Дизајн на алгоритми

Стрингови

Стефан Милев | 206055

1. Балансиран бит стринг

Опис

Бит стринг е стринг што се состои само од 0 и 1. Бит стрингот е ***k*** - балансиран ако секој подстринг од големина ***k*** има еднаков број на 0 и 1 карактери.

Даден е позитивен цел број ***k*** и стринг ***s*** составен од карактерите 0, 1 и ?. Треба да се одреди дали може да се направи ***k*** - балансиран бит стринг со заменување на сите ? карактери во ***s*** со 0 или 1.

Стринг ***a*** е подстринг на стринг ***b*** ако ***a*** може да се добие од ***b*** со бришење на 0 или повеќе карактери од почетокот и крајот на стрингот.

Влез

Влезот е во формат:

n
k
s

каде што ***n*** е должината на стрингот, ***k*** е должината на подстрингот, ***s*** е стрингот.

Излез

YES - ако е возможно да се замени секој ? карактер со 0 или 1 така што добиениот стринг е ***k*** - балансиран,
NO - инаку.

Ограничувања

s содржи само 0, 1 или ?
$2 \leq k \leq n \leq 3 \cdot 10^5$

Тест примери

#	Влез	Излез
1	6 4 100110	YES
2	3 2 1?1	YES
3	3 2 1?0	NO
4	4 4 ????	YES
5	7 4 1?0??1?	YES
6	10 10 11??11??11	NO
7	4 2 1??1	NO
8	4 4 ?0?0	YES
9	6 2 ????00	NO
10	5 2 1?0?1	NO
11	26 2 1111?0?11?1???1010??10?0?0	NO

Тест примери

#	Влез	Излез
12	7 2 ???????	YES
13	31 2 1??1??11?1?1??1?1?11???11??1???	NO
14	21 2 01?101?0010?0??1000??	NO
15	114 2 1?111???11???????1???11?1?111111?11?1111?11?1???111????????1?? ??1?1??1111?1????11???11?11?11?????11111??1?1??1??1?	NO
16	6 2 ?0?00?	NO
17	10 2 ??????????	YES
18	16 2 11111?????10111?	NO
19	16 2 ?101110??1?1??1?	NO
20	80 2 ????????0???0?0???0?0?0?0???0?0???0?????0?0???0???0?????0??? ??????0???0???0?0???	YES

Решение

Нека имаме стринг ***s[0 … n - 1]*** и бит стринг ***t[0 … k - 1]*** кој произлегува од ***s***.

За секое ***i*** важи: ***0 ≤ i < n - k***, $t_i = t_i + k$. Подстринговите со должина ***k*** почнувајќи од индекс ***i*** и ***i + 1*** ги делат истите ***k - 1*** карактери $t_{i+1} \dots t_{i+k-1}$. За да биде ист бројот на единици, карактерите t_{i+1} и t_{i+k-1} мора и двете да бидат 1 или и двете да бидат 0.

$t_i = t_j$ ако ***i ≡ j (mod k)***. За секое ***i*** во опсегот ***0 ≤ i < k*** треба да се одреди дали сите s_j за кои важи ***i ≡ j (mod k)*** може да се додели истата вредност (или 0 или 1). Карактерот t_i мора да биде 0 доколку барем еден s_j е 0, а мора да биде 1 доколку барем еден s_j е 1, а може да биде или 0 или 1 доколку сите s_j се ?.

Доволно е да се провери дали бит стрингот ***t[0 … k - 1]*** има точно половина карактери 1 (или половина карактери 0). Доколку има неодлучени карактери (места со ?), тогаш треба да се провери дали бројот на 0 и 1 карактери не надминуваат ***k / 2*** (должината на бит стрингот), во кој случај може ? карактерите да бидат сменети со 0 или 1 за да има изедначен број на 0 и 1 карактери.

Дадено е Python решение.

Комплексност

Комплексноста на алгоритамот е ***O(n)***.

За секој t_i во бит стрингот, се проверува дали и сите s_j за кои важи ***i ≡ j (mod k)*** дали се истиот карактер. Се итерираат сите t_i во ***k*** итерации, додека се итерираат сите s_j во ***n / k*** итерации, од каде доаѓа дека комплексноста е ***O(n)***.

Решение - Python

```python
def main():
    n = int(input())
    k = int(input())
    s = input()

    zeros = 0
    ones = 0

    for i in range(k):
        temp = -1

        for j in range(i, n, k):
            if s[j] != '?':
                if temp != -1 and int(s[j]) != temp:
                    return 'NO'

                temp = int(s[j])

        if temp == 0:
            zeros += 1
        elif temp == 1:
            ones += 1

    return 'YES' if max(zeros, ones) <= k // 2 else 'NO'

if __name__ == '__main__':
    print(main())
```

2. Добивање стринг

Опис

Дадени се 3 стрингови ***s***, ***t***, ***z*** така што ***s*** и ***t*** се составени од мали латинични букви, а ***z*** е празен. Целта е ***z*** да стане еднаков со ***t***. Можната акција е додавање било која подниза од ***s*** на крај на ***z***. Стрингот ***s*** не се менува.

Се бара најмалиот број на операции за да се претвори стрингот ***z*** во ***t***.

Подниза е секвенца добиена со бришење на нула или повеќе елементи без менување на редоследот на останатите.

Влез

Влезот е во формат:

s
t

Излез

Минималниот број на операции за да се претвори ***z*** во ***t*** доколку е возможно, инаку -1.

Ограничувања

$1 \leq |s| \leq 10^5$
$1 \leq |t| \leq 10^5$

Тест примери

#	Влез	Излез
1	aabce ace	1
2	abacaba aax	-1
3	ty yyt	3
4	a aaaaaaaaaaa	11
5	cba abcabcabcabcabcabcabc	15
6	bvdhsdvlbelrivbhxbhie x	1
7	abacabadabacaba abacabadabacaba	1
8	abacabadabacaba baabaabcaa	1
9	aabab aaababbbaaaabbab	5
10	t y	-1
11	u u	1
12	abcdefghijk kjihgfedcba	11
13	abcdefghijk kjihgfdecba	10
14	abb ababbbbb	4
15	a aaab	-1

Тест примери

#	Влез	Излез
16	bab aaa	3
17	abbabb ba	1
18	bbbbbba abbb	2
19	bbaa bbbabab	4
20	aaab bbabbaba	6

Решение

Ако има карактер во ***t*** кој го нема во ***s***, може да се заклучи дека нема решение.

Во обратниот случај, се пополнува низа ***dp*** на следниов начин:
dp[i][j] = минимум индекс ***x*** од ***i*** до **|*s*|** така што $s_x = j$ или бесконечно доколку не постои таков.

Нека ***res = 1***, ***pos = 0***

Нека имаме стринг $z[t_0 \dots t_{i-1}]$ и нека е s_{pos} последниот земен карактер. Тогаш:

1. ако ***dp[pos][i] ≠ бесконечно***, тогаш ***i += 1*** и ***pos = dp[pos + 1][i]***
2. инаку, ***pos = 0*** и ***res += 1***

Дадено е Python решение.

Комплексност

Комплексноста на алгоритамот е ***O(|S| + |T|)***.

Низата ***dp*** се пополнува во ***O(|S|)*** итерации, додека главниот циклус го наоѓа решението во најмногу ***O(|T|)*** итерации.

Решение - Python

```python
def main():
    s = input()
    t = input()

    dp = [[2 ** 63 for _ in range(26)] for _ in range(len(s) + 1)]

    for i in range(len(s) - 1, -1, -1):
        for j in range(26):
            dp[i][j] = dp[i + 1][j]
        dp[i][ord(s[i]) - ord('a')] = i

    res = 1
    pos = 0

    for i in range(len(t)):
        if pos == len(s):
            pos = 0
            res += 1

        if dp[pos][ord(t[i]) - ord('a')] == 2 ** 63:
            pos = 0
            res += 1

        if dp[pos][ord(t[i]) - ord('a')] == 2 ** 63 and pos == 0:
            return -1

        pos = dp[pos][ord(t[i]) - ord('a')] + 1

    return res


if __name__ == '__main__':
    print(main())
```