Jakub Pachocki
Treść zadania, Opracowanie

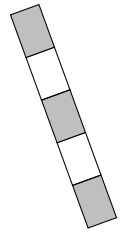
Mateusz Baranowski
Program

Dostępna pamięć: 64 MB.

OI, Etap I, 18.10–15.11.2010

Lizak

Bajtazar prowadzi w Bajtogrodzie sklep ze słodyczami. Wśród okolicznych dzieci najpopularniejszymi słodyczami są lizaki waniliowo-truskawkowe. Składają się one z wielu segmentów jednakowej długości, z których każdy ma jeden smak — waniliowy lub truskawkowy. Cena lizaka jest równa sumie wartości jego segmentów; segment waniliowy kosztuje jednego bajtalara, a truskawkowy dwa bajtalary.



Rys. 1: Przykładowy lizak o pięciu segmentach, trzech truskawkowych i dwóch waniliowych, ułożonych na przemian. Cena tego lizaka wynosi 8 bajtalarów.

Obecnie Bajtazarowi został na składzie tylko jeden (za to być może bardzo długi) lizak. Bajtazar zdaje sobie sprawę, że być może nikt nie będzie chciał go kupić w całości, dlatego dopuszcza możliwość łamania go na granicach segmentów w celu uzyskania lizaka o mniejszej długości. Fragment lizaka przeznaczony ostatecznie do sprzedaży musi pozostać niepołamany.

Doświadczenie pokazuje, że klienci najczęściej chcą kupić lizaka za całe swoje kieszonkowe. Bajtazar zastanawia się, dla wielu możliwych wartości k, jak przełamać posiadany lizak tak, aby otrzymać lizak o cenie równej dokładnie k bajtalarów. Ponieważ zadanie nie jest wcale proste, poprosił Cię o pomoc.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite n oraz m ($1 \leqslant n, m \leqslant 1\,000\,000$) oddzielone pojedynczym odstępem. Oznaczają one odpowiednio liczbę segmentów ostatniego pozostałego w sklepie lizaka oraz liczbę rozpatrywanych wartości k. Segmenty lizaka są ponumerowane kolejno od 1 do n. W drugim wierszu znajduje się n-literowy opis lizaka, złożony z liter `T` *i* `W`*, przy czym* `T` *oznacza segment truskawkowy, zaś* `W` *— waniliowy; i-ta z tych liter opisuje smak i-tego segmentu. W kolejnych m wierszach znajdują się kolejne wartości k do rozpatrzenia ($1 \leqslant k \leqslant 2\,000\,000$), po jednej w wierszu.*

Wyjście

Twój program powinien wypisać na standardowe wyjście dokładnie m wierszy zawierających wyniki dla kolejnych wartości k, po jednym wyniku w wierszu. Jeśli dla danej wartości k nie da

się wyłamać z lizaka spójnego fragmentu o wartości równej k bajtalarów, należy wypisać słowo `NIE`*. W przeciwnym przypadku należy wypisać dwie liczby l oraz r (*$1 \leqslant l \leqslant r \leqslant n$*) oddzielone pojedynczym odstępem, takie że fragment lizaka złożony z segmentów o numerach od l do r włącznie ma wartość dokładnie k bajtalarów. Jeśli istnieje wiele możliwych odpowiedzi, Twój program może podać dowolną z nich.*

Przykład

Dla danych wejściowych:	*poprawnym wynikiem jest:*
`5 3`	`1 3`
`TWTWT`	`2 2`
`5`	`NIE`
`1`	
`7`	

Wyjaśnienie do przykładu: *Przykład opisuje lizak z rys. 1. Segmenty o numerach od 1 do 3 tworzą lizak postaci* `TWT`*, wart 5 bajtalarów. Segment numer 2 ma smak waniliowy i kosztuje 1 bajtalara. Z tego lizaka nie da się w żaden sposób uzyskać lizaka wartego 7 bajtalarów.*

Rozwiązanie

Wprowadzenie

Występujący w zadaniu lizak możemy wyobrazić sobie jako n-elementowy ciąg złożony z jedynek (segmenty waniliowe) i dwójek (segmenty truskawkowe). Musimy umieć odpowiedzieć na m zapytań postaci: czy jakiś spójny (tzn. jednokawałkowy) fragment lizaka ma sumę dokładnie k. Zauważmy, że wartości parametrów n oraz m w zadaniu mogą być dosyć duże (górne ograniczenie: 1 000 000).

Rozwiązanie o koszcie czasowym $O(n^2 \cdot m)$

W najprostszym rozwiązaniu na każde zapytanie odpowiadamy niezależnie. Odpowiedź na pojedyncze zapytanie (z parametrem k) wymaga wówczas przejrzenia wszystkich $\frac{n \cdot (n+1)}{2}$ możliwych do wyłamania lizaków, obliczenia ceny każdego z nich i sprawdzenia, czy jest równa k.

Złożoność czasowa takiego rozwiązania wynosi $O(n^3 \cdot m)$. Przy bardziej przemyślanej implementacji można uzyskać czas działania $O(n^2 \cdot m)$, jeśli nie będziemy wyznaczać ceny każdego lizaka od nowa, ale skorzystamy z wcześniej obliczonych wartości. Przykładowo, możemy najpierw rozważyć wszystkie fragmenty lizaka zaczynające się od pierwszego segmentu (łącznie w czasie $O(n)$), następnie fragmenty rozpoczynające się od drugiego segmentu itd.

Rozwiązanie to uzyskiwało na zawodach około 14 punktów. Jego implementacja znajduje się w plikach `lizs0.c` i `lizs1.pas`.

Rozwiązanie o koszcie czasowym $O(n^2 + m)$

Powyższe rozwiązanie można usprawnić, przeglądając na wstępie wszystkie możliwe do wyłamania lizaki i zapamiętując, dla każdej możliwej ceny k z zakresu od 1 do $2n$, jeden z przedziałów reprezentujących fragment lizaka o koszcie k, oczywiście jeśli taki przedział istnieje. Umożliwia to późniejsze odpowiadanie na zapytania w czasie stałym.

Rozwiązanie to zostało zaimplementowane w plikach `lizs2.c` i `lizs3.pas`. Na zawodach uzyskiwało około 29 punktów.

Rozwiązanie wzorcowe

Rozwiązanie wzorcowe opiera się na następującej obserwacji.

Fakt 1. *Mając dany fragment lizaka $[l, r]$ o koszcie $k \geqslant 3$, możemy w czasie stałym wyznaczyć pewien fragment o koszcie $k - 2$.*

Dowód: Jeżeli pierwszy lub ostatni segment naszego fragmentu jest truskawkowy (ma koszt równy 2), to wystarczy go odłamać. Jeżeli oba końce są waniliowe, odłamujemy obydwa. ■

Bezpośrednio z powyższego faktu otrzymujemy następujący wniosek:

Wniosek 1. Jeżeli znamy fragment lizaka o maksymalnym koszcie parzystym i fragment o maksymalnym koszcie nieparzystym, jesteśmy w stanie w czasie $O(n)$ wyznaczyć fragmenty o wszystkich możliwych wartościach.

Zauważmy, że jednym z dwóch fragmentów wymienionych we Wniosku 1 jest zawsze cały lizak. Drugim natomiast jest najdłuższy fragment zawierający o jeden segment waniliowy mniej niż cały lizak. Taki fragment można znaleźć w czasie liniowym, wyszukując pozycje skrajnych segmentów waniliowych w lizaku, l i r, i wybierając dłuższy z fragmentów $[l + 1, n]$ oraz $[1, r - 1]$. Dodajmy dla jasności, że jeśli lizak nie zawiera segmentów waniliowych, to podane przedziały nie istnieją (wszystkie fragmenty lizaka mają parzyste ceny).

W ten sposób przed wczytaniem zapytań zapamiętujemy wszystkie możliwe odpowiedzi (podobnie jak w drugim rozwiązaniu nieoptymalnym), a potem odpowiadamy na zapytania w czasie stałym.

Poniżej ten algorytm zapisany w pseudokodzie. Opisy poszczególnych segmentów lizaka przechowujemy w nim w tablicy $smak[1..n]$, natomiast do zapamiętywania wyników wstępnych obliczeń wykorzystujemy tablicę $przedział[1..2n]$, której wszystkie elementy są początkowo ustawione na **nil**.

```
1: procedure Spamiętaj(l, r, k)
2: begin
3:   przedział[k] := [l, r];
4:   if k ⩾ 3 then begin
5:     if smak[l] = T then Spamiętaj(l + 1, r, k − 2)
```

```
6:      else if smak[r] = 'T' then Spamiętaj(l, r − 1, k − 2)
7:      else Spamiętaj(l + 1, r − 1, k − 2);
8:    end
9:  end
10:
11: begin
12:   Wczytaj(n, m, smak);
13:   cena := 0;
14:   for i := 1 to n do
15:     if smak[i] = 'W' then cena := cena + 1
16:     else cena := cena + 2;
17:   Spamiętaj(1, n, cena);
18:   l := −1;
19:   r := −1;
20:   for i := 1 to n do
21:     if smak[i] = 'W' then begin
22:       if l = −1 then l := i;
23:       r := i;
24:     end
25:   if l ≠ −1 and r < n − l + 1 then
26:     Spamiętaj(l + 1, n, cena − 2 · l + 1)
27:   else if r ≠ −1 then
28:     Spamiętaj(1, r − 1, cena − 2 · (n − r) − 1);
29:   for i := 1 to m do begin
30:     Wczytaj(k);
31:     if (k > 2 · n) or (przedział[k] = nil) then Wypisz(„NIE")
32:     else Wypisz(przedział[k]);
33:   end
34: end
```

Złożoność czasowa rozwiązania wzorcowego to $O(n + m)$. Jego implementację można znaleźć w plikach `liz.c` i `liz0.pas`.

Co dalej?

Kluczem do rozwiązania zadania okazał się fakt, że wszystkie segmenty mają koszty równe 1 lub 2. Pozostawiamy Czytelnikowi poszukiwanie algorytmu działającego w czasie $O(n)$, w przypadku gdy lizaki Bajtazara mogą mieć również segmenty wiśniowe, kosztujące 3 bajtalary. Ciekawostką niech będzie fakt, że gdyby koszty poszczególnych segmentów mogły być jeszcze większe (ale ograniczone z góry przez stałą), zadanie można by rozwiązać w złożoności czasowej $O(n \log n)$, stosując jednakże dość zaawansowaną technikę zwaną szybką transformatą Fouriera — opis tego algorytmu można znaleźć np. w książce [22].

Testy

Zadanie było sprawdzane na 12 zestawach danych testowych, z których każdy zawierał trzy pojedyncze testy.

Nazwa	n	m	Opis
liz1a.in	1	4	minimalny test
liz1b.in	7	3	prosty test poprawnościowy
liz1c.in	10	4	mały test poprawnościowy
liz2a.in	25	8	mały test poprawnościowy
liz2b.in	50	40	mały test losowy
liz2c.in	70	300	mały test losowy, dużo wartości
liz3a.in	200	30	mały test, mało wartości, mało segmentów waniliowych
liz3b.in	1 000	200	mały test, dużo segmentów waniliowych
liz3c.in	2 000	20 000	mały test, dużo wartości, mało segmentów waniliowych
liz4a.in	5 000	10 000	średni test
liz4b.in	6 500	10 000	średni test, mało segmentów waniliowych
liz4c.in	8 000	12 000	średni test, dużo segmentów waniliowych
liz5a.in	20 000	20 000	średni test, mało segmentów waniliowych
liz5b.in	20 000	20 000	średni test, dużo segmentów waniliowych
liz5c.in	20 000	100 000	średni test, segmenty waniliowe daleko od końców lizaka
liz6a.in	50 000	80 000	średni test, mało segmentów waniliowych
liz6b.in	50 000	400 000	średni test, dużo segmentów waniliowych, dużo wartości
liz6c.in	50 000	200 000	średni test, segmenty waniliowe daleko od końców lizaka
liz7a.in	70 000	70 000	średni test
liz7b.in	70 000	100 000	średni test, dużo segmentów waniliowych
liz7c.in	80 000	200 000	średni test, segmenty waniliowe daleko od końców lizaka
liz8a.in	100 000	300 000	duży test, mało segmentów waniliowych
liz8b.in	100 000	300 000	duży test, dużo segmentów waniliowych
liz8c.in	100 000	400 000	duży test, segmenty waniliowe bardzo daleko od końców lizaka

Nazwa	n	m	Opis
liz9a.in	300 000	600 000	duży test, mało segmentów waniliowych, segmenty waniliowe daleko od końców lizaka
liz9b.in	300 000	600 000	duży test, segmenty waniliowe daleko od końców lizaka
liz9c.in	400 000	700 000	duży test, segmenty waniliowe bardzo daleko od końców lizaka
liz10a.in	500 000	500 000	duży test
liz10b.in	500 000	700 000	duży test, dużo segmentów waniliowych
liz10c.in	600 000	800 000	duży test, segmenty waniliowe bardzo daleko od końców lizaka
liz11a.in	900 000	1 000 000	duży test, mało segmentów waniliowych
liz11b.in	900 000	1 000 000	duży test, dużo segmentów waniliowych
liz11c.in	900 000	1 000 000	duży test, segmenty waniliowe bardzo daleko od końców lizaka
liz12a.in	1 000 000	1 000 000	maksymalny test, losowy
liz12b.in	1 000 000	1 000 000	maksymalny test, losowy
liz12c.in	1 000 000	1 000 000	maksymalny test, trzy segmenty waniliowe umieszczone prawie na środku lizaka