3 白色光 2 (White Light 2)

Author : 蜂矢 倫久 (Mitsubachi)

小課題 1

点灯しているライトの数は 3 の倍数である．したがって，$N = 3$ のときには点灯しているライトの数は 0 または 3 である．

点灯しているライトの数を 0 にするために必要な金額の最小値は $A \leqq B$ のとき $3A$ で，そうでないときは $3B$ である．点灯しているライトの数を 3 にするとき，ライトの色の並びは ‘RGB’ となる．したがって，S の i 文字目と ‘RGB’ の i 文字目が異なるような i の数に C をかけた値が必要な金額の最小値である．

答えは上で求めた 2 つの値のうち小さい方である．$O(N)$ で答えが求まる．

```cpp
#include <iostream>
#include <vector>
using namespace std;
string rgb = "RGB";
int main(){
    int n;
    string s;
    long long a, b, c;
    cin >> n >> s >> a >> b >> c;

    long long ans = 0;
    for(int i = 0; i < 3; i++){
        if(s[i] != rgb[i]){
            ans += c;
        }
    }
    ans = min(ans, min(a, b) * 3);
    cout << ans << endl;
}
```

小課題 2

ライトをすべて消灯させるのに必要な金額の最小値は小課題 1 のようにして求められる．いくつかのライトが点灯している状態で操作が終わる場合を考える．

$1 \leqq L \leqq R \leqq N$，$R - L + 1$ を 3 の倍数として，操作を終えたときに点灯しているようなライトの範囲を左から L 番目から R 番目までと固定する．このとき，ライトを消灯させるのに必要なコストは $(L-1)A + (N-R)B$ である．

また L 番目から R 番目のライトのうち赤色へ点灯し直す必要があるものの個数は $L \leqq i \leqq R$，$i - L$ を 3 で割ったあまりが 0 となる i の中でライト i が赤色でないものの個数である．同様にして，緑色へ点灯し直す必要があるものと青色へ点灯し直す必要があるものの個数も分かる．

これより，操作を終えたときに点灯しているようなライトの範囲を左から L 番目から R 番目までと固定したときの必要な金額の最小値は $O(N)$ で求められる．点灯しているライトの範囲として考えられるものは $O(N^2)$ 通りあるので全体で $O(N^3)$ で答えが求まる．

```
    long long ans = 1e18;

    for(int i = 0; i <= n; i++){
        for(int j = i; j <= n; j++){
            // ライト i からライト j-1 まで点灯させる ただし i = j のときはすべて消灯する
            if((j - i) % 3 != 0){
                continue;
            }
            long long cost_now = a * i + b * (n - j);
            for(int k = i; k < j; k++){
                if(s[k] != rgb[(k - i) % 3]){
                    cost_now += c;
                }
            }
            ans = min(ans, cost_now);
        }
    }
```

小課題 3

$1 \leqq L \leqq R \leqq N-3$，$R-L+1$ を 3 の倍数として，操作を終えたときに点灯しているようなライトの範囲を左から L 番目から R 番目までと固定したときの，必要な金額の最小値が分かっているとする．このとき，操作を終えたときに点灯しているようなライトの範囲を左から L 番目から $R+3$ 番目までと固定したときの，必要な金額の最小値を求める．

ライトを消灯させるのに必要なコストは $3B$ 減少する．また L 番目から $R+3$ 番目のライトのうち点灯し直す必要があるものの個数は，L 番目から R 番目のライトのうち点灯し直す必要があるものの個数に $R+1$ 番目から $R+3$ 番目までの中で点灯し直す必要があるものの個数を足したものである．したがって，$O(1)$ で左から L 番目から $R+3$ 番目までと固定したときの，必要な金額の最小値も分かる．

よって操作を終えたときに点灯しているようなライトの範囲を L の小さい順に，L が同じときは R が小さい順に固定したときの必要な金額の最小値を求めることで全体で $O(N^2)$ で答えが求まる．

```
    long long ans = 1e18;

    for(int i = 0; i <= n; i++){
        long long cost_now = a * i;
        for(int j = i; j < n; j++){
            // ライト i からライト j までを点灯させる
            // cost_now: B 円を払う操作以外の操作に必要な金額
            if(s[j] != rgb[(j - i) % 3]){
                cost_now += c;
            }
            if((j - i + 1) % 3 == 0){
                ans = min(ans, cost_now + b * (n - 1 - j));
            }
        }
    }

    // ライトをすべて消灯させる場合
    ans = min(ans, min(a, b) * n);
```

小課題 3 別解

$1 \leqq L \leqq R \leqq N$, $R - L + 1$ を 3 の倍数として，操作を終えたときに点灯しているようなライトの範囲を左から L 番目から R 番目までと固定する．このとき，ライトを消灯させるのに必要なコストは $(L-1)A + (N-R)B$ である．

また L 番目から R 番目のライトのうち操作の前後で共に赤色となる個数は $L \leqq i \leqq R$, $i - L$ を 3 で割ったあまりが 0 となる i の中でライト i が赤色であるものの個数である．同様にして，緑色と青色の場合の個数も分かる．これらの個数は前もって累積和を取ることで $O(1)$ で入手できる．

これより，操作を終えたときに点灯しているようなライトの範囲を左から L 番目から R 番目までと固定したときの必要な金額の最小値は $O(1)$ で求められる．全体で $O(N^2)$ で答えが求まる．

```
    // 累積和を取る
    vector<vector<int>> sum(n, vector<int>(3, 0));
    for(int i = 0; i < n; i++){
        for(int j = 0; j < 3; j++){
            if(s[i] == rgb[j]){
                sum[i][j]++;
            }
            if(i > 2){
                sum[i][j] += sum[i - 3][j];
            }
        }
    }

    long long ans = min(a, b) * n;
    for(int i = 0; i < n; i++){
        for(int j = i; j < n; j++){
            if((j - i + 1) % 3 != 0){
                continue;
            }
            // 何個点灯し直すか
            int change = j - i + 1;

            // 最後赤色になっていて，かつ最初から赤色になっている数を引く
```

```
            change -= sum[i + (j - i) / 3 * 3][0];
            if(i > 2){
                change += sum[i - 3][0];
            }

            // 緑色や青色についても同様
            change -= sum[i + 1 + (j - i - 1) / 3 * 3][1];
            if(i > 1){
                change += sum[i - 2][1];
            }

            change -= sum[i + 2 + (j - i - 2) / 3 * 3][2];
            if(i > 0){
                change += sum[i - 1][2];
            }

            ans = min(ans, a * i + b * (n - 1 - j) + c * change);
        }
    }
```

小課題 4

消灯させる操作を 1 回でも行うと 10^9 円が少なくとも必要だが，すべてのライトを点灯し直す操作のみで条件を満たすようにしても高々 N 円で操作は完了する．よって，ライトを点灯し直す操作のみを考えればよい．

小課題 2 で述べたように，赤色へ点灯し直す必要があるものの個数は i を 3 で割ったあまりが 0 となる i の中でライト i が赤色でないものの個数である．同様にして，緑色へ点灯し直す必要があるものと青色へ点灯し直す必要があるものの個数も分かる．

これより，点灯し直す必要のあるライトの個数が $O(N)$ で分かるので，全体で $O(N)$ で答えが求まる．

```
    long long ans = 0;
    for(int i = 0; i < n; i++){
        if(s[i] != rgb[i % 3]){
            ans += c;
        }
    }
```

小課題 5

N を 3 で割ったあまりを x として，条件を満たすとき点灯しているライトの数は 3 の倍数なので x 回は消灯させる操作を行う必要がある．その後ライトを点灯し直す操作を行うとして条件を満たすためには高々 $N-x$ 回で十分である．よって必要な金額の最小値は $x \times 10^9 + N - x$ である．逆に，消灯させる操作を x 回より多く行う場合の必要な金額の最小値は少なくとも $(x+3) \times 10^9$ である．

これより消灯させる操作の回数は x 回としてよい．よって操作を終えたときに点灯しているようなライトの範囲は $x+1$ 通りのみを考えればよく，全体で $O(N)$ で答えが求まる．実装の際，左から消灯する個数と右から消灯する個数を 0 から 2 個まで全探索しても十分高速である．

```
long long ans = 1e18;

for(int i = 0; i < 3; i++){
    for(int j = 0; j < 3; j++){
        // 先頭から i 個のライト，末尾から j 個のライトを消灯させた場合
        if((n - i - j) % 3 != 0 || n < i + j){
            continue;
        }
        int left = i, right = n - j;
        long long cost_now = i * a + j * b;
        for(int k = left; k < right; k++){
            if(s[k] != rgb[(k - left) % 3]){
                cost_now += c;
            }
        }
        ans = min(ans, cost_now);
    }
}
```

満点

操作を終えるときに点灯しているライトの範囲が R 番目までとして，R の小さい順に必要な金額の最小値を求める．右から消灯させる操作に必要な金額は $(N-R)B$ である．よって，1 番目から R 番目までの操作について必要な金額の最小値を考えればよい．

$R > 3$ のとき，1 番目から $R-3$ 番目までの操作について必要な金額の最小値を x とする．$R-2$ 番目から R 番目を点灯させる場合，$R-2$ 番目から R 番目で点灯し直すライトの数を y として必要な金額の最小値は $x+yC$ である．R 番目まで左から消灯させるとき，必要な金額は RA である．

よって，1 番目から R 番目までの操作について必要な金額の最小値はこの 2 つの値のうち小さい方である．これより，点灯しているライトの範囲が R 番目までとして，1 番目から R 番目までの操作について必要な金額の最小値は R の小さい順に $O(N)$ ですべて求まる．

全体で $O(N)$ で答えが求まる．

```
    long long ans = min(a, b) * n;

    // i 番目までの操作について必要な金額の最小値を opt[i % 3] に保持し，更新していく
    vector<long long> opt = {a, 2 * a, 0};

    for(int i = 2; i < n; i++){
        if(s[i - 2] != 'R'){
            opt[i % 3] += c;
        }
        if(s[i - 1] != 'G'){
            opt[i % 3] += c;
        }
        if(s[i] != 'B'){
            opt[i % 3] += c;
        }

        opt[i % 3] = min(opt[i % 3], (i + 1) * a);
        ans = min(ans, opt[i % 3] + (n - 1 - i) * b);
    }
```

別解

各ライトについて操作を終了したときの状態は次のいずれかである.

- 状態 1: 赤色に点灯している.
- 状態 2: 緑色に点灯している.
- 状態 3: 青色に点灯している.
- 状態 4: A 円を払う操作により消灯している.
- 状態 5: B 円を払う操作により消灯している.

ここで，条件を満たすとき，ライトの状態について以下のように考えられる.

- 状態 1 のライトについて，そのライトは左端または左隣のライトは状態 3 か状態 4 である.
- 状態 2 のライトについて，左隣のライトは状態 1 である.
- 状態 3 のライトについて，左隣のライトは状態 2 である.
- 状態 4 のライトについて，そのライトは左端または左隣のライトは状態 4 である.
- 状態 5 のライトについて，そのライトは左端または左隣のライトは状態 3 か状態 5 である.

よって，ライト i が状態 j である条件を満たすライトの状態において，ライト 1 からライト i までに関する操作に必要な金額の最小値を求める動的計画法を考えることでこの問題は $O(N)$ で解くことができる.

```
    // dp[i][j]: ライト i が状態 j であり条件を満たしているときの金額の最小値
    vector<vector<long long>> dp(n + 1, vector<long long>(5,1e18));
    dp[0][3] = 0;

    for(int i = 0; i < n; i++){
        for(int j = 0; j < 3; j++){
            int k = (j + 2) % 3;

            if(s[i] != rgb[j]){
                dp[i + 1][j] = min(dp[i + 1][j], dp[i][k] + c);
                if(j == 0){
                    dp[i + 1][j] = min(dp[i + 1][j], dp[i][3] + c);
                }
            }
            else{
```

```
                dp[i + 1][j] = min(dp[i + 1][j], dp[i][k]);
                if(j == 0){
                    dp[i + 1][j] = min(dp[i + 1][j], dp[i][3]);
                }
            }
        }

        dp[i + 1][3] = min(dp[i + 1][3], dp[i][3] + a);

        for(int j = 2; j < 5; j++){
            dp[i + 1][4] = min(dp[i + 1][4], dp[i][j] + b);
        }
    }

    long long ans = 1e18;
    for(int j = 2; j < 5; j++){
        ans = min(ans, dp[n][j]);
    }
```