Tomasz Idziaszek — Treść zadania, Opracowanie
Karol Pokorski — Program

Dostępna pamięć: 128 MB. *OI, etap I, 6.10–3.11.2014*

Kinoman

Bajtazar jest zapalonym kinomanem, dlatego ucieszył się, gdy jego ulubione kino studyjne przygotowało bardzo ciekawą promocję na lato. Każdego z n dni lata w kinie będzie wyświetlany jeden z m filmów. Promocyjny karnet uprawnia do bezpłatnego wejścia na dowolną liczbę seansów, pod warunkiem że jego właściciel nie będzie robił przerw (tzn. ominięcie seansu unieważnia karnet; pierwszy seans można wybrać dowolnie).

Na podstawie internetowych recenzji Bajtazar przyporządkował każdemu z m filmów jego współczynnik fajności. Bajtazar chciałby wykorzystać promocyjny karnet w taki sposób, aby zmaksymalizować sumę współczynników fajności obejrzanych filmów. Niestety nie jest to takie proste, gdyż Bajtazar okropnie nie lubi oglądać dwa razy tego samego filmu. Powtórny seans nuży go i odbiera przyjemne wspomnienia, które wiązał z filmem. Zatem chce on tak naprawdę zmaksymalizować sumę współczynników fajności tych filmów, które obejrzy dokładnie raz.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite n i m ($1 \leqslant m \leqslant n \leqslant 1\,000\,000$) oddzielone pojedynczym odstępem, oznaczające liczbę dni promocji oraz liczbę filmów. Dla ułatwienia filmy numerujemy liczbami od 1 do m.

W drugim wierszu znajduje się ciąg n liczb całkowitych $f_1, f_2, \ldots, f_n$ ($1 \leqslant f_i \leqslant m$) pooddzielanych pojedynczymi odstępami: liczba f_i oznacza numer filmu wyświetlanego i-tego dnia promocji. W trzecim wierszu znajduje się ciąg m liczb całkowitych $w_1, w_2, \ldots, w_m$ ($1 \leqslant w_j \leqslant 1\,000\,000$) pooddzielanych pojedynczymi odstępami: liczba w_j oznacza współczynnik fajności filmu o numerze j. Może się tak zdarzyć, że pewne spośród podanych m filmów nie będą w ogóle wyświetlane w trakcie trwania letniej promocji.

W testach wartych 70% punktów zachodzi dodatkowy warunek $n \leqslant 100\,000$, a w podzbiorze tych testów wartym 20% punktów zachodzi warunek $n \leqslant 8000$.

Wyjście

W jedynym wierszu standardowego wyjścia należy wypisać jedną liczbę całkowitą oznaczającą sumaryczną fajność filmów, które Bajtazar obejrzy dokładnie raz, jeśli optymalnie wykorzysta promocyjny karnet.

Przykład

Dla danych wejściowych:

```
9 4
2 3 1 1 4 1 2 4 1
5 3 6 6
```

poprawnym wynikiem jest:

```
15
```

Wyjaśnienie do przykładu: *Bajtazar może wykorzystać karnet, aby obejrzeć 6 seansów, poczynając od drugiego dnia. W ten sposób obejrzy dokładnie raz filmy o numerach 2, 3 i 4.*

Testy „ocen”:

`1ocen`: *n = 10, m = 5, losowy;*

`2ocen`: *n = 100, m = 50, losowy;*

`3ocen`: *n = 1 000 000, m = 1 000 000, wszystkie filmy poza jednym mają fajność 200 000 i nie powtarzają się; jeden ma fajność 1 000 000 i powtarza się raz na 10 dni.*

Rozwiązanie

W opracowaniu założymy dla uproszczenia, że współczynnik fajności każdego filmu jest równy jego numerowi. W ten sposób, odzierając zadanie z warstwy fabularnej, możemy je wysłowić następująco: dany jest ciąg n liczb $a_1, a_2, \ldots, a_n$; należy znaleźć taki spójny fragment w tym ciągu, że suma liczb występujących dokładnie raz w tym fragmencie jest możliwie jak największa.

Dla przykładu rozważmy ciąg $a = 5, 6, 2, 5, 2, 8, 5, 5, 4, 4, 2$ i zastanówmy się, jak wyglądają sumy fragmentów, które zaczynają się pierwszym wyrazem ciągu (czyli sumy prefiksów tego ciągu). Niech s_i oznacza sumę prefiksu kończącego się wyrazem na pozycji i. Mamy wtedy:

a_i	5	6	2	5	2	8	5	5	4	4	2
s_i	5	11	13	8	6	14	14	14	18	14	14

Przykładowo prefiks $5, 6, 2, 5, 2, 8, 5$ zawiera dokładnie jedną szóstkę i ósemkę (a pozostałe liczby występują w nim wielokrotnie), więc jego suma wynosi $s_7 = 6 + 8 = 14$. Jak wyznaczyć ciąg s? Rozważmy dowolną wartość x występującą w ciągu a i niech $\{j_1, j_2, \ldots, j_k\}$ będzie zbiorem pozycji, na których ta wartość się znajduje (czyli $x = a_{j_1} = a_{j_2} = \ldots = a_{j_k}$). Wartość x będzie liczyła się do sumy tych prefiksów, w których występuje dokładnie raz; będą to więc prefiksy kończące się na pozycjach od j_1 do $j_2 - 1$. Jeśli zatem zdefiniujemy ciąg p w taki sposób, że dla wartości x ustalimy $p_{j_1} = +x$, $p_{j_2} = -x$ oraz $p_{j_3} = \ldots = p_{j_k} = 0$, to nie będzie zaskoczeniem, że ciąg s uzyskamy, licząc sumy prefiksowe ciągu p (czyli ze wzoru $s_i = s_{i-1} + p_i$):

a_i	5	6	2	5	2	8	5	5	4	4	2
p_i	+5	+6	+2	−5	−2	+8	0	0	+4	−4	0
s_i	5	11	13	8	6	14	14	14	18	14	14

Przyjrzyjmy się teraz, jak zmienią się ciągi p i s, jeśli usuniemy z ciągu a pierwszy wyraz (będzie to odpowiadało rozważaniu tych fragmentów, które zaczynają się drugim wyrazem ciągu a). Zauważmy, że w ciągu p będziemy musieli uaktualnić jedynie trzy wyrazy odpowiadające wartości pierwszego wyrazu ciągu a:

a_i	–	6	2	5	2	8	5	5	4	4	2
p_i	**0**	+6	+2	+**5**	−2	+8	−**5**	0	+4	−4	0
s_i	0	6	8	13	11	19	14	14	18	14	14

W analogiczny sposób możemy usuwać kolejne wyrazy ciągu a. W ogólności, jeśli usuwamy wyraz o wartości x z pozycji j_i, to musimy uaktualnić wyrazy ciągu p na pozycjach j_i, j_{i+1} oraz j_{i+2} (o ile istnieją). W ten sposób rozważymy sumy wszystkich spójnych fragmentów ciągu a. Zauważmy, że odpowiedzią do zadania jest największa wartość, która kiedykolwiek pojawiła się w ciągu s.

W ten sposób można otrzymać rozwiązanie o złożoności $O(n^2)$, zaimplementowane w pliku `kins2.cpp`.

Rozwiązanie wzorcowe

W celu efektywnej implementacji powyższego rozwiązania, możemy wykorzystać strukturę danych, która umożliwi nam szybkie wykonywanie następujących operacji na ciągu liczb $p_1, p_2, \ldots, p_n$:

- zmiana wartości jednego wyrazu ciągu p,
- wyznaczenie największej wartości w ciągu sum prefiksowych ciągu p.

Zauważmy, że nie potrzebujemy w całości konstruować ciągu s – wystarczy nam jedynie znajomość największego wyrazu tego ciągu. Powyższą strukturę danych można zaimplementować jako statyczne drzewo przedziałowe, w którym obie operacje będą realizowane w czasie $O(\log n)$. Taka struktura danych pojawiła się już na XVI Olimpiadzie Informatycznej w zadaniu *Łyżwy* [16] (patrz także opis w książce [42]).

Na strukturze danych co najwyżej $3n$ razy wykonamy operację zmiany wartości wyrazu i n razy wykonamy operację wyznaczenia maksimum sum prefiksowych. Zatem cały algorytm będzie miał złożoność czasową $O(n \log n)$. Potrzebujemy jeszcze wyznaczyć indeksy $j_1, j_2, \ldots, j_k$ dla wszystkich wartości x z ciągu a, ale można to zrobić na początku programu w sumarycznym czasie $O(n)$. Zainteresowanych Czytelników odsyłamy do rozwiązania wzorcowego zawartego w pliku `kin.cpp`.