contest 10 题解

未分类

D题 Devu and binary String

> - 题意：给你一个01串，你需要改动最少的字符使得这个串不存在k个连续相同的字符，每一次改动可以将0变1,1变0

此题是一个贪心+模拟的题，每当发现一个长度大于k的连续区间，最优的方法应该是每k+1个位置转变一个字符，比如11111，k=2，应该变成11011,
1111111，应该变成1101101，但是当这个长度能被(k+1)整除的时候需要特殊处理，因为11111100，k=2，如果每三个变一个，就会变成11011000,造成最后有3个0，应对的方法是将所有的改变都往前挪一个，即10110100
多组数据一定要记得初始化，被坑了好几发！！
代码

```
#include <bits/stdc++.h>
using namespace std;

const int N = 1000010;
char s[N], s1[N], s2[N];
int cnt[2];
int k;
int ret;

void change(int i) {
    int len = cnt[(s[i]-'0')];
    if (len > k) {
        ret += len / (k + 1);
        if (len % (k + 1) == 0) {
            for (int j = i - 1; j >= i - len + 1; j -= (k + 1)) {
                s[j] = '1' - s[j] + '0';
            }
        } else {
```

```
                for (int j = i - len + 1 + k; j <= i; j += (k + 1)) {
                    s[j] = '1' - s[j] + '0';
                }
            }
        }
    }

    int main() {
        int T, n;
        scanf("%d", &T);
        while (T--) {
            scanf("%d%d", &n, &k);
            scanf("%s", s);
            s1[n] = 0;
            s2[n] = 0;
            memset(cnt, 0, sizeof(cnt));//忘了初始化！！！坑
            for (int i = 0; i < n; i++) {
                s1[i] = (i&1) ? '1' : '0';
                s2[i] = (i&1) ? '0' : '1';
            }
            int cost0 = 0, cost1 = 0;
            for (int i = 0; i < n; i++) {
                if (i & 1) {
                    cost0 += (s[i] == '0');
                    cost1 += (s[i] == '1');
                } else {
                    cost0 += (s[i] == '1');
                    cost1 += (s[i] == '0');
                }
            }
            int mi = min(cost0, cost1);
            if (k == 1) {
                printf("%d\n", mi);
                if (cost0 == mi) {
                    printf("%s\n", s1);
                } else {
                    printf("%s\n", s2);
                }
            } else {
                ret = 0;
                cnt[s[0]-'0']++;
                for (int i = 1; i < n; i++) {
                    if (s[i] == s[i - 1]) {
                        cnt[s[i]-'0']++;
                    } else {
```

```
                        change(i - 1);
                        cnt[s[i]-'0'] = 1;
                        cnt[!(s[i]-'0')] = 0;
                    }
                }
                change(n - 1);
                printf("%d\n", ret);
                printf("%s\n", s);
            }
        }
        return 0;
    }
```

E 题Chef and Strings

> - 一共有四种字符构成的一个字符串，现在每次需要询问一段区间内满足开头结尾条件的子串的数量

本题是一个典型的前缀和统计的题目
设cnt[i][j][k]前缀k里面，i字符开头，j字符结尾的子串的数量
sum[i][j]表示前缀j里面i字符的数量
那么对于一段区间L R询问a开头b结尾的子串数量就等价于cnt数组的前缀相减，cnt[a][b][R]-cnt[a][b][L-1]
，但是这样减去之后a在0->L-1位置之间, b在L->R之间的的子串数量并没有减去，所以还要减去这个

```
#include <bits/stdc++.h>
using namespace std;

const int N = 1000010;
char s[N];

inline int mp(char ch) {
    if (ch == 'c') return 0;
    if (ch == 'h') return 1;
    if (ch == 'e') return 2;
    return 3;
```

```
}

long long cnt[4][4][N];
long long sum[4][N];

void init() {
    int n = strlen(s + 1);
    for (int i = 1; i <= n; i++) {
        for (int f = 0; f < 4; f++) {
            sum[f][i] = sum[f][i - 1];
            for (int e = 0; e < 4; e++) {
                cnt[f][e][i] = cnt[f][e][i - 1];
            }
        }
        sum[mp(s[i])][i]++;
        for (int j = 0; j < 4; j++) {
            cnt[j][mp(s[i])][i] += sum[j][i];
        }
    }
}

int main () {
    scanf("%s", s + 1);
    init();
    int q, L, R;
    char a[2], b[2];
    scanf("%d", &q);
    while (q--) {
        scanf("%s%s%d%d", a, b, &L, &R);//单个字符尽量都用字符串去读入
        int x = mp(a[0]);
        int y = mp(b[0]);
        printf("%lld\n", cnt[x][y][R] - cnt[x][y][L-1] - sum[x][L - 1] *
(sum[y][R]-sum[y][L-1]));
    }
    return 0;
}
```