP().z_function(s + "#" + t)
        tree = PointSetRangeMin(m + 1)
        tree.point_set(m, 0)
        for i in range(m - 1, -1, -1):
            s = z[i + n + 1]
            if s:
                nex = tree.range_min(i + 1, i + s) + 1
                tree.point_set(i, nex)
        ans = tree.range_min(0, 0)
        ac.st(ans if ans < inf else "Fake")
        return

    '''
===CodeForces=================================================================
    '''

    def cf_1326d2(ac=FastIO()):
        """
        url: https://codeforces.com/problemset/problem/1326/D2
        tag:
manacher|greedy|prefix_suffix|longest_prefix_suffix|palindrome_substring
        """
        for _ in range(ac.read_int()):
            s = ac.read_str()
            n = len(s)
            i, j = 0, n - 1
            while i < j:
                if s[i] == s[j]:
                    i += 1
```

```
                    j -= 1
                else:
                    break
            if i >= j:
                ac.st(s)
                continue

            a = KMP().find_longest_palindrome(s[i:j + 1])
            b = KMP().find_longest_palindrome(s[i:j + 1], "suffix")
            ans = s[:i + a] + s[j + 1:] if a > b else s[:i] + s[j - b + 1:]
            ac.st(ans)
        return

    def cf_432d(ac=FastIO()):
        """
        url: https://codeforces.com/contest/432/problem/D
        tag: kmp|z-function|sorted_list
        """
        s = ac.read_str()
        z = KMP().z_function(s)
        lst = SortedList()
        n = len(s)
        ans = []
        for i in range(1, n):
            if z[i] == n - i:
                ans.append((n - i, lst.bisect_right(i - n) + 2))
            lst.add(-z[i])
        ans.reverse()
        ans.append((n, 1))
        ac.st(len(ans))
        for ls in ans:
            ac.lst(ls)
        return

    def cf_25e(ac=FastIO()):
        """
        url: https://codeforces.com/contest/25/problem/E
        tag: kmp|prefix_suffix|greedy|longest_common_prefix_suffix
        """

        s = [ac.read_str() for _ in range(3)]

        ind = list(range(3))
        ans = sum(len(w) for w in s)
        kmp = KMP()
        for item in permutations(ind, 3):
            cur = len(kmp.merge_b_from_a(kmp.merge_b_from_a(s[item[0]],
s[item[1]]), s[item[2]]))
            if cur < ans:
                ans = cur
        ac.st(ans)
        return

    def cf_126b(ac=FastIO()):
        """
        url: https://codeforces.com/contest/126/problem/B
```

```
        tag: kmp|z-function|classical|brute_force
        """
        s = ac.read_str()
        n = len(s)
        z = KMP().z_function(s)
        pre = 0
        for i in range(1, n):
            if z[i] == n - i and pre >= z[i]:
                ac.st(s[:z[i]])
                break
            pre = ac.max(pre, z[i])
        else:
            ac.st("Just a legend")
        return

    def cf_471d(ac=FastIO()):
        """
        url: https://codeforces.com/contest/471/problem/D
        tag: kmp|brain_teaser|classical|diff_array
        """
        m, n = ac.read_list_ints()
        a = ac.read_list_ints()
        b = ac.read_list_ints()
        if n == 1:
            ac.st(m)
            return
        if m < n:
            ac.st(0)
            return
        a = [a[i + 1] - a[i] for i in range(m - 1)]
        b = [b[i + 1] - b[i] for i in range(n - 1)]
        ans = len(KMP().find_lst(a, b, -10 ** 9 - 1))
        ac.st(ans)
        return

    def cf_346b(ac=FastIO()):
        """
        url: https://codeforces.com/contest/346/problem/B
        tag: kmp|lcs|matrix_dp|specific_plan|classical
        """
        s = [ord(w) - ord("A") for w in ac.read_str()]
        t = [ord(w) - ord("A") for w in ac.read_str()]
        virus = [ord(w) - ord("A") for w in ac.read_str()]
        m, n, k = len(s), len(t), len(virus)

        nxt = [[-1] * 26 for _ in range(k)]
        kmp = KMP()
        pre = []
        for i in range(k):
            for j in range(26):
                nxt[i][j] = kmp.prefix_function(virus + [-1] + pre + [j])[-1]
            pre.append(virus[i])

        dp = [[[0] * (k + 1) for _ in range(n + 1)] for _ in range(m + 1)]
        for i in range(m - 1, -1, -1):
            for j in range(n - 1, -1, -1):
```

```
                for x in range(k):
                    a, b = dp[i + 1][j][x], dp[i][j + 1][x]
                    dp[i][j][x] = ac.max(a, b)
                    if s[i] == t[j] and nxt[x][s[i]] < k and dp[i + 1][j + 1]
[nxt[x][s[i]]] + 1 > dp[i][j][x]:
                        dp[i][j][x] = dp[i + 1][j + 1][nxt[x][s[i]]] + 1
        length = dp[0][0][0]
        if length:
            ans = []
            i, j, x = 0, 0, 0
            while len(ans) < length:
                if dp[i][j][x] == dp[i + 1][j][x]:
                    i += 1
                elif dp[i][j][x] == dp[i][j + 1][x]:
                    j += 1
                else:
                    ans.append(s[i])
                    i, j, x = i + 1, j + 1, nxt[x][s[i]]
            ac.st("".join([chr(i + ord("A")) for i in ans]))
        else:
            ac.st("0")
        return

    def cf_494b(ac=FastIO()):
        """
        url: https://codeforces.com/contest/494/problem/B
        tag: kmp|linear_dp|prefix_sum
        """
        s = ac.read_str()
        t = ac.read_str()
        m, n = len(t), len(s)
        pi = KMP().prefix_function(t + "#" + s)
        mod = 10 ** 9 + 7
        dp = [0] * (n + 1)
        pre = [0] * (n + 1)
        dp[0] = pre[0] = 1
        last = -1
        for i in range(1, n + 1):
            if pi[i + m] == m:
                last = i - m + 1
            if last != -1:
                dp[i] = dp[i - 1] + pre[last - 1]
            else:
                dp[i] = dp[i - 1]
            dp[i] %= mod
            pre[i] = (pre[i - 1] + dp[i]) % mod
        ac.st((dp[-1] - 1) % mod)
        return

    def cf_1200e(ac=FastIO()):
        """
        url: https://codeforces.com/contest/1200/problem/E
        tag: string_hash|kmp
        """
        ac.read_int()
        lst = ac.read_list_strs()
```

```
        ans = []
        kmp = KMP()
        for word in lst:
            if not ans:
                ans.extend(list(word))
            else:
                m = len(word)
                k = ac.min(len(ans), m)
                s = list(word[:k]) + ans[-k:]
                z = kmp.z_function(s)
                inter = 0
                for i in range(1, k + 1):
                    if z[-i] == i:
                        inter = i
                for j in range(inter, m):
                    ans.append(word[j])
        ac.st("".join(ans))
        return

    def cf_615c(ac=FastIO()):
        """
        url: https://codeforces.com/contest/615/problem/C
        tag: kmp|linear_dp|specific_plan
        """
        s = ac.read_str()
        t = ac.read_str()
        m, n = len(s), len(t)
        dp = [inf] * (n + 1)
        dp[0] = 0
        state = [() for _ in range(n + 1)]
        for i in range(n):
            pre = t[:i + 1][::-1]
            z_flip = KMP().z_function(pre + "#" + s)
            for j in range(i + 2, i + 2 + m):
                if z_flip[j] and dp[i + 1 - z_flip[j]] + 1 < dp[i + 1]:
                    dp[i + 1] = dp[i + 1 - z_flip[j]] + 1
                    a, b = j - i - 2, j - i - 2 + z_flip[j] - 1
                    state[i + 1] = (b, a)

            z_flip = KMP().z_function(pre + "#" + s[::-1])
            for j in range(i + 2, i + 2 + m):
                if z_flip[j] and dp[i + 1 - z_flip[j]] + 1 < dp[i + 1]:
                    dp[i + 1] = dp[i + 1 - z_flip[j]] + 1
                    a, b = j - i - 2, j - i - 2 + z_flip[j] - 1
                    state[i + 1] = (m - 1 - b, m - 1 - a)
        if dp[-1] == inf:
            ac.st(-1)
        else:
            ans = []
            x = n
            while x:
                ans.append(state[x])
                x -= abs(state[x][0] - state[x][1]) + 1
            ac.st(len(ans))
            ans.reverse()
            for ls in ans:
```

```
            ac.lst((x + 1 for x in ls))
        return

    def cf_1163d(ac=FastIO()):
        """
        url: https://codeforces.com/contest/1163/problem/D
        tag: kmp|matrix_dp|kmp_automaton
        """
        c = [ord(w) - ord("a") for w in ac.read_str()]
        s = [ord(w) - ord("a") for w in ac.read_str()]
        t = [ord(w) - ord("a") for w in ac.read_str()]
        n, m, k = len(c), len(s), len(t)

        nxt_s = KMP().kmp_automaton(s)
        nxt_t = KMP().kmp_automaton(t)

        dp = [[-inf] * (k + 1) * (m + 1) for _ in range(2)]
        dp[0][0] = 0
        for i in range(n):
            if chr(c[i] + ord("a")) == "*":
                lst = list(range(26))
            else:
                lst = [c[i]]
            for j in range(m + 1):
                for x in range(k + 1):
                    dp[(i & 1) ^ 1][j * (k + 1) + x] = -inf
            for j in range(m + 1):
                for x in range(k + 1):
                    cur = dp[i & 1][j * (k + 1) + x]
                    if cur == -inf:
                        continue
                    for w in lst:
                        tmp = cur
                        jj = nxt_s[j * 26 + w]
                        xx = nxt_t[x * 26 + w]
                        if jj == m:
                            tmp += 1
                        if xx == k:
                            tmp -= 1
                        if tmp > dp[(i & 1) ^ 1][jj * (k + 1) + xx]:
                            dp[(i & 1) ^ 1][jj * (k + 1) + xx] = tmp
        ac.st(max(dp[n & 1]))
        return

    def cf_526d(ac=FastIO()):
        """
        url: https://codeforces.com/contest/526/problem/D
        tag: brain_teaser|classical|kmp|circular_section
        """
        n, k = ac.read_list_ints()
        ans = ["0"] * n
        s = ac.read_str()
        pi = KMP().prefix_function(s)
        for i in range(n):
            c = i + 1 - pi[i]
            low = math.ceil((i + 1) / ((k + 1) * c))
```

```
            high = (i + 1) // (k * c)
            if low <= high:
                ans[i] = "1"
        ac.st("".join(ans))
        return

    def cf_808g(ac=FastIO()):
        """
        url: https://codeforces.com/contest/808/problem/G
        tag: kmp|kmp_automaton|z-function|matrix_dp
        """
        s = ac.read_str()
        t = ac.read_str()
        m, n = len(s), len(t)
        z = KMP().z_function(t)
        ind = [0]
        for i in range(1, n):
            if z[i] == n - i:
                ind.append(n - i)

        dp = [[-inf] * n for _ in range(2)]
        pre = 0
        dp[pre][0] = 0
        for w in s:
            cur = 1 - pre
            for j in range(n):
                dp[cur][j] = -inf
            dp[cur][0] = max(dp[pre])
            for j in range(n):
                if t[j] == w or w == "?":
                    if j == n - 1:
                        for x in ind:
                            dp[cur][x] = ac.max(dp[cur][x], dp[pre][j] + 1)
                    else:
                        dp[cur][j + 1] = ac.max(dp[cur][j + 1], dp[pre][j])
            pre = cur
        ac.st(max(dp[pre]))
        return

    def cf_182d(ac=FastIO()):
        """
        url: https://codeforces.com/contest/182/problem/D
        tag: kmp|circular_section|num_factor
        """
        s = ac.read_str()
        t = ac.read_str()
        m, n = len(s), len(t)
        c1 = m - KMP().prefix_function(s)[-1]
        c2 = n - KMP().prefix_function(t)[-1]
        if m % c1:
            c1 = m
        if n % c2:
            c2 = n
        if c1 != c2 or s[:c1] != t[:c2]:
            ac.st(0)
        else:
```

```
            ac.st(len(NumFactor().get_all_factor(math.gcd(m // c1, n // c2))))
        return

    def cf_535d(ac=FastIO()):
        """
        url: https://codeforces.com/contest/535/problem/D
        tag: kmp|z-function|union_find
        """
        n, k = ac.read_list_ints()
        s = ac.read_str()
        m = len(s)
        lst = [""] * n
        uf = UnionFind(n + 1)
        pos = ac.read_list_ints_minus_one()
        for i in pos:
            start = i
            end = i + m - 1
            while uf.find(i) <= end:
                j = uf.find(i)
                lst[j] = s[j - start]
                uf.union_right(j, j + 1)
                i = j + 1

        z = KMP().z_function(list(s) + ["#"] + lst)
        if not all(z[m + 1 + i] == m for i in pos):
            ac.st(0)
            return
        mod = 1000000007
        ans = pow(26, lst.count(""), mod)
        ac.st(ans)
        return

    def cf_1051e(ac=FastIO()):
        """
        url: https://codeforces.com/contest/1051/problem/E
        tag: kmp|z-function|linear_dp
        """
        s = ac.read_str()
        ll = ac.read_str()
        rr = ac.read_str()
        n = len(s)
        nll = len(ll)
        nrr = len(rr)
        zll = KMP().z_function(ll + "#" + s)[nll + 1:]
        zrr = KMP().z_function(rr + "#" + s)[nrr + 1:]

        def compare_ll(ind):
            lcp = zll[ind]
            if lcp == nll:
                return True
            return s[ind + lcp] >= ll[lcp]

        def compare_rr(ind):
            lcp = zrr[ind]
            if lcp == nrr:
                return True
```

```
            return s[ind + lcp] <= rr[lcp]

        mod = 998244353
        dp = [0] * (n + 1)
        post = [0] * (n + 1)
        dp[n] = post[n] = 1
        for i in range(n - 1, -1, -1):
            if s[i] == "0":
                if ll == "0":
                    dp[i] = dp[i + 1]
                post[i] = (post[i + 1] + dp[i]) % mod
                continue
            left = i + nll
            right = i + nrr
            if i + nll > n or not compare_ll(i):
                left += 1
            if i + nrr > n or not compare_rr(i):
                right -= 1
            if left <= right and left <= n:
                dp[i] = (post[left] - (post[right + 1] if right < n else 0)) %
mod
            post[i] = (post[i + 1] + dp[i]) % mod
        ac.st(dp[0])
        return

    def cf_1015f(ac=FastIO()):
        """
        url: https://codeforces.com/contest/1015/problem/F
        tag: kmp_automaton|matrix_dp
        """
        n = ac.read_int()
        s = ac.read_str()
        lst = [int(w == ")") for w in s]
        nxt = KMP().kmp_automaton(lst, 2)
        m = len(s)
        mod = 10 ** 9 + 7

        dp = [[0] * (m + 1) for _ in range(n + 1)]
        dp[0][0] = 1
        for _ in range(2 * n):
            ndp = [[0] * (m + 1) for _ in range(n + 1)]
            for s in range(n + 1):
                for p in range(m + 1):
                    if dp[s][p]:
                        for x in [0, 1]:
                            nxt_p = nxt[p * 2 + x] if p != m else m
                            nxt_s = s + 1 if not x else s - 1
                            if n >= nxt_s >= 0:
                                ndp[nxt_s][nxt_p] += dp[s][p]
            dp = [[x % mod for x in ls] for ls in ndp]
        ac.st(dp[0][m])
        return

    def cf_1690f(ac=FastIO()):
        """
        url: https://codeforces.com/contest/1690/problem/F
```

```
        tag: permutation_circle|kmp|circle_section
        """
        for _ in range(ac.read_int()):
            n = ac.read_int()
            s = ac.read_str()
            ind = ac.read_list_ints_minus_one()
            dct = {w: i for i, w in enumerate(ind)}
            ans = 1
            visit = [0] * n
            for i in range(n):
                if not visit[i]:
                    lst = [i]
                    visit[i] = 1
                    while not visit[dct[lst[-1]]]:
                        lst.append(dct[lst[-1]])
                        visit[lst[-1]] = 1
                    tmp = [s[j] for j in lst]
                    x = KMP().prefix_function(tmp)[-1]
                    if len(tmp) % (len(tmp) - x) == 0:
                        x = len(tmp) - x
                    else:
                        x = len(tmp)
                    ans = ans * x // math.gcd(ans, x)
            ac.st(ans)
        return

    def cf_1968g2(ac=FastIO()):
        """
        url: https://codeforces.com/contest/1968/problem/G2
        tag: z_algorithm|offline_query|binary_search|brute_force|preprocess
        """
        for _ in range(ac.read_int()):
            n, ll, rr = ac.read_list_ints()
            s = ac.read_str()
            z = KMP().z_function(s + "#" + s)

            lst = [(z[i + n + 1], i) for i in range(n)]
            lst.sort(reverse=True)

            ans = [-n - 1] * (n + 1)
            i = 0
            ind = SortedList()
            for x in range(n, 0, -1):
                while i < n and lst[i][0] >= x:
                    ind.add(lst[i][1])
                    i += 1
                if not ind:
                    continue
                cur = ind[0]
                cnt = 1
                while cur + x <= ind[-1]:
                    cur = ind[ind.bisect_left(cur + x)]
                    cnt += 1
                ans[x] = -cnt

            res = []
```

```
            for x in range(ll, rr + 1):
                res.append(bisect.bisect_right(ans, -x) - 1)

            ac.lst(res)
        return
```

2.manacher_palindrome

2.1 template:

```
class ManacherPlindrome:
    def __init__(self):
        return

    @staticmethod
    def max(a, b):
        return a if a > b else b

    @staticmethod
    def manacher(s):
        """template of get the palindrome radius for every i-th character as
center"""
        n = len(s)
        arm = [0] * n
        left, right = 0, -1
        for i in range(0, n):
            a, b = arm[left + right - i], right - i + 1
            a = a if a < b else b
            k = 0 if i > right else a
            while 0 <= i - k and i + k < n and s[i - k] == s[i + k]:
                k += 1
            arm[i] = k
            k -= 1
            if i + k > right:
                left = i - k
                right = i + k
        # s[i-arm[i]+1: i+arm[i]] is palindrome substring for every i
        return arm

    def palindrome_start_end(self, s: str) -> (list, list):
        """template of get the endpoint of palindrome substring for every i-th
character as start or end pos"""
        n = len(s)
        # trick to promise every palindrome substring has odd length
        # with # centered as the original even palindrome substring
        # letter centered as the original odd palindrome substring
        t = "#" + "#".join(list(s)) + "#"
        arm = self.manacher(t)
        m = len(t)

        # end position index of palindrome substring starting with the current
index as the boundary
        start = [[] for _ in range(n)]
        # the starting position index of the palindrome substring ending with the
current index as the boundary
        end = [[] for _ in range(n)]
        for j in range(m):
            left = j - arm[j] + 1
            right = j + arm[j] - 1
            while left <= right:
```

```
                if t[left] != "#":
                    start[left // 2].append(right // 2)
                    end[right // 2].append(left // 2)
                left += 1
                right -= 1
        return start, end

    def palindrome_post_pre(self, s: str) -> (list, list):
        """template of get the length of the longest palindrome substring that
starts or ends at a certain position"""
        n = len(s)
        t = "#" + "#".join(list(s)) + "#"
        arm = self.manacher(t)
        m = len(t)
        post = [1] * n
        pre = [1] * n
        for j in range(m):
            left = j - arm[j] + 1
            right = j + arm[j] - 1
            while left <= right:
                if t[left] != "#":
                    x, y = left // 2, right // 2
                    post[x] = max(post[x], y - x + 1)
                    pre[y] = max(pre[y], y - x + 1)
                    break
                left += 1
                right -= 1
        for i in range(1, n):
            if i - pre[i - 1] - 1 >= 0 and s[i] == s[i - pre[i - 1] - 1]:
                pre[i] = max(pre[i], pre[i - 1] + 2)
        for i in range(n - 2, -1, -1):
            pre[i] = max(pre[i], pre[i + 1] - 2)
        for i in range(n - 2, -1, -1):
            if i + post[i + 1] + 1 < n and s[i] == s[i + post[i + 1] + 1]:
                post[i] = max(post[i], post[i + 1] + 2)
        for i in range(1, n):
            post[i] = max(post[i], post[i - 1] - 2)

        return post, pre

    def palindrome_longest_length(self, s: str) -> (list, list):
        """template of get the longest palindrome substring of s"""
        t = "#" + "#".join(list(s)) + "#"
        arm = self.manacher(t)
        m = len(t)
        ans = 0
        for j in range(m):
            left = j - arm[j] + 1
            right = j + arm[j] - 1
            cur = (right - left + 1) // 2
            ans = ans if ans > cur else cur
        return ans

    def palindrome_just_start(self, s: str) -> (list, list):
        """template of get the endpoint of palindrome substring for every i-th
character as start or end pos"""
```

```
        n = len(s)
        # trick to promise every palindrome substring has odd length
        # with # centered as the original even palindrome substring
        # letter centered as the original odd palindrome substring
        t = "#" + "#".join(list(s)) + "#"
        arm = self.manacher(t)
        m = len(t)

        # end position index of palindrome substring starting with the current
index as the boundary
        start = []
        for j in range(m):
            left = j - arm[j] + 1
            right = j + arm[j] - 1
            while left <= right:
                if t[left] != "#":
                    if left // 2 == 0:
                        start.append(right // 2)
                    break
                left += 1
                right -= 1
        return start  # prefix palindrome

    def palindrome_just_end(self, s: str) -> (list, list):
        """template of get the endpoint of palindrome substring for every i-th
character as start or end pos"""
        n = len(s)
        # trick to promise every palindrome substring has odd length
        # with # centered as the original even palindrome substring
        # letter centered as the original odd palindrome substring
        t = "#" + "#".join(list(s)) + "#"
        arm = self.manacher(t)
        m = len(t)

        # end position index of palindrome substring starting with the current
index as the boundary
        end = []
        for j in range(m):
            left = j - arm[j] + 1
            right = j + arm[j] - 1
            while left <= right:
                if t[left] != "#":
                    if right // 2 == n-1:
                        end.append(left // 2)
                    break
                left += 1
                right -= 1
        return end  # suffix palindrome

    def palindrome_count_start_end(self, s: str) -> (list, list):
        """template of get the number of palindrome substring for every i-th
character as start or end pos"""
        n = len(s)
        # trick to promise every palindrome substring has odd length
        # with # centered as the original even palindrome substring
        # letter centered as the original odd palindrome substring
```

```
        t = "#" + "#".join(list(s)) + "#"
        arm = self.manacher(t)
        m = len(t)

        # end position index of palindrome substring starting with the current
index as the boundary
        start = [0] * n
        end = [0] * n
        for j in range(m):
            left = j - arm[j] + 1
            right = j + arm[j] - 1
            while left <= right:
                if t[left] != "#":
                    x, y = left // 2, right // 2
                    if (y - x + 1) % 2:
                        mid = x + (y - x + 1) // 2
                        start[x] += 1
                        if mid + 1 < n:
                            start[mid + 1] -= 1
                        end[mid] += 1
                        if y + 1 < n:
                            end[y + 1] -= 1
                    else:
                        mid = x + (y - x + 1) // 2 - 1
                        start[x] += 1
                        start[mid + 1] -= 1
                        end[mid + 1] += 1
                        if y + 1 < n:
                            end[y + 1] -= 1
                    break
                left += 1
                right -= 1
        for i in range(1, n):
            start[i] += start[i - 1]
            end[i] += end[i - 1]
        return start, end

    def palindrome_count_start_end_odd(self, s: str) -> (list, list):
        """template of get the number of palindrome substring for every i-th
character as start or end pos"""
        n = len(s)
        # trick to promise every palindrome substring has odd length
        # with # centered as the original even palindrome substring
        # letter centered as the original odd palindrome substring
        t = "#" + "#".join(list(s)) + "#"
        arm = self.manacher(t)
        m = len(t)

        # end position index of palindrome substring starting with the current
index as the boundary
        start = [0] * n
        end = [0] * n
        for j in range(m):
            left = j - arm[j] + 1
            right = j + arm[j] - 1
            while left <= right:
```

```
                if t[left] != "#":
                    x, y = left // 2, right // 2
                    if (y - x + 1) % 2:
                        mid = x + (y - x + 1) // 2
                        start[x] += 1
                        if mid + 1 < n:
                            start[mid + 1] -= 1
                        end[mid] += 1
                        if y + 1 < n:
                            end[y + 1] -= 1
                    break
                left += 1
                right -= 1
        for i in range(1, n):
            start[i] += start[i - 1]
            end[i] += end[i - 1]
        return start, end


    def palindrome_length_count(self, s: str) -> (list, list):
        """template of get the endpoint of palindrome substring for every i-th
character as start or end pos"""
        n = len(s)
        # trick to promise every palindrome substring has odd length
        # with # centered as the original even palindrome substring
        # letter centered as the original odd palindrome substring
        t = "#" + "#".join(list(s)) + "#"
        arm = self.manacher(t)
        m = len(t)

        # end position index of palindrome substring starting with the current
index as the boundary
        odd = [0] * (n + 2)
        even = [0] * (n + 2)

        for j in range(m):
            left = j - arm[j] + 1
            right = j + arm[j] - 1
            while left <= right:
                if t[left] != "#":
                    x, y = left // 2, right // 2
                    if (y - x + 1) % 2:
                        low, high = 1, (y - x + 2) // 2
                        odd[low] += 1
                        if high + 1 <= n:
                            odd[high + 1] -= 1
                    else:
                        low, high = 1, (y - x + 1) // 2
                        even[low] += 1
                        if high + 1 <= n:
                            even[high + 1] -= 1
                    break
                left += 1
                right -= 1
        cnt = [0] * (n + 1)
        for i in range(1, n + 1):
```

```
            odd[i] += odd[i - 1]
            even[i] += even[i - 1]
            if 2 * i - 1 <= n:
                cnt[2 * i - 1] += odd[i]
            if 2 * i <= n:
                cnt[2 * i] += even[i]
        return cnt

    def palindrome_count(self, s: str) -> (list, list):
        """template of get the endpoint of palindrome substring for every i-th
character as start or end pos"""
        # trick to promise every palindrome substring has odd length
        # with # centered as the original even palindrome substring
        # letter centered as the original odd palindrome substring
        t = "#" + "#".join(list(s)) + "#"
        arm = self.manacher(t)
        m = len(t)

        # end position index of palindrome substring starting with the current
index as the boundary
        ans = 0

        for j in range(m):
            left = j - arm[j] + 1
            right = j + arm[j] - 1
            while left <= right:
                if t[left] != "#":
                    x, y = left // 2, right // 2
                    if (y - x + 1) % 2:
                        ans += (y-x+2)//2
                    else:
                        ans += (y-x+1)//2
                    break
                left += 1
                right -= 1
        return ans
```

2.2 problem:

```
#Algorithm: manacher|palindrome_substring|plindrome_subsequence
#Description: dp|center|center_expansion_method|manacher
'''
====================================LuoGu=============================
P4555（https://www.luogu.com.cn/problem/P4555）
longest_palindrome_substring|prefix_suffix
P1210（https://www.luogu.com.cn/problem/P1210）longest_palindrome_substring
P4888（https://www.luogu.com.cn/problem/P4888）
center_expansion_method|two_pointers
P1872（https://www.luogu.com.cn/problem/P1872）
counter|palindrome_substring|manacher|classical
P6297（https://www.luogu.com.cn/problem/P6297）center_expansion_method|plindrome
P3805（https://www.luogu.com.cn/problem/P3805）palindrome_longest_length|manacher
P1659（https://www.luogu.com.cn/problem/P1659）manacher|palindrome_length_count
P3501（https://www.luogu.com.cn/problem/P3501）
manacher|palindrome_length_count|classical|change_manacher
P6216（https://www.luogu.com.cn/problem/P6216）
P5446（https://www.luogu.com.cn/problem/P5446）
==================================CodeForces==========================
1682A（https://codeforces.com/contest/1682/problem/A）palindromic|center_extension
1326D2（https://codeforces.com/problemset/problem/1326/D2）
palindrome_post_pre|manacher
7D（https://codeforces.com/problemset/problem/7/D）palindrome_just_start|manacher
835D（https://codeforces.com/problemset/problem/835/D）
17E（https://codeforces.com/contest/17/problem/E）
palindrome_count_start_end|manacher
1081H（https://codeforces.com/problemset/problem/1081/H）
1827C（https://codeforces.com/contest/1827/problem/C）
====================================================================
'''
```

```
from src.strings.manacher_palindrome.template import ManacherPlindrome
from src.utils.fast_io import FastIO

class Solution:
    def __init__(self):
        return

    def lg_4555(s):
        """
        url: https://www.luogu.com.cn/problem/P4555
        tag: longest_palindrome_substring|prefix_suffix
        """
        n = len(s)
        post, pre = ManacherPlindrome().palindrome_post_pre(s)
        ans = max(post[i + 1] + pre[i] for i in range(n - 1))
        return ans

    def lg_p1872(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P1872
```

```
        tag: counter|palindrome_substring|manacher|classical
        """

        s = ac.read_str()
        n = len(s)
        start, end = ManacherPlindrome().palindrome_start_end(s)
        start = [len(x) for x in start]
        end = [len(x) for x in end]
        pre = ans = 0
        for i in range(n):
            ans += pre * start[i]
            pre += end[i]
        ac.st(ans)
        return

    def lg_p6297(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P6297
        tag: center_expansion_method|plindrome
        """

        n, k = ac.read_list_ints()
        mod = 10 ** 9 + 7
        nums = ac.read_list_ints()
        ans = 0
        for i in range(n):
            cur = nums[i]
            rem = k
            x, y = i - 1, i + 1
            while x >= 0 and y < n:
                if nums[x] != nums[y]:
                    if not rem:
                        break
                    rem -= 1
                cur *= nums[x] * nums[y]
                x -= 1
                y += 1
            ans = ac.max(ans, cur)

            if i + 1 < n:
                cur = 0
                rem = k
                x, y = i, i + 1
                while x >= 0 and y < n:
                    if nums[x] != nums[y]:
                        if not rem:
                            break
                        rem -= 1
                    cur = cur if cur else 1
                    cur *= nums[x] * nums[y]
                    x -= 1
                    y += 1
                ans = ac.max(ans, cur)
        ac.st(ans % mod)
        return
```

```python
    def lg_p3805(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P3805
        tag: palindrome_longest_length|manacher
        """
        s = ac.read_str()
        ans = ManacherPlindrome().palindrome_longest_length(s)
        ac.st(ans)
        return

    def lg_p1659(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P1659
        tag: manacher|palindrome_length_count
        """
        n, k = ac.read_list_ints()
        s = ac.read_str()
        cnt = ManacherPlindrome().palindrome_length_count(s)
        ans = 1
        mod = 19930726
        for i in range(n, 0, -1):
            if i % 2:
                x = ac.min(cnt[i], k)
                ans *= pow(i, x, mod)
                ans %= mod
                k -= x
                if not k:
                    ac.st(ans)
                    return
        ac.st(-1)
        return

    def lg_p3501(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P3501
        tag: manacher|palindrome_length_count|classical|change_manacher
        """

        def manacher(s):
            """template of get the palindrome radius for every i-th character as
center"""
            n = len(s)
            arm = [0] * n
            left, right = 0, -1
            for i in range(0, n):
                a, b = arm[left + right - i], right - i + 1
                a = a if a < b else b
                k = 0 if i > right else a
                while 0 <= i - k and i + k < n and (
                        (s[i - k] != s[i + k] and s[i - k] != "#" and s[i + k] !=
"#") or s[i - k] == s[i + k] == "#"):
                    k += 1
                arm[i] = k
                k -= 1
                if i + k > right:
                    left = i - k
```

```
                right = i + k
        # s[i-arm[i]+1: i+arm[i]] is palindrome substring for every i
        return arm

    # 这里想记录以下，是1326的题目，但是代码原作者误触弄反了，在此更正
    def cf_1362d2(ac=FastIO()):
        """
        url: https://codeforces.com/problemset/problem/1326/D2
        tag: palindrome_post_pre|manacher
        """
        for _ in range(ac.read_int()):
            s = ac.read_str()
            n = len(s)

            i, j = 0, n - 1
            while i < j and s[i] == s[j]:
                i += 1
                j -= 1
            ans = s[:i]
            s = s[i:j + 1]
            post, pre = ManacherPlindrome().palindrome_post_pre(s)
            n = len(s)
            mid = ""
            for i in range(n - 1, -1, -1):
                if pre[i] == i + 1:
                    mid = s[:i + 1]
                    break
            for i in range(n):
                if post[i] == n - i:
                    if n - i > len(mid):
                        mid = s[-post[i]:]
                    break
            ac.st(ans + mid + ans[::-1])
        return
```

3.palindrome_num

3.1 template:

```
class PalindromeNum:
    def __init__(self):
        return

    @staticmethod
    def get_palindrome_num_1(n):
        """template of get all positive palindrome number with length not greater
than n"""
        dp = [[""], [str(i) for i in range(10)]]
        for k in range(2, n + 1):
            # like dp to add palindrome character
            if k % 2 == 1:
                m = k // 2
                lst = []
                for st in dp[-1]:
                    for i in range(10):
                        lst.append(st[:m] + str(i) + st[m:])
                dp.append(lst)
            else:
                lst = []
                for st in dp[-2]:
                    for i in range(10):
                        lst.append(str(i) + st + str(i))
                dp.append(lst)

        nums = []
        for lst in dp:
            for num in lst:
                if num and num[0] != "0":
                    nums.append(int(num))
        nums.sort()
        return nums

    @staticmethod
    def get_palindrome_num_2(n):
        assert n >= 1
        """template of get all positive palindrome number whose length not
greater than n"""
        nums = list(range(1, 10))
        x = 1
        while len(str(x)) * 2 <= n:
            num = str(x) + str(x)[::-1]
            nums.append(int(num))
            if len(str(x)) * 2 + 1 <= n:
                for d in range(10):
                    nums.append(int(str(x) + str(d) + str(x)[::-1]))
            x += 1
        nums.sort()
        return nums
```

```python
    @staticmethod
    def get_palindrome_num_3():
        """template of get all positive palindrome number whose length not
greater than n"""
        nums = list(range(10))
        for i in range(1, 10 ** 5):
            nums.append(int(str(i) + str(i)[::-1]))
            for j in range(10):
                nums.append(int(str(i) + str(j) + str(i)[::-1]))
        nums.sort()
        return nums

    @staticmethod
    def get_recent_palindrome_num(n: str) -> list:
        """template of recentest palindrome num of n"""
        m = len(n)
        candidates = [10 ** (m - 1) - 1, 10 ** m + 1]
        prefix = int(n[:(m + 1) // 2])
        for x in range(prefix - 1, prefix + 2):
            y = x if m % 2 == 0 else x // 10
            while y:
                x = x * 10 + y % 10
                y //= 10
            candidates.append(x)
        return candidates
```

3.2 problem:

```
# Algorithm：palindrome_number|brute_force
# Description：
```

~~关于回文串的题目比较少，先把后面的模板整理完了之后自己在洛谷或者cf上面找点题目然后补充上去~~

4.expression

4.1 template:

```
class Node(object):
    def __init__(self, val=" ", left=None, right=None):
        if val[0] == "+" and len(val) >= 2:
            val = val[1:]
        self.val = val
        self.left = left
        self.right = right


class TreeExpression:
    def __init__(self):
        return

    def exp_tree(self, s: str) -> Node:

        try:
            int(s)
            # number rest only
            return Node(s)
        except ValueError as _:
            pass

        # case start with -(
        if len(s) >= 2 and s[0] == "-" and s[1] == "(":
            cnt = 0
            for i, w in enumerate(s):
                if w == "(":
                    cnt += 1
                elif w == ")":
                    cnt -= 1
                    if not cnt:
                        pre = s[1:i].replace("+", "-").replace("-", "+")
                        s = pre + s[i:]
                        break
        # case start with -
        neg = ""
        if s[0] == "-":
            neg = "-"
            s = s[1:]

        n = len(s)
        cnt = 0
        # 按照运算符号的优先级reverse_order|遍历字符串
        for i in range(n - 1, -1, -1):
            cnt += int(s[i] == ')') - int(s[i] == '(')
            if s[i] in ['+', '-'] and not cnt:
                return Node(s[i], self.exp_tree(neg + s[:i]), self.exp_tree(s[i +
1:]))

        # 注意是从后往前
```

```
        for i in range(n - 1, -1, -1):
            cnt += int(s[i] == ')') - int(s[i] == '(')
            if s[i] in ['*', '/'] and not cnt:
                return Node(s[i], self.exp_tree(neg + s[:i]), self.exp_tree(s[i +
1:]))

        # 注意是从前往后
        for i in range(n):
            cnt += int(s[i] == ')') - int(s[i] == '(')
            if s[i] in ['^'] and not cnt:  # 这里的 ^ 表示幂
                return Node(s[i], self.exp_tree(neg + s[:i]), self.exp_tree(s[i +
1:]))

        # 其余则是开头结尾为括号的情况
        return self.exp_tree(s[1:-1])

    def main_1175(self, s):

        # 按照前序、中序与后序变成前缀中缀与后缀表达式
        def dfs(node):
            if not node:
                return
            dfs(node.left)
            dfs(node.right)
            pre.append(node.val)
            return

        ans = []
        root = self.exp_tree(s)
        pre = []
        dfs(root)
        while len(pre) > 1:
            ans.append(pre)
            n = len(pre)
            stack = []
            for i in range(n):
                if pre[i] in "+-*/^":
                    op = pre[i]
                    b = stack.pop()
                    a = stack.pop()
                    op = "//" if op == "/" else op
                    op = "**" if op == "^" else op
                    stack.append(str(eval(f"{a}{op}{b}")))
                    stack += pre[i + 1:]
                    break
                else:
                    stack.append(pre[i])
            pre = stack[:]
        ans.append(pre)
        return ans


class EnglishNumber:
    def __init__(self):
        return
```

```python
    @staticmethod
    def number_to_english(n):

        # 将 0-9999 的数字转换为美式英语即有 and
        one = ["", "one", "two", "three", "four",
               "five", "six", "seven", "eight", "nine",
               "ten", "eleven", "twelve", "thirteen", "fourteen",
               "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"]

        ten = [
            "twenty",
            "thirty",
            "forty",
            "fifty",
            "sixty",
            "seventy",
            "eighty",
            "ninety"]

        for word in ten:
            one.append(word)
            for i in range(1, 10):
                one.append(word + " " + one[i])

        ans = ""
        s = str(n)
        if n >= 1000:
            ans += one[n // 1000] + " thousand "

        if (n % 1000) // 100 > 0:
            ans += one[n % 1000 // 100] + " hundred "
        if (n >= 100 and 0 < n % 100 < 10) or (n >= 1000 and 0 < n % 1000 < 100):
            ans += "and "
        ans += one[n % 100]

        if ans == "":
            return "zero"
        return ans
```

4.2 problem:

```
# Algorithm：stack
# Description：xxx
'''
====================================LuoGu=============================
P1175（https://www.luogu.com.cn/problem/P1175）
P1617（https://www.luogu.com.cn/problem/P1617）
P1322（https://www.luogu.com.cn/problem/P1322）
'''
```

~~这个部分的也没有代码实现，也自己找一些题目然后弄上来吧~~

5.lyndon_decomposition

5.1 template:

```
class LyndonDecomposition:
    def __init__(self):
        return


    @staticmethod
    def solve_by_duval(s):
        """template of duval algorithm"""
        n, i = len(s), 0
        factorization = []
        while i < n:
            j, k = i + 1, i
            while j < n and s[k] <= s[j]:
                if s[k] < s[j]:
                    k = i
                else:
                    k += 1
                j += 1
            while i <= k:
                factorization.append(s[i: i + j - k])
                i += j - k
        return factorization

    @staticmethod
    def min_cyclic_string(s):
        """template of smallest cyclic string"""
        s += s
        n = len(s)
        i, ans = 0, 0
        while i < n // 2:
            ans = i
            j, k = i + 1, i
            while j < n and s[k] <= s[j]:
                if s[k] < s[j]:
                    k = i
                else:
```

```
                k += 1
            j += 1
        while i <= k:
            i += j - k
    return s[ans: ans + n // 2]

@staticmethod
def min_express(sec):
    """template of minimum lexicographic expression"""
    n = len(sec)
    k, i, j = 0, 0, 1
    while k < n and i < n and j < n:
        if sec[(i + k) % n] == sec[(j + k) % n]:
            k += 1
        else:
            if sec[(i + k) % n] > sec[(j + k) % n]:
                i = i + k + 1
            else:
                j = j + k + 1
            if i == j:
                i += 1
            k = 0
    i = i if i < j else j
    return i, sec[i:] + sec[:i]

@staticmethod
def max_express(sec):
    """template of maximum lexicographic expression"""
    n = len(sec)
    k, i, j = 0, 0, 1
    while k < n and i < n and j < n:
        if sec[(i + k) % n] == sec[(j + k) % n]:
            k += 1
        else:
            if sec[(i + k) % n] < sec[(j + k) % n]:
                i = i + k + 1
            else:
                j = j + k + 1
            if i == j:
                i += 1
            k = 0
    i = i if i < j else j
    return i, sec[i:] + sec[:i]
```

5.2 problem:

```
# Algorithm：lyndon_composition|minimum_expression|maximum_expression
# Description：rotate_string|lexicographical_order
'''
====================================LuoGu=============================
P1368（https://www.luogu.com.cn/problem/P1368）lyndon_decomposition|min_express
==================================CodeForces==========================
496B（https://codeforces.com/problemset/problem/496/B）
lyndon_decomposition|min_express
========================================================================
'''
```

```
from src.strings.lyndon_decomposition.template import LyndonDecomposition
from src.utils.fast_io import FastIO

class Solution:

    def __init__(self):
        return

    def lg_p1368(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P1368
        tag: lyndon_decomposition|min_express
        """
        ac.read_int()
        lst = ac.read_list_ints()
        ans = LyndonDecomposition().min_express(lst)
        ac.lst(ans[1])
        return


    def cf_496b(ac=FastIO()):
        """
        url: https://codeforces.com/problemset/problem/496/B
        tag: lyndon_decomposition|min_express
        """
        ac.read_int()
        lst = [int(w) for w in ac.read_str()]
        ld = LyndonDecomposition()
        ans = ld.min_express(lst)[1]
        for _ in range(10):
            lst = [(x + 1) % 10 for x in lst]
            cur = ld.min_express(lst)[1]
            if cur < ans:
                ans = cur
        ac.st("".join(str(x) for x in ans))
        return
```

~~这个题目也很少，但是这个考得也不多就写上模板有两道题目就算了，摆烂ing~~

6.string_hash

6.1 template:

```
import math
import random

from src.utils.fast_io import inf


class MatrixHashReverse:
    def __init__(self, m, n, grid):
        """
        primes = PrimeSieve().eratosthenes_sieve(100)
        primes = [x for x in primes if 26 < x < 100]
        """
        primes = [29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
        self.m, self.n = m, n

        self.p1 = primes[random.randint(0, len(primes) - 1)]
        while True:
            self.p2 = primes[random.randint(0, len(primes) - 1)]
            if self.p2 != self.p1:
                break

        ceil = self.m if self.m > self.n else self.n
        self.pp1 = [1] * (ceil + 1)
        self.pp2 = [1] * (ceil + 1)
        self.mod = random.randint(10 ** 9 + 7, (1 << 31) - 1)

        for i in range(1, ceil):
            self.pp1[i] = (self.pp1[i - 1] * self.p1) % self.mod
            self.pp2[i] = (self.pp2[i - 1] * self.p2) % self.mod

        # (x+1, y+1)
        # (i,j) > (i-1, j)p1 (i, j-1)p2 (i-1, j-1) p1p2
        self.left_up = [0] * (self.n + 1) * (self.m + 1)
        for i in range(self.m):
            for j in range(self.n):
                val = self.left_up[i * (self.n + 1) + j + 1] * self.p1 +
self.left_up[
                    (i + 1) * (self.n + 1) + j] * self.p2
                val -= self.left_up[i * (self.n + 1) + j] * self.p1 * self.p2 -
grid[i * self.n + j]
                self.left_up[(i + 1) * (self.n + 1) + j + 1] = val % self.mod

        # (x+1, y)
        # (i,j) > (i-1, j)p1 (i, j+1)p2 (i-1, j+1) p1p2
        self.right_up = [0] * (self.n + 1) * (self.m + 1)
        for i in range(self.m):
            for j in range(self.n - 1, -1, -1):
                val = self.right_up[i * (self.n + 1) + j] * self.p1 +
self.right_up[
                    (i + 1) * (self.n + 1) + j + 1] * self.p2
```

```
                val -= self.right_up[i * (self.n + 1) + j + 1] * self.p1 *
self.p2 - grid[i * self.n + j]
                self.right_up[(i + 1) * (self.n + 1) + j] = val % self.mod

        # (x, y)
        # (i,j) > (i+1, j)p1 (i, j+1)p2 (i+1, j+1) p1p2
        self.right_down = [0] * (self.n + 1) * (self.m + 1)
        for i in range(self.m - 1, -1, -1):
            for j in range(self.n - 1, -1, -1):
                val = self.right_down[(i + 1) * (self.n + 1) + j] * self.p1 +
self.right_down[
                    i * (self.n + 1) + j + 1] * self.p2
                val -= self.right_down[(i + 1) * (self.n + 1) + j + 1] * self.p1
* self.p2 - grid[i * self.n + j]
                self.right_down[i * (self.n + 1) + j] = val % self.mod

        # (x, y+1)
        # (i,j) > (i+1, j)p1 (i, j-1)p2 (i+1, j-1) p1p2
        self.left_down = [0] * (self.n + 1) * (self.m + 1)
        for i in range(self.m - 1, -1, -1):
            for j in range(self.n):
                val = self.left_down[(i + 1) * (self.n + 1) + j + 1] * self.p1 +
self.left_down[
                    i * (self.n + 1) + j] * self.p2
                val -= self.left_down[(i + 1) * (self.n + 1) + j] * self.p1 *
self.p2 - grid[i * self.n + j]
                self.left_down[i * (self.n + 1) + j + 1] = val % self.mod
        return

    def query_left_up(self, i, j, a, b):
        # (x+1, y+1)
        # (i,j) > (i-a, j)p1 (i, j-b)p2 (i-a, j-b) p1p2
        res = self.left_up[(i + 1) * (self.n + 1) + j + 1]
        res -= self.left_up[(i - a + 1) * (self.n + 1) + j + 1] * self.pp1[a] +
self.left_up[
            (i + 1) * (self.n + 1) + j - b + 1] * self.pp2[b]
        res += self.left_up[(i - a + 1) * (self.n + 1) + j - b + 1] * self.pp1[a]
* self.pp2[b]
        return res % self.mod

    def query_right_up(self, i, j, a, b):
        # (x+1, y)
        # (i,j) > (i-a, j)p1 (i, j+b)p2 (i-a, j+b) p1p2
        res = self.right_up[(i + 1) * (self.n + 1) + j]
        res -= self.right_up[(i - a + 1) * (self.n + 1) + j] * self.pp1[a] +
self.right_up[
            (i + 1) * (self.n + 1) + j + b] * self.pp2[b]
        res += self.right_up[(i - a + 1) * (self.n + 1) + j + b] * self.pp1[a] *
self.pp2[b]
        return res % self.mod

    def query_right_down(self, i, j, a, b):
        # (x, y)
        # (i,j) > (i+a, j)p1 (i, j+b)p2 (i+a, j+b) p1p2
        res = self.right_down[i * (self.n + 1) + j]
```

```
        res -= self.right_down[(i + a) * (self.n + 1) + j] * self.pp1[a] +
self.right_down[i * (self.n + 1) + j + b] * \
               self.pp2[b]
        res += self.right_down[(i + a) * (self.n + 1) + (j + b)] * self.pp1[a] *
self.pp2[b]
        return res % self.mod

    def query_left_down(self, i, j, a, b):
        # (x, y+1)
        # (i,j) > (i+a, j)p1 (i, j-b)p2 (i+a, j-b) p1p2
        res = self.left_down[i * (self.n + 1) + j + 1]
        res -= self.left_down[(i + a) * (self.n + 1) + j + 1] * self.pp1[a] +
self.left_down[
            i * (self.n + 1) + j - b + 1] * self.pp2[b]
        res += self.left_down[(i + a) * (self.n + 1) + j - b + 1] * self.pp1[a] *
self.pp2[b]
        return res % self.mod

    def query_in_build(self, i, j, a, b):
        assert 0 <= i <= i + a - 1 < self.m
        assert 0 <= j <= j + b - 1 < self.n
        res = self.left_up[(i + a) * (self.n + 1) + j + b] - self.left_up[i *
(self.n + 1) + j + b] * self.pp1[a] - \
              self.left_up[
                  (i + a) * (self.n + 1) + j] * self.pp2[b]
        res += self.left_up[i * (self.n + 1) + j] * self.pp1[a] * self.pp2[b]
        return res % self.mod


class MatrixHash:
    def __init__(self, m, n, grid):
        """
        primes = PrimeSieve().eratosthenes_sieve(100)
        primes = [x for x in primes if 26 < x < 100]
        """
        primes = [29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
        self.m, self.n = m, n

        self.p1 = primes[random.randint(0, len(primes) - 1)]
        while True:
            self.p2 = primes[random.randint(0, len(primes) - 1)]
            if self.p2 != self.p1:
                break

        ceil = self.m if self.m > self.n else self.n
        self.pp1 = [1] * (ceil + 1)
        self.pp2 = [1] * (ceil + 1)
        self.mod = random.randint(10 ** 9 + 7, (1 << 31) - 1)

        for i in range(1, ceil):
            self.pp1[i] = (self.pp1[i - 1] * self.p1) % self.mod
            self.pp2[i] = (self.pp2[i - 1] * self.p2) % self.mod

        self.pre = [0] * (self.n + 1) * (self.m + 1)

        for i in range(self.m):
```

```
            for j in range(self.n):
                val = self.pre[i * (self.n + 1) + j + 1] * self.p1 + self.pre[(i
+ 1) * (self.n + 1) + j] * self.p2
                val -= self.pre[i * (self.n + 1) + j] * self.p1 * self.p2 -
grid[i * self.n + j]
                self.pre[(i + 1) * (self.n + 1) + j + 1] = val % self.mod
        return

    def query_sub(self, i, j, a, b):
        # right_down corner
        assert a - 1 <= i < self.m
        assert b - 1 <= j < self.n
        res = self.pre[(i + 1) * (self.n + 1) + j + 1]
        res -= self.pre[(i - a + 1) * (self.n + 1) + j + 1] * self.pp1[a] +
self.pre[
            (i + 1) * (self.n + 1) + j - b + 1] * self.pp2[b]
        res += self.pre[(i - a + 1) * (self.n + 1) + j - b + 1] * self.pp1[a] *
self.pp2[b]
        return res % self.mod

    def query_matrix(self, a, b, mat):
        cur = [0] * (b + 1) * (a + 1)
        for i in range(a):
            for j in range(b):
                val = cur[i * (b + 1) + j + 1] * self.p1 + cur[(i + 1) * (b + 1)
+ j] * self.p2
                val -= cur[i * (b + 1) + j] * self.p1 * self.p2 - mat[i * b + j]
                cur[(i + 1) * (b + 1) + j + 1] = val % self.mod
        return cur[-1]


class StringHash:
    def __init__(self, lst):
        """two mod to avoid hash crush"""
        # use two class to compute is faster!!!
        self.n = len(lst)
        self.p = random.randint(26, 100)
        self.mod = random.randint(10 ** 9 + 7, 2 ** 31 - 1)

        self.pre = [0] * (self.n + 1)
        self.pp = [1] * (self.n + 1)
        for j, w in enumerate(lst):
            self.pre[j + 1] = (self.pre[j] * self.p + w) % self.mod
            self.pp[j + 1] = (self.pp[j] * self.p) % self.mod
        return

    def query(self, x, y):
        """range hash value index start from 0"""
        # assert 0 <= x <= y <= self.n - 1
        if y < x:
            return 0
        # with length y - x + 1 important!!!
        ans = (self.pre[y + 1] - self.pre[x] * self.pp[y - x + 1]) % self.mod
        return ans, y - x + 1
```

```python
class StringHashSingle:
    def __init__(self, lst):
        """two mod to avoid hash crush"""
        # use two class to compute is faster!!!
        self.n = len(lst)
        base = max(max(lst) + 1, 150)
        self.p = random.randint(base, base * 2)
        self.mod = random.getrandbits(64)

        self.pre = [0] * (self.n + 1)
        self.pp = [1] * (self.n + 1)
        for j, w in enumerate(lst):
            self.pre[j + 1] = (self.pre[j] * self.p + w) % self.mod
            self.pp[j + 1] = (self.pp[j] * self.p) % self.mod
        return

    def query(self, x, y):
        """range hash value index start from 0"""
        # assert 0 <= x <= y <= self.n - 1
        if y < x:
            return 0
        # with length y - x + 1 important!!!
        ans = (self.pre[y + 1] - self.pre[x] * self.pp[y - x + 1]) % self.mod
        return ans, y - x + 1

    def check(self, lst):
        ans = 0
        for w in lst:
            ans = (ans * self.p + w) % self.mod
        return ans, len(lst)


class PointSetRangeHashReverse:
    def __init__(self, n) -> None:
        self.n = n
        self.p = random.randint(26, 100)  # self.p = random.randint(150, 300)
        self.mod = random.randint(10 ** 9 + 7, (1 << 31) - 1)  # self.mod =
random.getrandbits(64)
        self.pp = [1] * (n + 1)
        for j in range(n):
            self.pp[j + 1] = (self.pp[j] * self.p) % self.mod
        self.left_to_right = [0] * (4 * n)
        self.right_to_left = [0] * (4 * n)
        return

    def build(self, nums):
        stack = [(0, self.n - 1, 1)]
        while stack:
            s, t, i = stack.pop()
            if i >= 0:
                if s == t:
                    self.left_to_right[i] = nums[s]
                    self.right_to_left[i] = nums[s]
                else:
                    stack.append((s, t, ~i))
                    m = s + (t - s) // 2
```

```
                    stack.append((s, m, i << 1))
                    stack.append((m + 1, t, (i << 1) | 1))
            else:
                i = ~i
                self._push_up(i, s, t)
        return

    def _push_up(self, i: int, s, t) -> None:
        m = s + (t - s) // 2
        length = t - m
        self.left_to_right[i] = (self.left_to_right[i << 1] * self.pp[length] +
self.left_to_right[
            (i << 1) | 1]) % self.mod

        length = m - s + 1
        self.right_to_left[i] = (self.right_to_left[(i << 1) | 1] *
self.pp[length] + self.right_to_left[
            i << 1]) % self.mod
        return

    def get(self):
        stack = [(0, self.n - 1, 1)]
        nums = [0] * self.n
        while stack:
            s, t, i = stack.pop()
            if s == t:
                nums[s] = self.left_to_right[i]
                continue
            m = s + (t - s) // 2
            stack.append((s, m, i << 1))
            stack.append((m + 1, t, (i << 1) | 1))
        return nums

    def point_set(self, left: int, right: int, val: int) -> None:
        stack = [(0, self.n - 1, 1)]
        while stack:
            s, t, i = stack.pop()
            if i >= 0:
                if left <= s and t <= right:
                    self.right_to_left[i] = self.left_to_right[i] = val
                    continue
                m = s + (t - s) // 2
                stack.append((s, t, ~i))
                if left <= m:
                    stack.append((s, m, i << 1))
                if right > m:
                    stack.append((m + 1, t, (i << 1) | 1))
            else:
                i = ~i
                self._push_up(i, s, t)
        return

    def range_hash(self, left: int, right: int):
        stack = [(0, self.n - 1, 1)]
        ans = 0
        while stack:
```

```
            s, t, i = stack.pop()
            if left <= s and t <= right:
                length = t - s + 1
                ans = (ans * self.pp[length] + self.left_to_right[i]) % self.mod
                continue
            m = s + (t - s) // 2
            if right > m:
                stack.append((m + 1, t, (i << 1) | 1))
            if left <= m:
                stack.append((s, m, i << 1))
        return ans

    def range_hash_reverse(self, left: int, right: int):
        stack = [(0, self.n - 1, 1)]
        ans = 0
        while stack:
            s, t, i = stack.pop()
            if left <= s and t <= right:
                length = t - s + 1
                ans = (ans * self.pp[length] + self.right_to_left[i]) % self.mod
                continue
            m = s + (t - s) // 2
            if left <= m:
                stack.append((s, m, i << 1))
            if right > m:
                stack.append((m + 1, t, (i << 1) | 1))
        return ans


class RangeSetRangeHashReverse:
    def __init__(self, n, tag=inf) -> None:
        self.n = n
        self.tag = tag
        while True:
            self.p = random.randint(26, 100)
            self.mod = random.randint(10 ** 9 + 7, (1 << 31) - 1)
            if math.gcd(self.p - 1, self.mod) == 1:
                break
        self.pp = [1] * (n + 1)
        for j in range(n):
            self.pp[j + 1] = (self.pp[j] * self.p) % self.mod
        self.rev = pow(self.p - 1, -1, self.mod)
        self.left_to_right = [0] * (4 * n)
        self.right_to_left = [0] * (4 * n)
        self.lazy = [self.tag] * (4 * self.n)
        return

    def build(self, nums):
        stack = [(0, self.n - 1, 1)]
        while stack:
            s, t, i = stack.pop()
            if i >= 0:
                if s == t:
                    self._make_tag(nums[s], i, s, t)
                else:
                    stack.append((s, t, ~i))
```

```
                    m = s + (t - s) // 2
                    stack.append((s, m, i << 1))
                    stack.append((m + 1, t, (i << 1) | 1))
            else:
                i = ~i
                self._push_up(i, s, t)
        return

    def _make_tag(self, val: int, i: int, s, t) -> None:
        self.lazy[i] = val
        m = t - s + 1
        self.left_to_right[i] = (val * (self.pp[m] - 1) * self.rev) % self.mod
        self.right_to_left[i] = (val * (self.pp[m] - 1) * self.rev) % self.mod
        return

    def _push_down(self, i: int, s, t) -> None:
        m = s + (t - s) // 2
        if self.lazy[i] != self.tag:
            self._make_tag(self.lazy[i], i << 1, s, m)
            self._make_tag(self.lazy[i], (i << 1) | 1, m + 1, t)
            self.lazy[i] = self.tag
        return

    def _push_up(self, i: int, s, t) -> None:
        m = s + (t - s) // 2
        length = t - m
        self.left_to_right[i] = (self.left_to_right[i << 1] * self.pp[length] +
self.left_to_right[
            (i << 1) | 1]) % self.mod

        length = m - s + 1
        self.right_to_left[i] = (self.right_to_left[(i << 1) | 1] *
self.pp[length] + self.right_to_left[
            (i << 1)]) % self.mod
        return

    def get(self):
        stack = [(0, self.n - 1, 1)]
        nums = [0] * self.n
        while stack:
            s, t, i = stack.pop()
            if s == t:
                nums[s] = self.left_to_right[i]
                continue
            m = s + (t - s) // 2
            self._push_down(i, s, t)
            stack.append((s, m, i << 1))
            stack.append((m + 1, t, (i << 1) | 1))
        return nums

    def range_set(self, left: int, right: int, val: int) -> None:
        stack = [(0, self.n - 1, 1)]
        while stack:
            s, t, i = stack.pop()
            if i >= 0:
                if left <= s and t <= right:
```

```
                    self._make_tag(val, i, s, t)
                    continue
                m = s + (t - s) // 2
                self._push_down(i, s, t)
                stack.append((s, t, ~i))

                if left <= m:
                    stack.append((s, m, i << 1))
                if right > m:
                    stack.append((m + 1, t, (i << 1) | 1))
            else:
                i = ~i
                self._push_up(i, s, t)
        return

    def range_hash(self, left: int, right: int):
        stack = [(0, self.n - 1, 1)]
        ans = 0
        while stack:
            s, t, i = stack.pop()
            if left <= s and t <= right:
                length = t - s + 1
                ans = (ans * self.pp[length] + self.left_to_right[i]) % self.mod
                continue
            m = s + (t - s) // 2
            self._push_down(i, s, t)
            if right > m:
                stack.append((m + 1, t, (i << 1) | 1))
            if left <= m:
                stack.append((s, m, i << 1))
        return ans

    def range_hash_reverse(self, left: int, right: int):
        stack = [(0, self.n - 1, 1)]
        ans = 0
        while stack:
            s, t, i = stack.pop()
            if left <= s and t <= right:
                length = t - s + 1
                ans = (ans * self.pp[length] + self.right_to_left[i]) % self.mod
                continue
            m = s + (t - s) // 2
            self._push_down(i, s, t)
            if left <= m:
                stack.append((s, m, i << 1))
            if right > m:
                stack.append((m + 1, t, (i << 1) | 1))
        return ans
```

6.2 problem:

```
# Algorithm:
string_hash|tree_hash|matrix_hash|tree_minimum_expression|longest_prefix_palindrome_substring|longest_suffix_palindrome_substring
# Description: counter|sliding_window|double_random_mod|hash_crush
'''
==========================================LuoGu==========================================
P6140（https://www.luogu.com.cn/problem/P6140）
greedy|implemention|lexicographical_order|string_hash|binary_search|reverse_order|lcs
P2870（https://www.luogu.com.cn/problem/P2870）
greedy|implemention|lexicographical_order|string_hash|binary_search|reverse_order|lcs
P5832（https://www.luogu.com.cn/problem/P5832）string_hash
P2852（https://www.luogu.com.cn/problem/P2852）
binary_search|suffix_array|height|monotonic_queue|string_hash
P4656（https://www.luogu.com.cn/problem/P4656）string_hash|greedy
P6739（https://www.luogu.com.cn/problem/P6739）prefix_suffix|string_hash
P3370（https://www.luogu.com.cn/problem/P3370）string_hash
P2601（https://www.luogu.com.cn/problem/P2601）matrix_hash
P4824（https://www.luogu.com.cn/problem/P4824）string_hash
P4503（https://www.luogu.com.cn/problem/P4503）string_hash
P3538（https://www.luogu.com.cn/problem/P3538）
string_hash|prime_factor|brute_force|circular_section
======================================CodeForces======================================
1800D（https://codeforces.com/contest/1800/problem/D）prefix_suffix|hash
514C（https://codeforces.com/problemset/problem/514/C）string_hash
1200E（https://codeforces.com/problemset/problem/1200/E）string_hash|kmp
580E（https://codeforces.com/problemset/problem/580/E）
segment_tree_hash|range_change|range_hash_reverse|circular_section
452F（https://codeforces.com/contest/452/problem/F）
segment_tree_hash|string_hash|point_set|range_hash|range_reverse
7D（https://codeforces.com/problemset/problem/7/D）
string_hash|palindrome|classical
835D（https://codeforces.com/problemset/problem/835/D）palindrome|string_hash
======================================================================================
'''
```

```
import random
from collections import defaultdict, Counter
from itertools import accumulate
from typing import List

from src.basis.binary_search.template import BinarySearch
from src.graph.dijkstra.template import Dijkstra
from src.mathmatics.fast_power.template import MatrixFastPower
from src.mathmatics.prime_factor.template import PrimeFactor
from src.strings.string_hash.template import StringHash, PointSetRangeHashReverse, RangeSetRangeHashReverse, \
    MatrixHash, MatrixHashReverse, StringHashSingle
from src.utils.fast_io import FastIO, inf
```

```python
class Solution:
    def __init__(self):
        return

    def lg_p2852(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P2852
        tag: binary_search|suffix_array|height|monotonic_queue|string_hash
        """

        def check(x):
            pre = defaultdict(int)
            for i in range(n):
                if i >= x - 1:
                    pre[(sh1.query(i - x + 1, i), sh2.query(i - x + 1, i))] += 1
            return max(pre.values()) >= k

        n, k = ac.read_list_ints()
        nums = [ac.read_int() for _ in range(n)]
        sh1 = StringHash(nums)
        sh2 = StringHash(nums)
        ans = BinarySearch().find_int_right(0, n, check)
        ac.st(ans)

        return

    def lg_p4656(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P4656
        tag: string_hash|greedy
        """
        # string_hashgreedy选取

        p1 = random.randint(26, 100)
        p2 = random.randint(26, 100)
        mod1 = random.randint(10 ** 9 + 7, 2 ** 31 - 1)
        mod2 = random.randint(10 ** 9 + 7, 2 ** 31 - 1)

        for _ in range(ac.read_int()):
            s = ac.read_str()
            ans = 0
            n = len(s)
            i, j = 0, n - 1
            while j - i + 1 >= 2:
                # 从两边依次选取
                flag = False
                pre1 = post1 = pre2 = post2 = 0
                pp1 = pp2 = 1
                x, y = i, j
                while True:
                    if pre1 == post1 and pre2 == post2 and x > i:
                        flag = True
                        i = x
                        j = y
                        break
                    if y - x + 1 <= 1:
```

```
                    break
                w = s[x]
                pre1 = (pre1 * p1 + ord(w) - ord("a")) % mod1
                pre2 = (pre2 * p2 + ord(w) - ord("a")) % mod2

                w = s[y]
                post1 = (post1 + pp1 * (ord(w) - ord("a"))) % mod1
                post2 = (post2 + pp2 * (ord(w) - ord("a"))) % mod2
                pp1 = (pp1 * p1) % mod1
                pp2 = (pp2 * p2) % mod2
                x += 1
                y -= 1
            # 如果构成一对回文增| 2 否则增| 1
            if flag:
                ans += 2
            else:
                ans += 1
                i = j + 1
                break
        # 特判还剩中间一个字母的情况
        if i == j:
            ans += 1
        ac.st(ans)

    return

def lg_p4656(ac=FastIO()):
    """
    url: https://www.luogu.com.cn/problem/P4656
    tag: string_hash|greedy
    """
    # string_hashgreedy选取

    p1 = random.randint(26, 100)
    p2 = random.randint(26, 100)
    mod1 = random.randint(10 ** 9 + 7, 2 ** 31 - 1)
    mod2 = random.randint(10 ** 9 + 7, 2 ** 31 - 1)

    for _ in range(ac.read_int()):
        s = ac.read_str()
        ans = 0
        n = len(s)
        i, j = 0, n - 1
        while j - i + 1 >= 2:
            # 从两边依次选取
            flag = False
            pre1 = post1 = pre2 = post2 = 0
            pp1 = pp2 = 1
            x, y = i, j
            while True:
                if pre1 == post1 and pre2 == post2 and x > i:
                    flag = True
                    i = x
                    j = y
                    break
                if y - x + 1 <= 1:
```

```
                    break
                w = s[x]
                pre1 = (pre1 * p1 + ord(w) - ord("a")) % mod1
                pre2 = (pre2 * p2 + ord(w) - ord("a")) % mod2

                w = s[y]
                post1 = (post1 + pp1 * (ord(w) - ord("a"))) % mod1
                post2 = (post2 + pp2 * (ord(w) - ord("a"))) % mod2
                pp1 = (pp1 * p1) % mod1
                pp2 = (pp2 * p2) % mod2
                x += 1
                y -= 1
            # 如果构成一对回文增| 2 否则增| 1
            if flag:
                ans += 2
            else:
                ans += 1
                i = j + 1
                break
        # 特判还剩中间一个字母的情况
        if i == j:
            ans += 1
        ac.st(ans)

    return

@staticmethod
def lg_p6739(ac=FastIO()):
    """
    url: https://www.luogu.com.cn/problem/P6739
    tag: prefix_suffix|string_hash
    """
    n = ac.read_int()
    s = ac.read_str()
    if n % 2 == 0:
        ac.st("NOT POSSIBLE")
        return

    sh1 = StringHash([ord(w) - ord("a") for w in s])
    sh2 = StringHash([ord(w) - ord("a") for w in s])

    ans = dict()
    for i in range(n):
        if len(ans) > 1:
            break
        if i < n // 2:
            ss = (sh1.query(0, i - 1), sh2.query(0, i - 1))
            tt = (sh1.query(i + 1, n // 2), sh2.query(i + 1, n // 2))
            a = (ss[0] * sh1.pp[n // 2 - i] + tt[0]) % sh1.mod
            b = (ss[1] * sh2.pp[n // 2 - i] + tt[1]) % sh2.mod

            if sh1.query(n // 2 + 1, n - 1) == a and sh2.query(n // 2 + 1, n
- 1) == b:
                ans[(a, b)] = i

        elif i == n // 2:
```

```
                a, b = sh1.query(0, n // 2 - 1), sh2.query(0, n // 2 - 1)
                if sh1.query(n // 2 + 1, n - 1) == a and sh2.query(n // 2 + 1, n
- 1) == b:
                    ans[(a, b)] = i
            else:
                ss = (sh1.query(n // 2, i - 1), sh2.query(n // 2, i - 1))
                tt = (sh1.query(i + 1, n - 1), sh2.query(i + 1, n - 1))
                a = (ss[0] * sh1.pp[n - 1 - i] + tt[0]) % sh1.mod
                b = (ss[1] * sh2.pp[n - 1 - i] + tt[1]) % sh2.mod

                if sh1.query(0, n // 2 - 1) == a and sh2.query(0, n // 2 - 1) ==
b:
                    ans[(a, b)] = i
        if not ans:
            ac.st("NOT POSSIBLE")
        elif len(ans) > 1:
            ac.st("NOT UNIQUE")
        else:
            i = list(ans.values())[0]
            if i >= n // 2:
                ac.st(s[:n // 2])
            elif i < n // 2:
                ac.st(s[-(n // 2):])
        return

    def lg_p3370_1(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P3370
        tag: string_hash
        """
        ans = set()
        p1, p2 = [random.randint(26, 100), random.randint(26, 100)]
        mod1, mod2 = [random.randint(10 ** 9 + 7, 2 ** 31 - 1), random.randint(10
** 9 + 7, 2 ** 31 - 1)]
        for _ in range(ac.read_int()):
            s = ac.read_str()
            x1 = x2 = 0
            for w in s:
                x1 = (x1 * p1 + ord(w)) % mod1
                x2 = (x2 * p2 + ord(w)) % mod2
            ans.add((x1, x2))
        ac.st(len(ans))
        return

    def lg_p3370_2(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P3370
        tag: string_hash
        """
        ans = set()
        for _ in range(ac.read_int()):
            ans.add(hash(ac.read_str()))
        ac.st(len(ans))
        return

    def lg_p2601(ac=FastIO()):
```

```
        """
        url: https://www.luogu.com.cn/problem/P2601
        tag: matrix_hash|string_hash
        """
        m, n = ac.read_list_ints()
        lst = []
        for _ in range(m):
            lst.extend(ac.read_list_ints())
        mh = MatrixHashReverse(m, n, lst)

        def check1(x):

            res1 = mh.query_left_up(i + x - 1, j + x - 1, 2 * x - 1, 2 * x - 1)
            res2 = mh.query_right_up(i + x - 1, j - x + 1, 2 * x - 1, 2 * x - 1)
            res3 = mh.query_left_down(i - x + 1, j + x - 1, 2 * x - 1, 2 * x - 1)
            res4 = mh.query_right_down(i - x + 1, j - x + 1, 2 * x - 1, 2 * x -
1)
            return res1 == res2 == res3 == res4

        def check2(x):

            res1 = mh.query_left_up(i + x, j + x, 2 * x, 2 * x)
            res2 = mh.query_right_up(i + x, j - x + 1, 2 * x, 2 * x)
            res3 = mh.query_left_down(i - x + 1, j + x, 2 * x, 2 * x)
            res4 = mh.query_right_down(i - x + 1, j - x + 1, 2 * x, 2 * x)
            return res1 == res2 == res3 == res4

        bs = BinarySearch()
        ans = i = j = 0

        for i in range(m):
            for j in range(n):
                y = min(i + 1, m - i, j + 1, n - j)
                ans += bs.find_int_right(0, y, check1)
                y = min(i + 1, j + 1, m - i - 1, n - j - 1)
                ans += bs.find_int_right(0, y, check2)
        ac.st(ans)
        return

    def lg_p4824(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P4824
        tag: string_hash|stack|implemention
        """

        s = ac.read_str()
        t = ac.read_str()
        m, n = len(s), len(t)
        sh = StringHash([ord(w) - ord("a") for w in s + t])
        del t
        target = sh.query(m, m + n - 1)
        i = 0
        stack = []
        for w in s:
            x = ord(w) - ord("a")
```

```
            stack.append(w)
            sh.pre[i + 1] = (sh.pre[i] * sh.p + x) % sh.mod
            i += 1
            if i >= n and sh.query(i - n, i - 1) == target:
                i -= n
                for _ in range(n):
                    stack.pop()
        ac.st("".join(stack))
        return

    def lg_p4503(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P4503
        tag: string_hash
        """

        ind = dict()
        words = []
        for i in range(10):
            ind[str(i)] = i
            words.append(str(i))
        for i in range(26):
            ind[chr(i + ord("a"))] = 10 + i
            words.append(chr(i + ord("a")))
        for i in range(26):
            ind[chr(i + ord("A"))] = 36 + i
            words.append(chr(i + ord("A")))
        ind["_"] = 62
        ind["@"] = 63
        words.extend(["_", "@"])

        n, ll, s = ac.read_list_ints()
        sh1 = StringHash([0] * ll)
        sh2 = StringHash([0] * ll)
        cnt = dict()
        ans = 0
        for _ in range(n):
            st = [ind[w] + 1 for w in ac.read_str()]
            for j in range(ll):
                sh1.pre[j + 1] = (sh1.pre[j] * sh1.p + st[j]) % sh1.mod
                sh2.pre[j + 1] = (sh2.pre[j] * sh2.p + st[j]) % sh2.mod
            for j in range(ll):
                pre1 = sh1.pre[j]
                post1 = sh1.query(j + 1, ll - 1)
                pre2 = sh2.pre[j]
                post2 = sh2.query(j + 1, ll - 1)
                right = ll - j - 1
                cur1 = (pre1 * sh1.pp[right + 1] + post1) % sh1.mod
                cur2 = (pre2 * sh2.pp[right + 1] + post2) % sh2.mod
                ans += cnt.get((cur1, cur2), 0)
                cnt[(cur1, cur2)] = cnt.get((cur1, cur2), 0) + 1
        ac.st(ans)
        return

    @staticmethod
    def lg_p3538(ac=FastIO()):
```

```
    """
    url: https://www.luogu.com.cn/problem/P3538
    tag: string_hash|prime_factor|brute_force|circular_section
    """
    n = ac.read_int()
    lst = [ord(w) - ord("a") for w in ac.read_str()]
    sh1 = StringHash(lst)
    sh2 = StringHash(lst)

    def check(cur):
        pre1 = sh1.query(a, b - cur)
        post1 = sh1.query(a + cur, b)

        pre2 = sh2.query(a, b - cur)
        post2 = sh2.query(a + cur, b)
        return pre1 == post1 and pre2 == post2

    pf = PrimeFactor(n)
    for _ in range(ac.read_int()):
        a, b = ac.read_list_ints_minus_one()
        length = b - a + 1
        ans = length
        while length > 1:
            p = pf.min_prime[length]
            if check(ans // p):
                ans //= p
            length //= p
        ac.st(ans)
    return

def cf_1800d_1(ac=FastIO()):
    """
    url: https://codeforces.com/contest/1800/problem/D
    tag: prefix_suffix|hash
    """

    n = 2 * 10 ** 5
    p1 = random.randint(26, 100)
    p2 = random.randint(26, 100)
    mod1 = random.randint(10 ** 9 + 7, 2 ** 31 - 1)
    mod2 = random.randint(10 ** 9 + 7, 2 ** 31 - 1)

    dp1 = [1]
    for _ in range(1, n + 1):
        dp1.append((dp1[-1] * p1) % mod1)
    dp2 = [1]
    for _ in range(1, n + 1):
        dp2.append((dp2[-1] * p2) % mod2)

    for _ in range(ac.read_int()):
        n = ac.read_int()
        s = ac.read_str()

        post1 = [0] * (n + 1)
        for i in range(n - 1, -1, -1):
```

```
                post1[i] = (post1[i + 1] + (ord(s[i]) - ord("a")) * dp1[n - 1 -
i]) % mod1

            post2 = [0] * (n + 1)
            for i in range(n - 1, -1, -1):
                post2[i] = (post2[i + 1] + (ord(s[i]) - ord("a")) * dp2[n - 1 -
i]) % mod2

            ans = set()
            pre1 = pre2 = 0
            for i in range(n - 1):
                x1 = pre1
                y1 = post1[i + 2]
                x2 = pre2
                y2 = post2[i + 2]
                ans.add(((x1 * dp1[n - i - 2] + y1) % mod1, (x2 * dp2[n - i - 2]
+ y2) % mod2))
                pre1 = (pre1 * p1) % mod1 + ord(s[i]) - ord("a")
                pre2 = (pre2 * p2) % mod2 + ord(s[i]) - ord("a")
            ac.st(len(ans))
        return

    def cf_1800d_2(ac=FastIO()):
        """
        url: https://codeforces.com/contest/1800/problem/D
        tag: prefix_suffix|hash
        """

        for _ in range(ac.read_int()):
            n = ac.read_int()
            s = ac.read_str()
            ans = n - 1
            for i in range(2, n):
                if s[i] == s[i - 2]:
                    ans -= 1
            ac.st(ans)
        return

    def cf_1200e(ac=FastIO()):
        """
        url: https://codeforces.com/contest/1200/problem/E
        tag: string_hash|kmp
        """
        ac.read_int()
        lst = ac.read_list_strs()
        n = sum(len(s) for s in lst)
        p = [random.randint(26, 100), random.randint(26, 100)]
        mod = [random.randint(10 ** 9 + 7, 2 ** 31 - 1), random.randint(10 ** 9 +
7, 2 ** 31 - 1)]
        pre = [[0] * (n + 1), [0] * (n + 1)]
        pp = [[1] * (n + 1), [1] * (n + 1)]
        for j in range(n):
            for i in range(2):
                pp[i][j + 1] = (pp[i][j] * p[i]) % mod[i]

        def query1(x, y):
```

```
            if y < x:
                return 0, 0
            res = tuple((pre[ii][y + 1] - pre[ii][x] * pp[ii][y - x + 1]) %
mod[ii] for ii in range(2))
            return res

        def query2(x, y):
            if y < x:
                return 0, 0
            res = tuple((cur[ii][y + 1] - cur[ii][x] * pp[ii][y - x + 1]) %
mod[ii] for ii in range(2))
            return res

        ans = []
        k = 0
        for word in lst:
            m = len(word)
            cur = [[0] * (m + 1), [0] * (m + 1)]
            inter = 0
            for j, w in enumerate(word):
                for i in range(2):
                    cur[i][j + 1] = (cur[i][j] * p[i] + ord(w)) % mod[i]
                if query1(k - j - 1, k - 1) == query2(0, j):
                    inter = j + 1
            for j in range(inter, m):
                w = word[j]
                ans.append(w)
                for i in range(2):
                    pre[i][k + 1] = (pre[i][k] * p[i] + ord(w)) % mod[i]
                k += 1
        ac.st("".join(ans))
        return

    def cf_580e(ac=FastIO()):
        """
        url: https://codeforces.com/contest/580/problem/E
        tag: segment_tree_hash|range_change|range_hash_reverse|circular_section
        """
        n, m, k = ac.read_list_ints()
        tree1 = RangeSetRangeHashReverse(n, 10)
        tree2 = RangeSetRangeHashReverse(n, 10)
        s = ac.read_str()
        tree1.build([int(w) for w in s])
        tree2.build([int(w) for w in s])
        for _ in range(m + k):
            lst = ac.read_list_ints()
            if lst[0] == 1:
                l, r, c = lst[1:]
                tree1.range_set(l - 1, r - 1, c)
                tree2.range_set(l - 1, r - 1, c)
            else:
                l, r, d = lst[1:]
                if d == r - l + 1:
                    ac.yes()
                    continue
                else:
```

```
                    if tree1.range_hash(l - 1, r - d - 1) == tree1.range_hash(l +
d - 1, r - 1):
                        if tree2.range_hash(l - 1, r - d - 1) ==
tree2.range_hash(l + d - 1, r - 1):
                            ac.yes()
                        else:
                            ac.no()
                    else:
                        ac.no()
        return

    def cf_452f(ac=FastIO()):
        """
        url: https://codeforces.com/contest/452/problem/F
        tag: segment_tree_hash|string_hash|point_set|range_hash|range_reverse
        """
        n = ac.read_int()
        tree1 = PointSetRangeHashReverse(n)
        tree2 = PointSetRangeHashReverse(n)
        nums = ac.read_list_ints_minus_one()
        for num in nums:
            tree1.point_set(num, num, 1)
            tree2.point_set(num, num, 1)
            if num == 0 or num == n - 1:
                continue
            length = ac.min(num + 1, n - num)
            cur1 = [tree1.range_hash(num - length + 1, num), tree2.range_hash(num
- length + 1, num)]
            cur2 = [tree1.range_hash_reverse(num, num + length - 1),
tree2.range_hash_reverse(num, num + length - 1)]
            if cur1 != cur2:
                ac.yes()
                break
        else:
            ac.no()
        return

    def cf_7d(ac=FastIO()):
        """
        url: https://codeforces.com/problemset/problem/7/D
        tag: string_hash|palindrome|classical
        """
        p = 131
        mod = 10 ** 9 + 7
        s = ac.read_str()
        n = len(s)
        pre = rev = 0
        pp = 1
        dp = [0] * (n + 1)
        for i in range(n):
            x = ord(s[i]) - ord("a")
            pre = (pre * p + x) % mod
            rev = (x * pp + rev) % mod
            pp = (pp * p) % mod
            if pre != rev:
                continue
```

```
            dp[i + 1] = dp[(i + 1) // 2] + 1
        ac.st(sum(dp))
        return

    def cf_835d(ac=FastIO()):
        """
        url: https://codeforces.com/problemset/problem/835/D
        tag: palindrome|string_hash
        """
        s = ac.read_str()
        lst = [ord(w) - ord("a") for w in s]
        n = len(s)
        dp = [0] * (n + 1)
        ans = [0] * (n + 1)
        p = 131
        mod = 10 ** 9 + 7
        for i in range(n):
            dp[i] = 1
            pp = p
            pre = rev = lst[i]
            ans[1] += 1
            for j in range(i + 1, n):
                pre = (pre * p + lst[j]) % mod
                rev = (lst[j] * pp + rev) % mod
                pp = (pp * p) % mod
                if pre == rev:
                    dp[j] = dp[i + (j - i + 1) // 2 - 1] + 1
                    ans[dp[j]] += 1
                else:
                    dp[j] = 0
        for i in range(n - 1, -1, -1):
            ans[i] += ans[i + 1]
        ac.lst(ans[1:])
        return
```

7.suffix_array

7.1 template:

```
class SuffixArray:

    def __init__(self):
        return

    @staticmethod
    def build(s, sig):
        # sa: index is rank and value is pos
        # rk: index if pos and value is rank
        # height: lcp of rank i-th suffix and (i-1)-th suffix
        # sum(height): count of same substring of s
        # n*(n+1)//2 - sum(height): count of different substring of s
        # max(height): can compute the longest duplicate substring,
        # which is s[i: i + height[j]] and j = height.index(max(height)) and i =
sa[j]
        # sig: number of unique rankings which initially is the size of the
character set

        n = len(s)
        sa = list(range(n))
        rk = s[:]
        ll = 0  # ll is the length that has already been sorted, and now it needs
to be sorted by 2ll length
        tmp = [0] * n
        while True:
            p = [i for i in range(n - ll, n)] + [x - ll for i, x in enumerate(sa)
if x >= ll]
            # for suffixes with a length less than l, their second keyword
ranking is definitely
            # the smallest because they are all empty
            # for suffixes of other lengths, suffixes starting at 'sa [i]' rank
i-th, and their
            # first ll characters happen to be the second keyword of suffixes
starting at 'sa[i] - ll'
            # start cardinality sorting, and first perform statistics on the
first keyword
            # first, count how many values each has
            cnt = [0] * sig
            for i in range(n):
                cnt[rk[i]] += 1
            # make a prefix and for easy cardinality sorting
            for i in range(1, sig):
                cnt[i] += cnt[i - 1]

            # then use cardinality sorting to calculate the new sa
            for i in range(n - 1, -1, -1):
                w = rk[p[i]]
                cnt[w] -= 1
                sa[cnt[w]] = p[i]
```

```
            # new_sa to check new_rk
            def equal(ii, jj, lll):
                if rk[ii] != rk[jj]:
                    return False
                if ii + lll >= n and jj + lll >= n:
                    return True
                if ii + lll < n and jj + lll < n:
                    return rk[ii + lll] == rk[jj + lll]
                return False

            sig = -1
            for i in range(n):
                tmp[i] = 0

            for i in range(n):
                # compute the lcp
                if i == 0 or not equal(sa[i], sa[i - 1], ll):
                    sig += 1
                tmp[sa[i]] = sig

            for i in range(n):
                rk[i] = tmp[i]
            sig += 1
            if sig == n:
                break
            ll = ll << 1 if ll > 0 else 1

        # height
        k = 0
        height = [0] * n
        for i in range(n):
            if rk[i] > 0:
                j = sa[rk[i] - 1]
                while i + k < n and j + k < n and s[i + k] == s[j + k]:
                    k += 1
                height[rk[i]] = k
                k = 0 if k - 1 < 0 else k - 1
        return sa, rk, height
```

7.2 problem:

```
# Algorithm：suffix_array
# Description：suffix_array
'''
====================================LuoGu=================================
P3809（https://www.luogu.com.cn/problem/P3809）suffix_array
P2852（https://www.luogu.com.cn/problem/P2852）
binary_search|suffix_array|height|monotonic_queue|string_hash
P2408（https://www.luogu.com.cn/problem/P2408）suffix_array|height
P3804（https://www.luogu.com.cn/problem/P3804）suffix_array|height|monotonic_stack
P4248（https://www.luogu.com.cn/problem/P4248）
suffix_array|height|lcp|monotonic_stack
P3975（https://www.luogu.com.cn/problem/P3975）greedy|bfs|suffix_array|height
P3796（https://www.luogu.com.cn/problem/P3796）
suffix_array|height|sa|monotonic_stack|prefix_sum
P5546（https://www.luogu.com.cn/problem/P5546）
suffix_array|lcs|lcp|monotonic_queue
P4341（https://www.luogu.com.cn/problem/P4341）suffix_array|height
P4070（https://www.luogu.com.cn/problem/P4070）
P6095（https://www.luogu.com.cn/problem/P6095）
================================CodeForces================================
123D（https://codeforces.com/problemset/problem/123/D）
suffix_array|height|monotonic_stack
271D（https://codeforces.com/contest/271/problem/D）
suffix_array|height|different_limited_substring
802I（https://codeforces.com/contest/802/problem/I）
suffix_array|height|monotonic_stack
128B（https://codeforces.com/contest/128/problem/B）greedy|bfs|suffix_array|height
427D（https://codeforces.com/contest/427/problem/D）
suffix_array|height|sa|lcp|trick|lcs
1526E（https://codeforces.com/contest/1526/problem/E）
suffix_array|reverse_thinking|comb|construction
611D（https://codeforces.com/problemset/problem/611/D）
600A（https://codeforces.com/problemset/problem/600/A）
873F（https://codeforces.com/contest/873/problem/F）
suffix_array|reverse_thinking|lcp|prefix_sum
================================================================================
'''
```

```
from collections import deque
from functools import cmp_to_key
from typing import List

from src.basis.binary_search.template import BinarySearch
from src.data_structure.monotonic_stack.template import Rectangle
from src.data_structure.sorted_list.template import SortedList
from src.data_structure.sparse_table.template import SparseTable
from src.mathmatics.comb_perm.template import Combinatorics
from src.strings.suffix_array.template import SuffixArray
from src.utils.fast_io import FastIO, inf

class Solution:
```

```
    def __init__(self):
        return

    def lg_p3809(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P3809
        tag: suffix_array
        """
        words = ([str(x) for x in range(10)]
                 + [chr(i + ord("A")) for i in range(26)]
                 + [chr(i + ord("a")) for i in range(26)])
        ind = {st: i for i, st in enumerate(words)}
        rk = [ind[w] for w in ac.read_str()]
        sa = SuffixArray().build(rk, len(ind))[0]
        ac.lst([x + 1 for x in sa])
        return

    def lg_p2852(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P2852
        tag: binary_search|suffix_array|height|monotonic_queue
        """
        n, k = ac.read_list_ints()
        s = [ac.read_int() for _ in range(n)]
        ind = {num: i for i, num in enumerate(sorted(list(set(s))))}
        s = [ind[d] for d in s]

        _, _, height = SuffixArray().build(s, len(ind))
        stack = deque()
        ans = []
        for i in range(n):
            while stack and stack[0][1] <= i - (k - 1):
                stack.popleft()
            while stack and stack[-1][0] >= height[i]:
                stack.pop()
            stack.append((height[i], i))
            if i >= (k - 1) - 1:
                ans.append(stack[0][0])
        ac.st(max(ans))
        return

    def lg_p2408(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P2408
        tag: suffix_array|height
        """
        ac.read_int()
        s = [ord(w) - ord("a") for w in ac.read_str()]
        sa, rk, height = SuffixArray().build(s, 26)
        n = len(s)
        ans = sum(height)
        ac.st(n * (n + 1) // 2 - ans)
        return

    def lg_p3804(ac=FastIO()):
        """
```

```
        url: https://www.luogu.com.cn/problem/P3804
        tag: suffix_array|height|monotonic_stack
        """

        s = ac.read_str()
        sa, rk, height = SuffixArray().build([ord(w) - ord("a") for w in s], 26)
        n = len(s)
        left = [1] * n
        right = [n - 1] * n
        stack = []
        for i in range(n):
            while stack and height[stack[-1]] > height[i]:
                right[stack.pop()] = i - 1
            stack.append(i)

        stack = []
        for i in range(n - 1, -1, -1):
            while stack and height[stack[-1]] > height[i]:
                left[stack.pop()] = i + 1
            stack.append(i)

        ans = 0
        for i in range(n):
            cur = height[i] * (right[i] - left[i] + 2)
            ans = ans if ans > cur else cur

        ac.st(ans)
        return

    def lg_p4248(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P4248
        tag: suffix_array|height|lcp
        """
        s = [ord(w) - ord("a") for w in ac.read_str()]
        sa, rk, height = SuffixArray().build(s, 26)
        height.append(0)
        n = len(s)
        pre = [0] * (n + 1)
        stack = [0]
        for i in range(n):
            if i:
                pre[i] = pre[i - 1]
            while stack[-1] and height[stack[-1]] > height[i]:
                j = stack.pop()
                pre[i] -= (j - stack[-1]) * height[j]
            pre[i] += (i - stack[-1]) * height[i]
            stack.append(i)

        post = [0] * (n + 1)
        stack = [n]
        for i in range(n - 1, -1, -1):
            post[i] = post[i + 1]
            while stack[-1] < n and height[stack[-1]] > height[i]:
                j = stack.pop()
                post[i] -= (stack[-1] - j) * height[j]
```

```
            post[i] += (stack[-1] - i) * height[i]
            stack.append(i)

        suf = ans = 0
        for i in range(n - 1, -1, -1):
            ans += (n - i) * (n - i - 1) + suf
            suf += n - i

        for i in range(n):
            ans -= pre[rk[i]] + post[rk[i] + 1]
        ac.st(ans)
        return

    def lg_p3796(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P3796
        tag: suffix_array|height|sa|monotonic_stack|prefix_sum
        """
        while True:  # MLE
            k = ac.read_int()
            if not k:
                break
            lst = []
            for _ in range(k + 1):
                lst.append([ord(w) - ord("a") for w in ac.read_str()])

            ind = []
            nums = []
            for i in range(k + 1):
                m = len(lst[i])
                ind.extend([i] * m)
                ind.append(i)
                nums.extend(lst[i])
                nums.append(26 + i)

            sa, _, height = SuffixArray().build(nums, 26 + k + 1)
            del nums

            n = len(height)
            height.append(0)

            right = [n - 1] * (n + 1)
            stack = []
            for i in range(n):
                while stack and height[stack[-1]] > height[i]:
                    right[stack.pop()] = i - 1
                stack.append(i)

            left = [0] * (n + 1)
            stack = []
            for i in range(n - 1, -1, -1):
                while stack and height[stack[-1]] > height[i]:
                    left[stack.pop()] = i + 1
                stack.append(i)
            pre = ac.accumulate([int(ind[sa[i]] == k) for i in range(n)])
            cnt = [0] * k
```

```
            for i in range(n):
                j = sa[i]
                if ind[j] < k and height[i] == len(lst[ind[j]]):
                    a, b = left[i], right[i]

                    if pre[b + 1] - pre[a - 1] > cnt[ind[j]]:
                        cnt[ind[j]] = pre[b + 1] - pre[a - 1]
            ceil = max(cnt)
            ac.st(ceil)
            for i in range(k):
                if cnt[i] == ceil:
                    ac.st("".join([chr(ord("a") + w) for w in lst[i]]))
        return

    def lg_p5546(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P5546
        tag: suffix_array|lcs|lcp|monotonic_queue
        """
        lst = []
        for _ in range(ac.read_int()):
            lst.append([ord(w) - ord("a") for w in ac.read_str()])

        ind = []
        nums = []
        k = len(lst)
        for i in range(k):
            m = len(lst[i])
            ind.extend([i] * m)
            ind.append(i)
            nums.extend(lst[i])
            nums.append(26 + i)
        if k == 1:
            return len(nums)
        sa, rk, height = SuffixArray().build(nums, 26 + k)

        def add(ii):
            nonlocal cnt
            for w in [ind[sa[ii - 1]], ind[sa[ii]]]:
                if not item[w]:
                    cnt += 1
                item[w] += 1
            return

        def remove(ii):
            nonlocal cnt
            for w in [ind[sa[ii - 1]], ind[sa[ii]]]:
                item[w] -= 1
                if not item[w]:
                    cnt -= 1
            return

        ans = cnt = 0
        j = 1
        item = [0] * k
        n = len(height)
```

```
        stack = deque()
        for i in range(1, n):
            while stack and stack[0] < i:
                stack.popleft()
            while j < n and cnt < k:
                add(j)
                while stack and height[stack[-1]] > height[j]:
                    stack.pop()
                stack.append(j)
                j += 1
            if cnt == k and stack and height[stack[0]] > ans:
                ans = height[stack[0]]
            remove(i)

        ac.st(ans)
        return

    def lg_p4341(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P4341
        tag: suffix_array|height
        """
        n = ac.read_int()
        lst = [int(w) for w in ac.read_str()]
        sa, rk, height = SuffixArray().build(lst, 2)

        for i in range(n):
            j = sa[i]
            for x in range(height[i] + 1, n - j + 1):
                y = 1
                for k in range(i + 1, n):
                    if height[k] >= x:
                        y += 1
                    else:
                        break
                if y > 1:
                    ac.st(y)
        return

    def cf_123d_1(ac=FastIO()):
        """
        url: https://codeforces.com/problemset/problem/123/D
        tag: suffix_array|height|monotonic_stack
        """
        s = [ord(w) - ord("a") for w in ac.read_str()]
        sa, rk, height = SuffixArray().build(s, 26)
        ans = Rectangle().compute_number([h + 1 for h in height])
        ac.st(ans)
        return

    def cf_123d_2(ac=FastIO()):
        """
        url: https://codeforces.com/problemset/problem/123/D
        tag: suffix_array|height|monotonic_stack
        """
        s = [ord(w) - ord("a") for w in ac.read_str()]
```

```
        n = len(s)
        sa, rk, height = SuffixArray().build(s, 26)
        ans = Rectangle().compute_number(height)
        ans += sum(n - sa[j] for j in range(n))
        ac.st(ans)
        return

    def cf_271d(ac=FastIO()):
        """
        url: https://codeforces.com/contest/271/problem/D
        tag: suffix_array|height|different_limited_substring
        """
        s = [ord(w) - ord("a") for w in ac.read_str()]
        good = [1 - int(w) for w in ac.read_str()]
        k = ac.read_int()
        n = len(s)
        j = cnt = ans = 0
        right = [0] * n
        for i in range(n):
            while j < n and cnt + good[s[j]] <= k:
                cnt += good[s[j]]
                j += 1
            ans += j - i
            right[i] = j
            cnt -= good[s[i]]
        sa, rk, height = SuffixArray().build(s, 26)
        dup = sum(min(height[i], right[sa[i]] - sa[i]) for i in range(1, n))
        ans -= dup
        ac.st(ans)
        return

    def cf_802i(ac=FastIO()):
        """
        url: https://codeforces.com/contest/802/problem/I
        tag: suffix_array|height|monotonic_stack
        """

        for _ in range(ac.read_int()):
            s = ac.read_str()
            n = len(s)
            sa, rk, height = SuffixArray().build([ord(w) - ord("a") for w in s],
26)
            stack = [[-1, 0]]
            ans = n * (n + 1) // 2
            for i in range(n):
                while stack[-1][0] >= 0 and height[i] < height[stack[-1][0]]:
                    stack.pop()
                stack.append([i, stack[-1][1] + (i - stack[-1][0]) * height[i]])
                ans += stack[-1][1] * 2
            ac.st(ans)

        return

    def cf_128b(ac=FastIO()):
        """
        url: https://codeforces.com/contest/128/problem/B
```

```
        tag: greedy|bfs|suffix_array|height
        """
        s = ac.read_str()
        lst = [ord(w) - ord("a") for w in s]
        k = ac.read_int()
        n = len(s)
        if k > n * (n + 1) // 2:
            ac.st("No such line.")
            return
        ans = []
        ind = list(range(n))
        while ind and k > 0:
            dct = [[] for _ in range(26)]
            for i in ind:
                dct[lst[i]].append(i)
            for x in range(26):
                cur = sum(n - i for i in dct[x])
                if cur < k:
                    k -= cur
                else:
                    ans.append(chr(x + ord("a")))
                    ind = [i + 1 for i in dct[x] if i + 1 < n]
                    k -= len(dct[x])
                    break
        ac.st("".join(ans))
        return

    def cf_427d(ac=FastIO()):
        """
        url: https://codeforces.com/contest/427/problem/D
        tag: suffix_array|height|sa|lcp|trick|lcs
        """
        s = ac.read_str()
        nums1 = [ord(w) - ord("a") for w in s]
        nums2 = [ord(w) - ord("a") for w in ac.read_str()]
        m, n = len(nums1), len(nums2)
        nums = nums1 + [26] + nums2
        sa, rk, height = SuffixArray().build(nums, 27)
        height.append(0)
        ans = inf
        for i in range(1, m + n + 1):
            if not height[i]:
                continue
            if sa[i - 1] < m < sa[i] or sa[i - 1] > m > sa[i]:
                a = ac.max(height[i - 1], height[i + 1])
                if a + 1 <= height[i] and a + 1 < ans:
                    ans = a + 1
        ac.st(ans if ans < inf else -1)
        return

    def cf_1526e(ac=FastIO()):
        """
        url: https://codeforces.com/contest/1526/problem/E
        tag: suffix_array|reverse_thinking|comb|construction
        """
        mod = 998244353
```

```
        n, k = ac.read_list_ints()
        sa = ac.read_list_ints()
        cb = Combinatorics(n + k, mod)
        rk = [0] * n
        for i in range(n):
            rk[sa[i]] = i
        m = 0
        for i in range(1, n):
            if sa[i] < n - 1 and sa[i - 1] < n - 1 and rk[sa[i] + 1] > rk[sa[i -
1] + 1]:
                m += 1
            elif sa[i - 1] == n - 1:
                m += 1
        ans = cb.comb(m + k, n)
        ac.st(ans)
        return
    # 原作者这里的也写反了，已更正
    def cf_873f(ac=FastIO()):
        """
        url: https://codeforces.com/contest/873/problem/F
        tag: suffix_array|reverse_thinking|lcp|prefix_sum
        """
        n = ac.read_int()
        a = ac.read_str()[::-1]
        s = ac.read_str()[::-1]
        sa, rk, height = SuffixArray().build([ord(w) - ord("a") for w in a], 26)

        left = [1] * n
        right = [n - 1] * n
        stack = []
        for i in range(1, n):
            while stack and height[stack[-1]] > height[i]:
                right[stack.pop()] = i - 1
            stack.append(i)

        stack = []
        for i in range(n - 1, -1, -1):
            while stack and height[stack[-1]] > height[i]:
                left[stack.pop()] = i + 1
            stack.append(i)

        pre = ac.accumulate([1 - int(s[i]) for i in sa])
        ans = max(height[i] * (pre[right[i] + 1] - pre[left[i] - 1]) for i in
range(n))
        # special case
        for i in sa:
            if s[i] == "0" and n - i > ans:
                ans = n - i
        ac.st(ans)
        return
```

8.automation

8.1 template:

```python
from collections import defaultdict

from queue import Queue
from typing import List, Dict, Iterable


class AhoCorasick(object):
    class Node(object):

        def __init__(self, name: str):
            self.name = name  # 节点代表的字符
            self.children = {}  # 节点的孩子，键为字符，值为节点对象
            self.fail = None  # failpointer，root的pointer为None
            self.exist = []  # 如果节点为单词结尾，存放单词的长度

    def __init__(self, keywords: Iterable[str] = None):
        """AC自动机"""
        self.root = self.Node("root")
        self.finalized = False
        if keywords is not None:
            for keyword in set(keywords):
                self.add(keyword)

    def add(self, keyword: str):
        if self.finalized:
            raise RuntimeError('The tree has been finalized!')
        node = self.root
        for char in keyword:
            if char not in node.children:
                node.children[char] = self.Node(char)
            node = node.children[char]
        node.exist.append(len(keyword))

    def contains(self, keyword: str) -> bool:
        node = self.root
        for char in keyword:
            if char not in node.children:
                return False
            node = node.children[char]
        return bool(node.exist)

    def finalize(self):
        """构建failpointer"""
        queue = Queue()
        queue.put(self.root)
        # 对树层次遍历
        while not queue.empty():
            node = queue.get()
            for char in node.children:
                child = node.children[char]
```

```
                f_node = node.fail
                # 关键点！需要沿着failpointer向上追溯直至根节点
                while f_node is not None:
                    if char in f_node.children:
                        # 如果该pointer指向的节点的孩子中有该字符，则字符节点的
failpointer需指向它
                        f_child = f_node.children[char]
                        child.fail = f_child
                        # 同时将长度合并过来，以便最后输出
                        if f_child.exist:
                            child.exist.extend(f_child.exist)
                        break
                    f_node = f_node.fail
                # 如果到根节点也没找到，则将failpointer指向根节点
                if f_node is None:
                    child.fail = self.root
                queue.put(child)
        self.finalized = True

    def search_in(self, text: str) -> Dict[str, List[int]]:
        """在一段文本中查找关键字及其开始位置（可能重复多个）"""
        result = dict()
        if not self.finalized:
            self.finalize()
        node = self.root
        for i, char in enumerate(text):
            matched = True
            # 如果当前节点的孩子中找不到该字符
            while char not in node.children:
                # failpointer为None，说明走到了根节点，找不到匹配的
                if node.fail is None:
                    matched = False
                    break
                # 将failpointer指向的节点作为当前节点
                node = node.fail
            if matched:
                # 找到匹配，将匹配到的孩子节点作为当前节点
                node = node.children[char]
                if node.exist:
                    # 如果该节点存在多个长度，则输出多个关键词
                    for length in node.exist:
                        start = i - length + 1
                        word = text[start: start + length]
                        if word not in result:
                            result[word] = []
                        result[word].append(start)
        return result


class TrieNode:
    def __init__(self):
        self.child = {}
        self.fail_to = None
        self.is_word = False
        '''
        下面节点值可以根据具体场景赋值
```

```
        '''
        self.str_ = ''
        self.num = 0


class AhoCorasickAutomation:
    def __init__(self):
        """
        Initialize your data structure here.
        """
        self.root = TrieNode()

    def build_trie_tree(self, word_lst):
        for word in word_lst:
            cur = self.root
            for i, c in enumerate(word):
                if c not in cur.child:
                    cur.child[c] = TrieNode()
                ps = cur.str_
                cur = cur.child[c]
                cur.str_ = ps + c
            cur.is_word = True
            cur.num += 1

    def build_ac_from_trie(self):
        queue = []
        for child in self.root.child:
            self.root.child[child].fail_to = self.root
            queue.append(self.root.child[child])

        while len(queue) > 0:
            cur = queue.pop(0)
            for child in cur.child.keys():
                fail_to = cur.fail_to
                while True:
                    if not fail_to:
                        cur.child[child].fail_to = self.root
                        break
                    if child in fail_to.child:
                        cur.child[child].fail_to = fail_to.child[child]
                        break
                    else:
                        fail_to = fail_to.fail_to
                queue.append(cur.child[child])

    def ac_search(self, str_):
        cur = self.root
        result = defaultdict(int)
        dct = {}  # 输出具体索引
        i = 0
        n = len(str_)
        while i < n:
            c = str_[i]
            if c in cur.child:
                cur = cur.child[c]
                if cur.is_word:
```

```
                temp = cur.str_
                result[temp] += 1
                dct.setdefault(temp, [])
                dct[temp].append(i - len(temp) + 1)

            '''
            处理所有其他长度公共字串
            '''
            fl = cur.fail_to
            while fl:
                if fl.is_word:
                    temp = fl.str_
                    result[temp] += 1
                    dct.setdefault(temp, [])
                    dct[temp].append(i - len(temp) + 1)
                fl = fl.fail_to
            i += 1

        else:
            cur = cur.fail_to
            if not cur:
                cur = self.root
                i += 1
    return result, dct
```

8.2 problem:

```
# Algorithm:
automaton|sub_sequence_automaton|suffix_automaton|palindrome_automaton
# Description: kmp|trie_like|word_count|text
'''
==================================LuoGu==================================
P3808（https://www.luogu.com.cn/problem/P3808）automaton|counter|trie_like
P3796（https://www.luogu.com.cn/problem/P3796）automaton|counter|trie_like
P5357（https://www.luogu.com.cn/problem/P5357）automaton|counter|trie_like
P5826（https://www.luogu.com.cn/problem/P5826）
sub_sequence_automaton|binary_search
P9572（https://www.luogu.com.cn/problem/P9572）
sub_sequence_automaton|binary_search
================================CodeForces================================
91A（https://codeforces.com/contest/91/problem/A）sub_sequence_automaton
1845C（https://codeforces.com/contest/1845/problem/C）sub_sequence_automaton
'''
```

```
import bisect
from itertools import permutations
from typing import List

from src.strings.automaton.template import AhoCorasick
from src.utils.fast_io import FastIO

class Solution:
    def __init__(self):
        return

    def lg_p5826(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P5826
        tag: sub_sequence_automaton|binary_search
        """
        _, n, q, m = ac.read_list_ints()
        ind = [[] for _ in range(m + 1)]
        lst = ac.read_list_ints()
        for i, num in enumerate(lst):
            ind[num].append(i)
        for _ in range(q):
            lst = ac.read_list_ints()[1:]
            i = 0
            for w in lst:
                j = bisect.bisect_left(ind[w], i)
                if j >= len(ind[w]):
                    ac.no()
                    break
                i = ind[w][j] + 1
            else:
                ac.yes()
        return
```

```python
    def lg_p9572(ac=FastIO()):
        """
        url: https://www.luogu.com.cn/problem/P9572
        tag: sub_sequence_automaton
        """
        n, m, c1, c2 = ac.read_list_ints()
        s = ac.read_list_ints()
        dct = dict()
        for i, w in enumerate(s):
            if w not in dct:
                dct[w] = []
            dct[w].append(i)
        t = [w for w in ac.read_list_ints() if w in dct]
        if not t:
            ac.lst([0, 0])
            return
        ans = 1
        i = dct[t[0]][0]
        for x in t[1:]:
            if dct[x][-1] > i:
                i = dct[x][bisect.bisect_right(dct[x], i)]
            else:
                ans += 1
                i = dct[x][0]
        ac.lst([c1 * len(t), c2 * ans])
        return

    def cf_91a(ac=FastIO()):
        """
        url: https://codeforces.com/contest/91/problem/A
        tag: sub_sequence_automaton
        """
        s = [ord(w) - ord("a") for w in ac.read_str()]
        t = [ord(w) - ord("a") for w in ac.read_str()]
        n, m = len(s), len(t)

        nxt = [-1] * 26 * n
        post = dict()
        for i in range(n - 1, -1, -1):
            post[s[i]] = i

        for i in range(n - 1, -1, -1):
            for j in post:
                nxt[i * 26 + j] = post[j]
            post[s[i]] = i

        if nxt[(n - 1) * 26 + t[0]] == -1:
            ac.st(-1)
            return

        i = nxt[(n - 1) * 26 + t[0]]
        ans = 1
        for j in t[1:]:
            k = nxt[i * 26 + j]
            if k == -1:
                ac.st(-1)
```

```
                return
            if k <= i:
                ans += 1
            i = k
        ac.st(ans)
        return

    def cf_1845c(ac=FastIO()):

        """
        url: https://codeforces.com/contest/1845/problem/C
        tag: sub_sequence_automaton
        """

        def check():
            i = 0
            ll, rr = int(s1[i]), int(s2[i])
            pre = set()
            for w in s:
                if ll <= int(w) <= rr:
                    pre.add(int(w))
                if len(pre) == rr - ll + 1:
                    pre = set()
                    i += 1
                    if i == m:
                        ac.no()
                        return
                    ll, rr = int(s1[i]), int(s2[i])
            ac.yes()
            return

        for _ in range(ac.read_int()):
            s = ac.read_str()
            m = ac.read_int()
            s1 = ac.read_str()
            s2 = ac.read_str()
            check()
        return
```