Wojciech Rytter
Treść zadania, Opracowanie

Jan Kanty Milczek
Program

Dostępna pamięć: 128 MB. *OI, etap II, dzień drugi, 12.02.2015*

Kurs szybkiego czytania

Bajtazar zapisał się na kurs szybkiego czytania, na którym nauczył się wielu ćwiczeń poprawiających spostrzegawczość. Jego ulubionym ćwiczeniem jest znajdowanie wzorca w ciągu symboli. Aby przygotować ćwiczenie, Bajtazar wykorzystuje komputer do wygenerowania bardzo długiego ciągu zer i jedynek. Wybiera liczby n, a, b, p takie, że n i a są względnie pierwsze, a komputer generuje ciąg $c_0, c_1, \ldots, c_{n-1}$, gdzie $c_i = 0$ wtedy i tylko wtedy, gdy $(ai + b) \bmod n < p$. Następnie Bajtazar wymyśla drugi, krótszy ciąg m symboli $w_0, w_1, \ldots, w_{m-1}$. Jego zadaniem jest jak najszybsze znalezienie wszystkich wystąpień krótszego ciągu w ciągu wygenerowanym przez komputer. Ciebie poprosił o pomoc w napisaniu programu, który sprawdzi, czy znalazł wszystkie.

Wejście

Pierwszy wiersz standardowego wejścia zawiera pięć liczb całkowitych n, a, b, p i m ($2 \leqslant n \leqslant 1\,000\,000\,000$, $1 \leqslant p, a, b, m < n$, $1 \leqslant m \leqslant 1\,000\,000$) pooddzielanych pojedynczymi odstępami. Liczby a i n są względnie pierwsze. W drugim wierszu zapisane jest jedno m-literowe słowo $w_0, w_1, \ldots, w_{m-1}$ złożone z symboli `0` *i* `1`.

Istnieją następujące, rozłączne grupy testów:

- *w testach wartych 8% punktów zachodzi warunek $n \leqslant 1000$;*
- *w innych testach wartych 8% punktów zachodzi warunek $n \leqslant 1\,000\,000$;*
- *w jeszcze innych testach wartych 66% punktów zachodzi warunek $m \leqslant 1000$.*

Wyjście

Pierwszy i jedyny wiersz standardowego wyjścia powinien zawierać liczbę całkowitą, będącą liczbą wystąpień ciągu $w_0, w_1, \ldots, w_{m-1}$ w ciągu $c_0, c_1, \ldots, c_{n-1}$.

Przykład

Dla danych wejściowych:

```
9 5 6 4 3
101
```

poprawnym wynikiem jest:

```
3
```

Wyjaśnienie do przykładu: *Dla $n = 9$, $a = 5$, $b = 6$ i $p = 4$ komputer wygeneruje ciąg zgodnie z poniższą tabelką. Ciąg* `101` *występuje w wygenerowanym ciągu trzy razy.*

i	0	1	2	3	4	5	6	7	8
$ai + b$	6	11	16	21	26	31	36	41	46
$(ai + b) \bmod n$	6	2	7	3	8	4	0	5	1
c_i	1	0	1	0	1	1	0	1	0

Testy „ocen”:

`1ocen:` *szukamy wystąpień ciągu* `0010` *w ciągu* `10010000100100100100`*;*

`2ocen:` *szukamy wystąpień ciągu* `00000` *w ciągu* `00000001000000010000000000000001`*;*

`3ocen:` $n = 1\,000\,000\,000$*,* $m = 1\,000\,000$*. Szukamy wystąpień ciągu* `011...11` *(cyfra 0 oraz same jedynki) w ciągu* `00...0011...110` *(499 999 999 zer, 500 000 000 jedynek, 1 zero).*

Rozwiązanie

Zadanie to, pozornie natury tekstowej, należy zakwalifikować raczej jako teoriolicz-bowe. W zadaniu mamy ciągi $\mathbf{c}$ oraz w, a naszym celem jest zliczyć wystąpienia ciągu w (wzorca) w ciągu $\mathbf{c}$ (tekście). Problemem jest duży rozmiar $\mathbf{c}$. Rozwiązanie otrzymamy, wykorzystując arytmetykę postępów arytmetycznych i przedziałów na cyklu.

Niech $\oplus$ oznacza operację dodawania modulo n. Powiemy, że wzorzec w występuje cyklicznie w tekście $\mathbf{c}$ na pozycji i, gdy

$$w[k] = \mathbf{c}[i \oplus k] \quad \text{dla wszystkich } k = 0, \ldots, m-1.$$

Wystąpienie cykliczne, które nie jest standardowym wystąpieniem, nazywamy *nadmiarowym*.

W przykładzie z treści zadania w występuje w $\mathbf{c}$ trzy razy, natomiast cyklicznie występuje cztery razy. Wystąpienie cykliczne zaczynające się na przedostatniej pozycji jest nadmiarowe.

Definiujemy

$$\mathbf{P}(k) \;=\; \{\, i \;:\; w[k] = \mathbf{c}[i \oplus k] \,\}.$$

Obserwacja 1. Zbiór pozycji wystąpień cyklicznych w jest równy $\bigcap_{k=0}^{m-1} \mathbf{P}(k)$.

Zgrubnie nasz algorytm wygląda następująco.

1: **Algorytm**
2: Oblicz $\bigcap_{k=0}^{m-1} \mathbf{P}(k)$;
3: *nadmiar* := liczba nadmiarowych wystąpień cyklicznych;
4: **return** $|\bigcap_{k=0}^{m-1} \mathbf{P}(k)| - \mathit{nadmiar}$;

Reprezentacja zbiorów wystąpień

Teraz zajmiemy się taką reprezentacją zbiorów $\mathbf{P}(k)$, by pojedyncze przecięcia tych zbiorów (w odpowiedniej kolejności) i liczenie nadmiaru były operacjami wykonywalnymi możliwie szybko.

Zdefiniujmy $v_i = (a \cdot i + b) \mod n$; wówczas $\mathbf{c}[i] = 0$ wtedy i tylko wtedy, gdy $v_i < p$. Przypomnijmy, że a oraz n są względnie pierwsze.

Fakt 1. *Liczby* $v_0, \ldots, v_{n-1}$ *są parami różne.*

Dowód: Załóżmy przez sprzeczność, że tak nie jest. To znaczy, że istnieją indeksy $0 \leqslant i < j < n$, takie że $v_i = v_j$. Wtedy $v_i = v_j \Leftrightarrow n|(a \cdot j + b) - (a \cdot i + b) \Leftrightarrow n|a(j-i)$. Skoro a i n są względnie pierwsze, to $n|j-i$, ale $0 < j - i < n$. Otrzymaliśmy sprzeczność dowodzącą prawdziwości faktu. ■

Skoro liczb $v_0, \ldots, v_{n-1}$ jest n, przyjmują wartości od 0 do $n-1$ i są wszystkie różne, to znaczy, że są permutacją liczb od 0 do $n-1$. Nie jest to jednak zupełnie dowolna permutacja. Aby to zauważyć, zastanówmy się, jak wygląda permutacja odwrotna do v, tzn. ciąg $\mathbf{C}_i$ oznaczający pozycję, na której w ciągu v stoi wartość i ($v_{\mathbf{C}_i} = i$).

Odwrotnością a modulo n nazywamy taką liczbę r, że $a \cdot r \equiv 1 \pmod{n}$. Jeśli a jest względnie pierwsze z n, to jest ona wyznaczona jednoznacznie. Odwrotność a modulo n oznaczamy przez $a^{-1} \bmod n$. Można ją wyznaczyć w czasie $O(\log n)$ za pomocą rozszerzonego algorytmu Euklidesa.

Fakt 2. $\mathbf{C}_k = ((k-b) \cdot a^{-1}) \bmod n$*, a zatem ciąg* $\mathbf{C}$ *jest cyklicznym ciągiem arytmetycznym modulo* n *o różnicy* $a^{-1} \bmod n$.

Dowód: $\mathbf{C}_k$ jest równe takiej wartości i, dla której $ai + b \equiv k \pmod{n}$. Szukana wartość i to rzeczywiście $((k-b) \cdot a^{-1}) \bmod n$. To oznacza, że $\mathbf{C}_{k \oplus 1} - \mathbf{C}_k = a^{-1} \bmod n$, czyli $\mathbf{C}$ jest cyklicznym ciągiem arytmetycznym modulo n. ■

Przykład 1. Przyjrzyjmy się przykładowi z treści zadania. W tym przypadku ciąg v to ciąg

$$6,\ 2,\ 7,\ 3,\ 8,\ 4,\ 0,\ 5,\ 1.$$

Ciąg $\mathbf{C}$ jest cyklicznym ciągiem arytmetycznym modulo 9 o różnicy $r = 5^{-1} \bmod 9 = 2$:

$$\underline{6 \to 8 \to 1 \to 3} \to \overline{5 \to 7 \to 0 \to 2 \to 4}.$$

Zauważmy, że $4 + 2 = 6$ (pierwszy element cyklu jest następnikiem ostatniego).

Na pozycjach w $\mathbf{c}$ odpowiadających pierwszym czterem elementom ciągu $\mathbf{C}$ są symbole 0, a na kolejnych pięciu symbole 1. Generowany tekst $\mathbf{c}$ to

$$1\,0\,1\,0\,1\,1\,0\,1\,0.$$

Jeśli A jest podzbiorem $\{0, \ldots, n-1\}$ i $k \in \{0, \ldots, n-1\}$, to oznaczymy

$$A \ominus k = \{\, a \oplus (-k) \,:\, a \in A \,\}.$$

Analiza ciągu $\mathbf{C}$ prowadzi nas do eleganckiej charakteryzacji zbiorów $\mathbf{P}(k)$.

Fakt 3. $\mathbf{P}(k)$ *jest ciągiem arytmetycznym modulo* n *o różnicy* $a^{-1} \bmod n$.

Dowód: Z definicji zbioru $\mathbf{P}(k)$ mamy:

$$\mathbf{P}(k) = \{\, i \,:\, w[k] = \mathbf{c}[i] \,\} \ominus k.$$

Dla $j \in \{0, 1\}$ oznaczmy przez $S^{(j)} = \{\, i \,:\, \mathbf{c}[i] = j \,\}$ zbiór pozycji tekstu $\mathbf{c}$, na których znajduje się cyfra j. Wówczas wzór na $\mathbf{P}(k)$ przyjmuje postać $\mathbf{P}(k) = S^{(w[k])} \ominus k$.

Wiemy, że $S^{(0)}$ jest prefiksem ciągu $\mathbf{C}$, a $S^{(1)}$ jest sufiksem ciągu $\mathbf{C}$. Każdy z nich jest więc ciągiem arytmetycznym modulo n o różnicy $a^{-1} \bmod n$, więc po odjęciu od wszystkich jego wyrazów tej samej liczby k wciąż jest takim właśnie ciągiem. ■

Na podstawie dowodu faktu łatwo podać konkretne wzory na długość oraz pierwszy wyraz ciągu arytmetycznego reprezentowanego przez $\mathbf{P}(k)$. Pozostawiamy to Czytelnikom jako ćwiczenie.

Przykład 2. Wróćmy jeszcze raz do przykładu z treści zadania. Mamy:

$$\mathbf{P}(0) = \{\, i \;:\; \mathbf{c}[i] = w[0] = 1 \,\} = \{\, 5, 7, 0, 2, 4 \,\}$$
$$\mathbf{P}(1) = \{\, i \;:\; \mathbf{c}[i \oplus 1] = w[1] = 0 \,\} = \{\, 6, 8, 1, 3 \,\} \ominus 1 = \{5, 7, 0, 2\}$$
$$\mathbf{P}(2) = \{\, i \;:\; \mathbf{c}[i \oplus 2] = w[2] = 1 \,\} = \{\, 5, 7, 0, 2, 4 \,\} \ominus 2 = \{3, 5, 7, 0, 2\}.$$

Mamy więc $\mathbf{P}(0) \cap \mathbf{P}(1) \cap \mathbf{P}(2) = \{0, 2, 5, 7\}$.

Zliczanie wystąpień cyklicznych

Przedział cykliczny to taki, w którym po pozycji $n-1$ następują pozycje $0, 1, 2, \ldots$ Na przykład dla $n = 9$ przedział cykliczny $[7..2]$ odpowiada zbiorowi $\{7, 8, 0, 1, 2\}$. Odtąd zamiast zbiorów

$$\mathbf{P}(k) \;=\; \{\, i \;:\; w[k] = \mathbf{c}[i \oplus k] \,\}$$

będziemy posługiwać się zbiorami

$$\mathbf{V}(k) \;=\; \{\, v_i \;:\; w[k] = \mathbf{c}[i \oplus k] \,\}.$$

Innymi słowy, zbiór $\mathbf{V}(k)$ oznacza zakres indeksów w ciągu $\mathbf{C}$ odpowiadających elementom zbioru $\mathbf{P}(k)$. Na mocy faktu 3 mamy więc:

Fakt 4. *Zbiory* $\mathbf{V}(k)$ *są przedziałami cyklicznymi.*

Przykład 3. Dla poprzedniego przykładu mamy:

$$\mathbf{V}(0) = [4..8], \quad \mathbf{V}(1) = [4..7], \quad \mathbf{V}(2) = [3..7].$$

Ostatecznie $\mathbf{V}(0) \cap \mathbf{V}(1) \cap \mathbf{V}(2) = [4..7]$.

Zamiast przecinać zbiory $\mathbf{P}(k)$, będziemy przecinać zbiory $\mathbf{V}(k)$. Wyznaczenie przecięcia zbiorów $\mathbf{V}(k)$ sprowadza się do kolejnego wykonywania przecięcia pojedynczych przedziałów cyklicznych, co jest łatwe, o ile wynik też jest przedziałem cyklicznym. W naszym przypadku kolejność obliczania przecięć jest istotna. Skorzystamy z następującej oczywistej obserwacji.

Obserwacja 2. Przecięcie dwóch cyklicznych przedziałów zbioru $[0..n-1]$, których suma długości nie przekracza n, jest pojedynczym przedziałem cyklicznym.

W algorytmie startujemy z przedziałem $[0..n-1]$, a potem przecinamy go kolejno z przedziałami $\mathbf{V}(k)$, gdzie $w[k] = a$ oraz symbol a jest rzadziej występującym w tekście $\mathbf{c}$. Skoro a jest rzadziej występującym symbolem, to długość $\mathbf{V}(k)$ jest równa co najwyżej $\frac{n}{2}$, czyli suma długości takich przedziałów nie przekracza n. Następnie przecinamy rezultat kolejno ze zbiorami $\mathbf{V}(k)$ dla pozycji k zawierających częstszy symbol. Z powyższej obserwacji wynika, że zawsze wynikiem będzie przedział cykliczny.

Rozmiar ostatniego przedziału daje nam liczbę wystąpień cyklicznych (pamiętamy, że każdej wartości v odpowiada dokładnie jedna pozycja, na której w tekście występuje taka wartość).

Zliczanie wystąpień nadmiarowych

Pozostaje jeszcze problem obliczenia liczby *nadmiar*. Możemy go rozwiązać, sprawdzając dla każdego $i > n - m$ (czyli dla pozycji blisko końca tekstu $\mathbf{c}$), czy v_i należy do otrzymanego poprzednio przedziału wynikowego. Liczba pozycji, dla których to jest spełnione, jest równa liczbie *nadmiar*. Takich sprawdzeń jest m i są one łatwe, ponieważ sprawdzamy, czy jakaś liczba należy do danego przedziału cyklicznego.

Otrzymaliśmy zatem rozwiązanie całego zadania działające w złożoności czasowej $O(m + \log n)$ (składnik $O(\log n)$ pochodzi z obliczania odwrotności a modulo n) i pamięciowej $O(m)$.

Alternatywne spojrzenie na przecinanie przedziałów cyklicznych

W poprzednim rozwiązaniu zawsze przecinaliśmy przedziały cykliczne o sumie długości nieprzekraczającej n, dzięki czemu mogliśmy stosować obserwację 2. Warto jednak wspomnieć, że istnieje ogólny algorytm przecinania przedziałów cyklicznych. Jeżeli mamy do czynienia z przedziałem cyklicznym $[a..b]$, takim że $a \leqslant b$, to reprezentujemy go po prostu jako przedział $[a..b]$, a jeżeli $a > b$, to reprezentujemy go jako sumę przedziałów $[a..n-1]$ oraz $[0..b]$. Następnie wszystkie standardowe przedziały użyte w reprezentacjach przedziałów cyklicznych (jest ich co najwyżej $2m$) umieszczamy na osi liczbowej, sortujemy niemalejąco ich początki i końce i przetwarzamy je wszystkie w takiej kolejności, utrzymując w każdym momencie liczbę otwartych przedziałów. Jeżeli w pewnym momencie przetwarzania jest ona równa m, to znaczy, że aktualnie rozpatrywany punkt należy do wszystkich przedziałów cyklicznych. Konkretniej, ostateczna reprezentacja takiego przecięcia będzie sumą (być może wielu) standardowych przedziałów, a ich łączna długość jest rozmiarem przecięcia zbiorów $\mathbf{V}(k)$.

Nie będąc świadomym, że otrzymane przedziały w naszym zadaniu tworzą jeden przedział cykliczny, trudniej jest wykluczyć nadmiarowe wystąpienia w rozsądnym czasie. Aby to zrobić, powinniśmy przetwarzać je razem ze zdarzeniami typu „początek przedziału" oraz „koniec przedziału", które obsługiwaliśmy przy wyznaczaniu przecięcia przedziałów cyklicznych.

To alternatywne rozwiązanie ma tę niewielką wadę, że ze względu na sortowanie działa w złożoności czasowej $O(m \log m + \log n)$.

Jacek Tomasiewicz
Treść zadania, Opracowanie

Marek Sommer, Paweł Wegner
Program

XXII OI, etap II, dzień drugi, 12.02.2015

Trzy wieże

Bitoni uwielbia się bawić. W swoim pokoju ułożył w jednym rzędzie n klocków. Każdy z klocków ma jeden z trzech kolorów: biały, szary lub czarny. Bitoni chciałby wybrać pewien spójny fragment rzędu klocków, a następnie z klocków z tego fragmentu zbudować wieże.

Każda wieża może się składać z klocków tylko jednego koloru i nie może być dwóch wież o tym samym kolorze (zatem Bitoni zbuduje co najwyżej trzy wieże). Ponadto nie może być dwóch wież o tej samej wysokości (tzn. każda wieża musi być zbudowana z innej liczby klocków niż pozostałe). Zakładamy, że Bitoni musi wykorzystać wszystkie wybrane przez siebie klocki. Pomóż Bitoniemu i napisz program, który znajdzie najdłuższy fragment rzędu klocków spełniający jego wymagania.

Wejście

Pierwszy wiersz standardowego wejścia zawiera jedną liczbę całkowitą n ($1 \leqslant n \leqslant 1\,000\,000$), oznaczającą liczbę klocków. Kolejny wiersz zawiera napis złożony z n liter $a_1a_2\ldots a_n$, w którym a_i jest jedną z liter `B`, `S` *lub* `C` *i oznacza kolor i-tego klocka w rzędzie (litera* `B` *oznacza klocek koloru białego, litera* `S` *klocek szary, a litera* `C` *klocek czarny).*

W testach wartych 30% punktów zachodzi dodatkowy warunek $n \leqslant 2500$.

Wyjście

Pierwszy i jedyny wiersz standardowego wyjścia powinien zawierać jedną liczbę całkowitą, równą liczbie klocków w najdłuższym spójnym fragmencie rzędu, który spełni wymagania Bitoniego.

Przykład

Dla danych wejściowych:	*poprawnym wynikiem jest:*
`9` `CBBSSBCSC`	`6`

Wyjaśnienie do przykładu: *Bitoni może wybrać fragment złożony z 6 klocków:* `BSSBCS`*, z których zbuduje szarą wieżę złożoną z trzech klocków, białą z dwóch klocków oraz czarną z jednego klocka.*

Testy „ocen":

- `1ocen`*: $n = 2500$, rząd klocków jest następujący:* $\texttt{B}^{1248}\underline{\texttt{CSB}^{1250}}$ *(napis* $\texttt{B}^k$ *oznacza k-krotne powtórzenie litery* `B`*); najdłuższy fragment rzędu klocków, który Bitoni może wybrać, został podkreślony;*
- `2ocen`*: $n = 1\,000\,000$, rząd klocków jest okresowy:* `BSCBSCBSC`$\ldots$`BSCBSCB`*; Bitoni może zbudować tylko jedną wieżę z jednego klocka.*

Rozwiązanie

W zadaniu mamy dane słowo Z o długości n składające się z literek `A`, `B`, `C` (dla czytelności opisu zamiast literki `S` używamy literki `A`) reprezentujących kolory kolejnych klocków ułożonych w rzędzie. Musimy znaleźć najdłuższy fragment ciągu klocków, taki że liczba wystąpień klocków każdego koloru w tym fragmencie będzie inna (tak aby wysokości wież zbudowanych z klocków tego samego koloru były różne). Fragment (niekoniecznie najdłuższy) spełniający ten warunek nazwiemy *poprawnym*.

Rozwiązanie siłowe $O(n^3)$

Dla każdego spójnego fragmentu ciągu klocków możemy sprawdzić, czy jest on poprawny. Łatwo to wykonać w czasie liniowym: wystarczy policzyć i porównać liczbę wystąpień klocków każdego koloru. Jako że wszystkich spójnych fragmentów jest $O(n^2)$, a pojedyncze sprawdzenie zajmuje czas liniowy względem długości fragmentu, rozwiązanie to ma złożoność czasową $O(n^3)$.

Rozwiązanie to zaimplementowane jest w pliku `trzs2.cpp`. Za poprawne zaprogramowanie takiego rozwiązania na zawodach można było uzyskać około 15% punktów.

Rozwiązanie wolne $O(n^2)$

Sprawdzenie poprawności fragmentu możemy wykonać szybciej, w czasie $O(1)$, po wcześniejszym przetworzeniu ciągu klocków w czasie $O(n)$. Aby móc efektywnie obliczać liczbę wystąpień klocków każdego koloru w dowolnym fragmencie, w pierwszym kroku wyznaczymy ciąg sum częściowych słowa Z dla każdego koloru oddzielnie. Załóżmy, że rozpatrujemy kolor `A`.

Niech $w_i = 1$, jeśli $Z_i =$ `A`, zaś w przeciwnym przypadku $w_i = 0$. i-tą sumę częściową (dla $1 \leqslant i \leqslant n$) definiujemy jako $b_i = w_1 + w_2 + \ldots + w_i$, jednocześnie przyjmując $b_0 = 0$. Zauważmy, że $b_i = b_{i-1} + w_i$, więc ciąg $b_1, \ldots, b_n$ można obliczyć w czasie $O(n)$. Wartości b_i pozwalają obliczyć liczbę wystąpień klocków koloru `A` w dowolnym fragmencie klocków $Z_i, \ldots, Z_j$ w czasie stałym ze wzoru $b_j - b_{i-1}$. W analogiczny sposób możemy wyznaczyć ciąg sum częściowych dla koloru `B` i `C`.

Implementacja takiego rozwiązania znajduje się w pliku `trzs3.cpp`. Rozwiązanie tego typu otrzymywało na zawodach około 30% punktów.

Rozwiązanie wzorcowe $O(n)$

Udowodnimy, że optymalne wyniki znajdują się blisko jednego z końców ciągu klocków. Dokładniej, jeśli $[i, j]$ jest przedziałem reprezentującym optymalny fragment ciągu klocków (spośród wszystkich optymalnych wybierzmy ten o najmniejszym i), to $i \leqslant 3$ lub $j \geqslant n-2$. Jeśli tak rzeczywiście jest, to można wyznaczyć $O(n)$ kandydatów (przedziałów), wśród których znajduje się optymalny wynik. Te przedziały to $[1, j]$, $[2, j]$, $[3, j]$ oraz $[i, n]$, $[i, n-1]$, $[i, n-2]$ dla wszystkich i, j, takich że przedziały te będą poprawnie zdefiniowane – początek przedziału nie będzie większy od końca przedziału.

Po przetworzeniu ciągu w czasie $O(n)$, będziemy mogli w czasie stałym stwierdzić, czy przedział odpowiada poprawnemu fragmentowi (powiemy wtedy, że przedział jest poprawny). Ostatecznie, rozwiązanie to ma złożoność czasową $O(n)$ i zostało zaimplementowane w pliku `trz2.cpp`. Zaimplementowanie takiego rozwiązania w trakcie zawodów było nagradzane maksymalną liczbą punktów. Autorem tego rozwiązania jest Marek Sommer.

Dowód

Niech przedział $[i, j]$ będzie najdłuższym poprawnym przedziałem, a spośród wszystkich najdłuższych niech będzie tym, który ma najmniejsze i. Spróbujemy założyć, że przedział nie znajduje się blisko żadnego z końców ciągu, czyli $i > 3$ i $j < n - 2$. To oznacza, że z każdej strony przedziału znajdują się przynajmniej po trzy klocki, których nie można dołożyć do tego przedziału.

$\cdots$	?	?	?	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	?	?	?	$\cdots$

Dowód będzie opierał się na rozważeniu wielu przypadków (ale dzięki temu nie ma ich już w algorytmie) i pokazaniu, że w każdym z nich dochodzimy do sprzeczności – pokażemy, że będzie istniał lepszy lub równoważny przedział, który znajduje się blisko jednego z końców ciągu. Najpierw jednak zachęcamy Czytelnika do próby samodzielnego przeprowadzenia dowodu, który można wykonać w łatwy sposób, rozpisując różne przypadki na kartce. Poniżej przedstawiamy sposoby rozpatrzenia poszczególnych przypadków.

W przedziale $[i, j]$ znajdują się klocki tylko jednego koloru

Mamy więc $Z_i = Z_{i+1} = \ldots = Z_{j-1} = Z_j$. Jeśli $i \neq j$, to dokładając element Z_{i-1}, dostaniemy również poprawny przedział (klocków koloru Z_i jest co najmniej 2, więc nowy klocek tego samego bądź innego koloru niczego nie popsuje). Z tego wynika, że przedział $[i, j]$ nie jest najdłuższy poprawny – dostajemy sprzeczność. Jeśli natomiast $i = j$, to przedział jest długości 1. Dowolny przedział długości 1 jest poprawny, więc poprawny jest również przedział $[1, 1]$. Otrzymujemy sprzeczność, bo przedział $[i, j]$ miał być optymalnym rozwiązaniem o najmniejszym i.

W przedziale $[i, j]$ znajdują się klocki dokładnie dwóch kolorów

W tym przypadku możemy przyjąć, że w przedziale $[i, j]$ znajdują się klocki trzech kolorów, przy czym jedna z utworzonych wież ma wysokość 0, co sprowadza się do następnego przypadku, opisanego poniżej.

W przedziale $[i, j]$ znajdują się klocki dokładnie trzech kolorów

Niech $|X|$ oznacza liczbę wystąpień klocków koloru X w przedziale $[i, j]$. Bez straty ogólności możemy założyć, że $|\texttt{A}| < |\texttt{B}| < |\texttt{C}|$. Co teraz może się kryć pod znakami zapytania?

$\cdots$	$?_1$	$?_2$	$?_3$	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	$?_4$	$?_5$	$?_6$	$\cdots$

Znaki $?_3$ i $?_4$ nie mogą być równe `C` (w przeciwnym przypadku można by było powiększyć nimi rozwiązanie).

Przypadek, w którym $?_3 =$ `B`

$\cdots$	$?_1$	$?_2$	`B`	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	$?_4$	$?_5$	$?_6$	$\cdots$

Czy $?_4$ może być równe `B`? Nie, ponieważ wtedy przedział $[i, j]$ sąsiadowałby z dwiema literkami `B` i w zależności od tego, czy $|\texttt{B}| + 1 = |\texttt{C}|$, czy nie, moglibyśmy powiększyć rozwiązanie albo przy pomocy jednej, albo dwóch literek `B`. Znak $?_4$ nie może być też równy `C` (co było powiedziane wcześniej). Zatem otrzymujemy $?_4 =$ `A`.

$\cdots$	$?_1$	$?_2$	`B`	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	`A`	$?_5$	$?_6$	$\cdots$

Wiadomo, że $|\texttt{A}| + 1 = |\texttt{B}|$ i $|\texttt{B}| + 1 = |\texttt{C}|$ (gdyby tak nie było, to można by było powiększyć rozwiązanie o jedną literkę `A` lub jedną literkę `B`). Możemy więc zapisać, że $|\texttt{A}| = x$, $|\texttt{B}| = x + 1$ i $|\texttt{C}| = x + 2$ dla pewnego x.

Czy $?_5$ może być równe `C`? Nie, ponieważ wtedy moglibyśmy dodać znaki $?_3$, $?_4$ i $?_5$, otrzymując lepsze rozwiązanie, gdzie $|\texttt{A}| = x + 1$, $|\texttt{B}| = x + 2$ i $|\texttt{C}| = x + 3$. Czy $?_5$ może być równe `B`? Nie, ponieważ wtedy moglibyśmy dodać znaki $?_3$, $?_4$ i $?_5$, otrzymując lepsze rozwiązanie, gdzie $|\texttt{A}| = x + 1$, $|\texttt{C}| = x + 2$ i $|\texttt{B}| = x + 3$. Zatem otrzymujemy $?_5 =$ `A`.

$\cdots$	$?_1$	$?_2$	`B`	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	`A`	`A`	$?_6$	$\cdots$

$?_6$ nie może być równe `A` ani `C`, ponieważ dodając znaki $?_4$, $?_5$ i $?_6$, otrzymalibyśmy lepsze rozwiązanie. Zatem otrzymujemy $?_6 =$ `B`.

$\cdots$	$?_1$	$?_2$	`B`	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	`A`	`A`	`B`	$\cdots$

$?_2$ nie może być równe `B` ani `C`, ponieważ dodając znaki $?_2$, $?_3$ i $?_4$, otrzymalibyśmy lepsze rozwiązanie. Zatem otrzymujemy $?_2 =$ `A`.

$\cdots$	$?_1$	`A`	`B`	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	`A`	`A`	`B`	$\cdots$

$?_1$ nie może być równe `B` ani `C`, ponieważ dodając znaki $?_1$, $?_2$ i $?_3$, otrzymalibyśmy lepsze rozwiązanie. Zatem otrzymujemy $?_1 =$ `A`.

$\cdots$	`A`	`A`	`B`	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	`A`	`A`	`B`	$\cdots$

Okazuje się, że dodając wszystkie znaki od $?_1$ do $?_6$, otrzymalibyśmy lepsze rozwiązanie, w którym $|\texttt{C}| = x + 2$, $|\texttt{B}| = x + 3$ i $|\texttt{A}| = x + 4$, więc mamy sprzeczność.

Przypadek, w którym $?_3 = \texttt{A}$

$\cdots$	$?_1$	$?_2$	A	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	$?_4$	$?_5$	$?_6$	$\cdots$

Gdyby $?_4 = \texttt{B}$, to otrzymalibyśmy przypadek symetryczny do poprzedniego przypadku, w którym $?_3 = \texttt{B}$ (który był sprzeczny), więc $?_4$ nie może być równe B ani też C (co było powiedziane wcześniej). Zatem otrzymujemy $?_4 = \texttt{A}$.

$\cdots$	$?_1$	$?_2$	A	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	A	$?_5$	$?_6$	$\cdots$

Wiadomo, że $|\texttt{A}| + 1 = |\texttt{B}|$ i $|\texttt{A}| + 2 = |\texttt{C}|$ (gdyby tak nie było, to można by było dodać jedną lub dwie literki A, otrzymując lepsze rozwiązanie). Możemy więc zapisać, że $|\texttt{A}| = x$, $|\texttt{B}| = x + 1$ i $|\texttt{C}| = x + 2$, dla pewnego x.

$?_5$ nie może być równe A ani C, ponieważ dodając znaki $?_3$, $?_4$ i $?_5$, otrzymalibyśmy lepsze rozwiązanie. Zatem otrzymujemy $?_5 = \texttt{B}$.

$\cdots$	$?_1$	$?_2$	A	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	A	B	$?_6$	$\cdots$

Znak $?_2$ jest w tym przypadku symetryczny do $?_5$, więc tak samo możemy wywnioskować, że $?_2 = \texttt{B}$.

$\cdots$	$?_1$	B	A	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	A	B	$?_6$	$\cdots$

$?_6$ nie może być równe B ani C, ponieważ dodając znaki $?_4$, $?_5$ i $?_6$, otrzymalibyśmy lepsze rozwiązanie. Zatem otrzymujemy $?_6 = \texttt{A}$.

$\cdots$	$?_1$	B	A	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	A	B	A	$\cdots$

Znak $?_1$ jest w tym przypadku symetryczny do $?_6$, więc tak samo możemy wywnioskować, że $?_1 = \texttt{A}$.

$\cdots$	A	B	A	Z_i	Z_{i+1}	$\cdots$	Z_{j-1}	Z_j	A	B	A	$\cdots$

Okazuje się, że dodając wszystkie znaki od $?_1$ do $?_6$, otrzymalibyśmy lepsze rozwiązanie, w którym $|\texttt{C}| = x + 2$, $|\texttt{B}| = x + 3$ i $|\texttt{A}| = x + 4$, więc mamy sprzeczność.

Mniejsza liczba kandydatów

Udowodniliśmy, że wystarczy sprawdzić przedziały, które są w odległości nie większej niż 2 od któregoś z końców ciągu klocków. Jest to najmniejsze możliwe ograniczenie. Gdyby algorytm sprawdzał tylko przedziały w odległości nie większej niż 1, to źle odpowiedziałby w następującym przypadku:

ABCACBACBA

Nie wystarczy również sprawdzać przedziałów znajdujących się blisko początku:

BBBBBBCAACBBBBBBBBBBBBBBBBBBB

Zawody III stopnia

opracowania zadań