Wojciech Rytter, Bartosz Szreder
Treść zadania, Opracowanie

Karol Farbiś, Szymon Stankiewicz
Program

Dostępna pamięć: 64 MB.

OI, etap I, 7.10–4.11.2013

Klocki

Mały Bitek i jego koledzy cały wczorajszy dzień w przedszkolu spędzili na zabawie kolorowymi klockami. Tworzyli przeróżne budowle, ale szybko im się to znudziło. Postanowili więc poustawiać klocki jeden za drugim. Unikali oni przy tym sytuacji, w której pewien klocek stoi tuż przed klockiem tego samego koloru. Po dłuższym czasie udało im się ustawić w ten sposób wszystkie klocki. Wtedy zajęcia się skończyły i rodzice odebrali dzieci z przedszkola.

Dzisiaj Bitek przybył do przedszkola pół godziny przed zajęciami. Zobaczył, że dzieło zbudowane poprzedniego dnia nadal istniało. Niestety, Bitek przewrócił się i wszystkie klocki się wymieszały. Chłopiec natychmiast uporządkował klocki i zaczął zastanawiać się, jak szybko odbudować to, co zepsuł. Przypomniał sobie, jakiego koloru klocki stały na końcach rzędu.

Pomóż małemu Bitkowi i podpowiedz, jak może ustawić klocki, żeby żaden klocek nie stał przed klockiem tego samego koloru oraz żeby klocki na końcach miały takie kolory, jakie podał Bitek. Jeśli Bitkowi coś się pomyliło lub chłopiec przypadkowo zagubił niektóre klocki podczas upadku, tak że teraz ustawienie zgodne z jego warunkami nie jest możliwe, pomóż mu to stwierdzić.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się trzy liczby całkowite k, p i q ($1 \leqslant k \leqslant 1\,000\,000$, $1 \leqslant p, q \leqslant k$), pooddzielane pojedynczymi odstępami i oznaczające liczbę kolorów klocków oraz kolory pierwszego i ostatniego klocka w szukanym ustawieniu. W drugim wierszu znajduje się k liczb całkowitych $i_1, i_2, \ldots, i_k$, pooddzielanych pojedynczymi odstępami ($1 \leqslant i_j \leqslant 1\,000\,000$). Liczba i_j oznacza, że Bitek ma dokładnie i_j klocków koloru j. Możesz założyć, że liczba wszystkich klocków nie przekracza miliona, tzn. $n = i_1 + i_2 + \ldots + i_k \leqslant 1\,000\,000$.

Wyjście

Twój program powinien wypisać na standardowe wyjście n liczb całkowitych pooddzielanych pojedynczymi odstępami reprezentujących kolory kolejnych klocków w szukanym ułożeniu. Jeśli takie ułożenie nie istnieje, Twój program powinien wypisać tylko jedną liczbę całkowitą: 0.

Jeśli jest wiele poprawnych odpowiedzi, Twój program może podać dowolną z nich.

Przykład

Dla danych wejściowych:

```
3 3 1
2 3 3
```

jednym z poprawnych wyników jest:

```
3 2 1 3 2 3 2 1
```

a dla danych wejściowych:	*poprawnym wynikiem jest:*
`3 3 1` `2 4 2`	`0`

Wyjaśnienie do przykładów:
Innym poprawnym ułożeniem klocków w pierwszym przykładzie jest `3 1 2 3 2 3 2 1`. *W drugim przykładzie Bitek musiał się pomylić – nie można ułożyć klocków tak, żeby spełnione były warunki z treści zadania.*

Rozwiązanie

Wstęp

W zadaniu *Klocki* naszym celem jest ułożyć z podanego zestawu kolorowych klocków *ciąg alternujący*, czyli taki, w którym żadna liczba (reprezentująca kolor klocka) nie występuje dwa razy pod rząd. Ponadto ustalony jest element początkowy oraz końcowy ciągu.

Niech i_1 oznacza liczbę dostępnych klocków koloru 1, i_2 liczbę klocków koloru 2 itd. Kolor, którym mamy rozpocząć ciąg, oznaczmy przez p, natomiast kolor „końcowy" oznaczmy przez q. Ponadto k to liczba różnych kolorów klocków, a n to liczba wszystkich klocków, tzn. $n = i_1 + i_2 + \ldots + i_k$. Dla uproszczenia opisu rozwiązania przyjmujemy od teraz, że liczności klocków poszczególnych kolorów są posortowane niemalejąco, czyli i_1 to liczba klocków najrzadziej występujących w podanym zbiorze, a i_k to liczba klocków najpopularniejszych w zestawie. Kolory elementu początkowego i końcowego nazwiemy kolorami *krańcowymi*.

Ciągi alternujące

Rozwiążmy najpierw odrobinę prostsze zadanie: z podanego zestawu klocków chcemy ułożyć **jakikolwiek** ciąg alternujący – bez ustalonego elementu początkowego i końcowego. Okazuje się, że w tym przypadku łatwo sprawdzić, czy jest to w ogóle możliwe i, jeśli jest, ułożyć taki ciąg. Warunek konieczny i wystarczający do istnienia ciągu alternującego prezentuje się następująco:

$$i_k \leqslant \frac{n+1}{2}.$$

Przypomnijmy, że i_k oznacza liczbę najczęściej występujących klocków.

Konieczność wynika z zasady szufladkowej Dirichleta: gdyby powyższa nierówność nie była spełniona, to nawet jeśli wstawilibyśmy klocek koloru k na co drugiej pozycji w ciągu, zabrakłoby nam klocków o innych kolorach i musielibyśmy wstawić obok siebie dwa klocki koloru k.

Dostateczność można wykazać poprzez podanie algorytmu konstrukcji ciągu alternującego:

1. Załóżmy, że $n = i_1 + i_2 + \ldots + i_k$ jest nieparzyste.

2. Znajdź j i taki podział i_j na $i'_j + i''_j = i_j$, żeby

$$i_1 + i_2 + \ldots + i'_j + 1 = i''_j + i_{j+1} + \ldots + i_k.$$

3. Niech S_1 będzie ciągiem i_1 klocków koloru 1, i_2 klocków koloru 2, …, i'_j klocków koloru j.

4. Niech S_2 będzie ciągiem i''_j klocków koloru j, i_{j+1} klocków koloru $j+1$, …, i_k klocków koloru k.

5. Zbuduj docelowy ciąg, przeplatając elementy ciągów S_2 i S_1.

Dla n parzystych jest tak samo, tylko bez $+1$ w szukaniu podziału (drugi podpunkt algorytmu).

Właściwy problem

Okazuje się, że warunek wystarczający do skonstruowania ciągu alternującego dla **dowolnej** pary kolorów krańcowych (oznaczanych dalej p i q) jest bardzo podobny do warunku koniecznego istnienia dowolnego ciągu alternującego:

$$i_k \leqslant \frac{n-1}{2}.$$

Jeśli $p = q$, dochodzi do tego prosty warunek brzegowy, polegający na sprawdzeniu, czy $i_p \geqslant 2$.

Konstrukcję docelowego ciągu zaczynamy od usunięcia dwóch elementów ze zbioru klocków: jednego w kolorze p i jednego w kolorze q. Nawet jeśli $p \neq k$ i $q \neq k$, to nadal spełniony jest warunek wystarczający do zbudowania jakiegokolwiek ciągu alternującego, co też czynimy. Na końcach tego ciągu należy następnie dodać dwa wcześniej wyjęte klocki p i q.

Teraz zadanie polega na tym, żeby nie doprowadzić do powtórzenia koloru na lewym albo prawym brzegu ciągu. Oznaczmy kolory klocków na końcach otrzymanego ciągu alternującego jako $\hat{p}$ i $\hat{q}$. Mamy następujące przypadki:

1. $\{p, q\} \cap \{\hat{p}, \hat{q}\} = \emptyset$. Przypadek trywialny do rozwiązania: dostawiamy klocki p i q jakkolwiek i zawsze będzie poprawnie.

2. $p \neq q \wedge \hat{p} \neq \hat{q}$. Przypadek niewiele trudniejszy: może być tak, że p lub q równe jest $\hat{p}$ lub $\hat{q}$, więc możemy być zmuszeni wstawić p (albo q) po konkretnej stronie, żeby nie wystąpiło powtórzenie koloru.

3. $\hat{p} = \hat{q}$. Podany wcześniej algorytm konstrukcji ciągu alternującego doprowadzi do takiej sytuacji tylko wtedy, gdy kolor $\hat{p}$ jest najliczniejszy w zredukowanym ciągu oraz kolor ten występuje maksymalną dopuszczalną liczbę razy. Ale z tego wynika, że ani p, ani q nie są tego samego koloru co $\hat{p}$, bo wtedy byłoby $i_k > \frac{n-1}{2}$. To oznacza, że jest to ten sam przypadek co (1).

4. Możemy teraz przyjąć, że $\hat{p} \neq \hat{q}$, $p = q$ oraz p jest takie samo jak $\hat{p}$ albo $\hat{q}$. Bez straty ogólności przyjmijmy $p = \hat{p}$. Mamy więc trzy elementy koloru p i jeden koloru $\hat{q}$. Jeden klocek koloru p oraz klocek koloru $\hat{q}$ ustawiamy po jednej stronie. Teraz musimy przepchnąć jedno p w takie miejsce w środku ciągu, w którym sąsiadują dwa elementy koloru innego od p. Takie miejsce zawsze istnieje – gdyby nie istniało, to element koloru p musiałby występować na co drugiej pozycji w zredukowanym ciągu, a zatem doliczając dwa dodatkowe elementy koloru p (tzn. elementy p i q), zaburzylibyśmy nierówność z warunku koniecznego.

Jeśli warunek wystarczający do ułożenia ciągu alternującego dla dowolnych p i q nie jest spełniony, nie wszystko stracone. Istnieje jeszcze kilka przypadków, gdy poprawny ciąg da się ułożyć:

- Dla n nieparzystego oraz $i_k = \frac{n+1}{2}$ musimy rozmieszczać klocki koloru k na co drugiej pozycji, także początkowej i końcowej, aby ciąg był alternujący. Rozwiązanie zatem istnieje dla $p = q = k$.
- Dla n parzystego oraz $i_k = \frac{n}{2}$ przynajmniej jeden z klocków p, q musi być koloru k. Co więcej, jeśli oba są koloru k, to poprawny ciąg da się ułożyć wtedy i tylko wtedy, gdy $i_{k-1} < \frac{n}{2}$, czyli gdy są przynajmniej trzy różne kolory klocków. Jeśli jednakże $k = 2$ i $i_1 = i_2 = \frac{n}{2}$, to jedyny ciąg alternujący przeplata kolory 1 i 2, więc początek i koniec muszą mieć takie właśnie kolory.

Rozwiązanie wzorcowe

Ponieważ musimy posortować klocki według liczności ich kolorów, złożoność czasowa rozwiązania wzorcowego wyznaczona jest przez złożoność zastosowanego algorytmu sortowania. Wszystkie późniejsze operacje wykonywane są w czasie stałym (sprawdzanie warunków istnienia ciągu alternującego) bądź liniowym (konstrukcja ciągu alternującego).

Jako wzorcowe uznano programy działające w czasie $O(k \log k + n)$, stosujące efektywny algorytm sortowania przez porównania, lub $O(n)$, używające sortowania kubełkowego. Rozwiązanie o złożoności $O(k \log k + n)$ zaimplementowano w plikach `klo.cpp`, `klo1.c` i `klo2.pas`.