



Rozwiązanie zadania F 703.

Załóżmy, że ciśnienie wewnątrz cylindrów zwiększyło się do wielkości p . Wtedy tłoki zostały przesunięte w lewo na odległość d . Z prawa Boyle'a-Mariotte'a mamy, że:

$$p_0 V = p(V - S_1 d + S_2 d),$$

stąd

$$d = \frac{(1 - p_0/p)V}{S_1 - S_2}.$$

W całym artykule $\log n$ oznacza logarytm o podstawie 2.

Problem RMQ

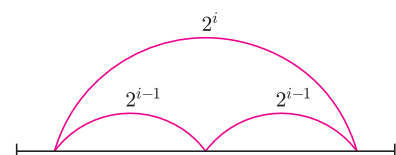
Jakub RADOSZEWSKI*

Przypomnienie. W *Delcie* 9/2007 zajmowaliśmy się dwoma problemami: RMQ i LCA. Przypomnijmy, że problem LCA (*Lowest Common Ancestor*) polega na zbudowaniu struktury danych, która dla danego drzewa pozwoli szybko (w czasie $O(1)$) odpowiadać na zapytania o najniższego wspólnego przodka danych dwóch wierzchołków. Z kolei w problemie RMQ (*Range Minimum Query*) poszukujemy takiej struktury, która pozwoli dla danego ciągu a_1, \dots, a_n szybko odpowiadać na zapytania $RMQ(i, j)$ o minimalny element fragmentu ciągu $a_i, a_{i+1}, \dots, a_{j-1}, a_j$, wyznaczonego przez i -ty i j -ty jego wyraz.

Udowodniliśmy, że te dwa problemy są równoważne, a dowód polegał na pokazaniu liniowego sprowadzenia problemu RMQ do LCA dla pewnego drzewa, a następnie problemu LCA do RMQ dla pewnego ciągu. W tym numerze – zgodnie z zapowiedzią – zaprezentujemy efektywne rozwiązanie problemu RMQ, które na mocy powyższych rozważań będzie jednocześnie rozwiązaniem problemu LCA. W celu uproszczenia dalszego wywodu będziemy zakładać, że długość analizowanego ciągu (n) jest potęgą dwójki; nie jest to duże ograniczenie, gdyż w przeciwnym przypadku możemy ciąg uzupełnić, na przykład, zerami do uzyskania odpowiedniej długości, co nie spowoduje nadmiernego jego wydłużenia.

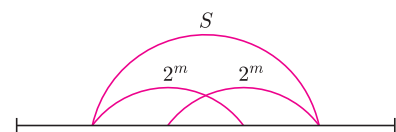
Pierwsze podejście. Na początek skupimy się na spójnych fragmentach ciągu a_1, \dots, a_n (będziemy je odtąd nazywać *segmentami*), których liczba elementów jest potęgą dwójki i spróbujemy dla każdego z nich wyznaczyć minimalny element. Zastanówmy się najpierw, ile jest takich segmentów w naszym ciągu. Liczba różnych potęg dwójki nie większych od n jest rzędu $\log n$, a dla każdej długości mamy co najwyżej n odpowiadających jej segmentów, co daje łączną liczbę $O(n \log n)$.

Segmentów długości 1 mamy n i wynik dla każdego z nich to po prostu zadany element ($RMQ(i, i) = a_i$). Każdy segment długości 2 składa się z dwóch sąsiadujących ze sobą segmentów długości 1, więc dla każdego z nich jako minimum wybieramy mniejsze z minimów odpowiednich segmentów długości 1 ($RMQ(i, i+1) = \min(RMQ(i, i), RMQ(i+1, i+1))$). To postępowanie kontynuujemy dla kolejnych długości: minimum z segmentu długości 2^i liczymy jako minimum z minimów segmentów długości 2^{i-1} , które się składają na rozważany większy segment (rys. 1). Ostatecznie dla każdego z rozważanych segmentów obliczyliśmy wynik w czasie stałym, co daje łączną złożoność tej fazy algorytmu równą $O(n \log n)$.



Rys. 1

Po wykonaniu wstępnych obliczeń możemy teraz udzielać odpowiedzi na zapytania o RMQ dowolnego segmentu S ciągu. Wystarczy znaleźć największą wartość potęgi dwójki 2^m , nie większą od długości rozważanego przedziału (złożoność czasowa $O(\log n)$), a następnie jako wynik przyjąć minimum z wyników dwóch nachodzących się segmentów długości 2^m , które pokrywają cały rozważany segment (rys. 2). Możemy dla każdej długości segmentu od 1 do n obliczyć na wstępie największą wartość potęgi dwójki nie większą od niej w łącznej złożoności czasowej $O(n \log n)$. W ten sposób złożoność czasowa wstępnych obliczeń w takim algorytmie będzie $O(n \log n)$, a każde zapytanie będziemy mogli obsłużyć w złożoności czasowej $O(1)$.



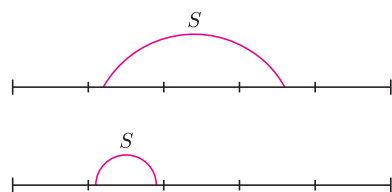
Rys. 2

Otrzymany algorytm jest bardzo szybki i wystarczający dla większości praktycznych zastosowań. My jednak zajmiemy się jego doskonaleniem, dążąc do algorytmu, wykonującego jedynie $O(n)$ wstępnych obliczeń i obsługującego zapytania w czasie stałym.

Właściwy algorytm. W celu rozwiązania problemu RMQ sprowadzamy go w liniowej złożoności czasowej do problemu LCA. Następnie, również w liniowej złożoności czasowej, wykonujemy sprowadzenie otrzymanego LCA z powrotem do RMQ. Nasze postępowanie z pozoru wydaje się nie przynosić żadnych

*student, Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

korzyści, gdyż w jego efekcie znajdujemy się w punkcie wyjścia – przeszliśmy z jednego problemu RMQ do innego. Są to jednak tylko pozory: końcowe zagadnienie RMQ nie jest wszakże tym, od którego rozpoczynaliśmy, a ma jedną bardzo istotną własność: w otrzymanym ciągu różnica kolejnych dwóch elementów wynosi 1 albo -1 . Wynika to z faktu, że wynikowy ciąg składa się z wysokości kolejno odwiedzanych w algorytmie przeszukiwania węzłów drzewa. Z kolei w żadnym kroku przeszukiwania nie występują „przeskoki” – za każdym razem z danego wierzchołka przechodzimy albo do jego ojca, albo do jednego z jego synów.



Rys. 3

Podzielmy otrzymany ciąg a na fragmenciki o długości $\frac{\log n}{2}$ każdy (sposób doboru tej wartości wyjaśni się dalej). Dla każdego fragmencika obliczymy minimum z elementów, które go tworzą i z tych minimów utworzymy nowy ciąg b o długości $\frac{2n}{\log n}$. Każdy segment S w a składa się z pewnej liczby całych fragmencików oraz maksymalnie dwóch kawałków fragmencików – możliwe sposoby takich rozkładów są przedstawione na rysunku 3. Odpowiadanie na zapytania RMQ dla ciągu a możemy zatem w dużej mierze sprowadzić do RMQ dla b , o ile założymy, że będziemy jakoś potrafili osobno zająć się maksymalnie dwoma niepełnymi fragmencikami na brzegach. Dla ciągu b możemy sobie pozwolić na wykonanie wstępnych obliczeń poprzednio otrzymanego algorytmu – złożoność czasowa tego kroku będzie równa $O(m \log m)$, gdzie $m = \frac{2n}{\log n}$ to długość ciągu b . Wykonując podstawienie (i pomijając stały czynnik 2), otrzymujemy złożoność:

$$O\left(\frac{n}{\log n} \log\left(\frac{n}{\log n}\right)\right) = O\left(\frac{n}{\log n} \log n\right) = O(n).$$

W tym szacowaniu istotne było, że długość pojedynczego fragmencika ciągu jest logarytmicznego rzędu, co daje częściowe uzasadnienie takiego a nie innego wyboru jej wartości. W powyższym rozumowaniu pominęliśmy kwestię zamiany opisu segmentu S w ciągu a do segmentu S' w b , odpowiadającego pełnym fragmencikom składającym się na S ; nietrudno jednak zauważyć, że tę operację można wykonać w czasie stałym za pomocą kilku prostych wzorów (dokładny opis pozostawiamy Czytelnikowi).

Pozostał nam do rozpatrzenia sposób wyznaczania minimum z kawałków fragmencików, których maksymalnie dwa otrzymujemy przy każdym zapytaniu. Nie możemy tego kroku wykonywać siłowo, gdyż wówczas obsługa zapytań RMQ wymagałaby pesymistycznie wykonania $\log n$ operacji, czyli nie miałyby żądanej złożoności czasowej $O(1)$. Zastanówmy się więc, ile jest *istotnie różnych* rodzajów fragmencików, jakie mogą powstać przy podziale naszego ciągu. Każdy taki fragmencik jest wyznaczony jednoznacznie przez swój element początkowy oraz przez ciąg złożony wyłącznie z jedynek i minus jedynek, będący ciągiem różnic między kolejnymi elementami fragmencika. Liczba różnych ciągów złożonych z 1 i -1 o długości $\frac{\log n}{2} - 1$ to:

$$2^{(\frac{\log n}{2} - 1)} = \frac{1}{2} \cdot (2^{\log n})^{1/2} = \frac{1}{2} \cdot n^{1/2} = \frac{1}{2} \sqrt{n}.$$

Każdy taki fragmencik możemy opisać jednoznacznie przez jedną liczbę całkowitą między 0 a $\frac{1}{2} \sqrt{n} - 1$, której cyfry w układzie dwójkowym determinują, czy odpowiednie różnice kolejnych elementów fragmencika są równe jeden, czy minus jeden. Dla każdego spójnego kawałka każdego typu fragmencika możemy teraz siłowo wyznaczyć minimum przy założeniu, że pierwszy element fragmencika jest równy 0; koszt czasowy wykonania takiego obliczenia to iloczyn liczby różnych możliwych fragmencików przez sześćcian z długości fragmencika, czyli $O(\sqrt{n}(\log n)^3) = O(n)$. Wszystkie otrzymane wartości możemy – dzięki prostej numeracji różnych fragmencików – spamiętać w trójwymiarowej tablicy (jej wymiary to typ fragmencika oraz początek i koniec jego kawałka), co da nam stały koszt odwołania się do pojedynczej jej komórki. To pokazuje także, skąd się wziął czynnik $\frac{1}{2}$ w dobranej przez nas długości pojedynczego fragmencika: dzięki niemu złożoność czasowa wyznaczania opisanej tablicy pomocniczej nie jest większa niż liniowa.



Rozwiązanie zadania M 1186.

Przyjmijmy, że nieskończony ciąg $a_n = x_n^2$ ($n = 1, 2, \dots$), gdzie $x_n > 0$, spełnia warunki zadania. Liczba

$x_{n+1}^2 - x_n^2 = (x_{n+1} - x_n)(x_{n+1} + x_n)$ jest liczbą pierwszą lub kwadratem liczby pierwszej, a ponadto

$$0 < x_{n+1} - x_n < x_{n+1} + x_n.$$

Wobec tego $x_{n+1} - x_n = 1$, czyli

$$x_n = x_1 + n - 1.$$

Stąd obliczamy

$$x_{n+1}^2 - x_n^2 = x_{n+1} + x_n = 2x_1 + 2n - 1$$

dla $n = 1, 2, \dots$. Zatem gdyby ciąg (a_n) był nieskończony, to każda liczba nieparzysta większa lub równa $2x_1 - 1$ byłaby liczbą pierwszą lub kwadratem liczby pierwszej. Uzyskana sprzeczność dowodzi, że rozpatrywany ciąg musi być skończony.



Na podstawie tak przeprowadzonych wstępnych obliczeń możemy już sobie poradzić z obsługą niepełnych fragmencików. Na początku dla każdego fragmencika w ciągu a wyznaczamy jego typ (liczbę całkowitą od 0 do $\frac{1}{2}\sqrt{n} - 1$) oraz początkowy element. Przy obsłudze zapytania identyfikujemy jeden lub dwa niepełne fragmenciki, składające się na segment S z zapytania, następnie na podstawie ich typów i tego, jakie ich kawałki są zawarte w S , za pomocą pojedynczych odwołań do wyżej skonstruowanej tablicy wyznaczamy szukane minima. Ponieważ były one obliczone przy założeniu, że początkowy element fragmencika jest równy 0, to musimy je przeskalować o faktyczne początkowe elementy rozważanych fragmencików. Za pomocą kilku prostych wzorów możemy z opisu zapytania w czasie stałym wyłuskać potrzebne nam parametry dotyczące skrajnych fragmencików i ich kawałków, które są zawarte w segmencie, zatem cały ten krok może zostać wykonany dla każdego zapytania RMQ w złożoności czasowej $O(1)$. Łączny czas wszystkich wykonanych po drodze wstępnych obliczeń jest liniowy względem n , co pokazuje, że otrzymaliśmy algorytm, działający tak szybko, jak chcieliśmy. Jest to zarazem najszybszy algorytm na jaki można było praktycznie liczyć (nie sposób sobie wyobrazić istnienia algorytmu o asymptotycznie mniejszej niż liniowa złożoności czasowej wstępnych obliczeń), możemy zatem uznać, że znaleźliśmy najlepsze możliwe rozwiązanie zarówno problemu RMQ, jak i LCA.

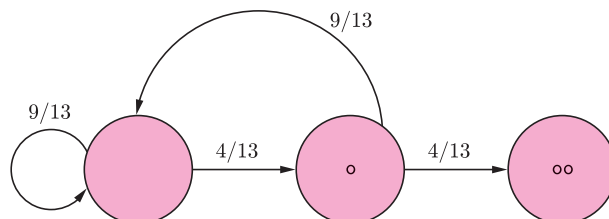
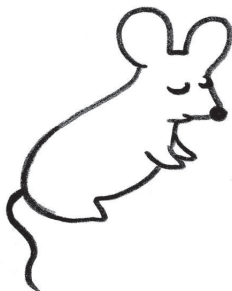
Empiryczne badanie uogólnionej wersji gry Hamleta



Andrzej WALAT

W październikowym numerze *Delty* rozważaliśmy następującą grę. Losujemy litery ze słynnej kwestii Hamleta: *to be or not to be* (być albo nie być) tak długo, aż otrzymamy dwuliterowe słowo *to*; za każde wylosowanie litery płacimy złotówkę, ale na końcu, po uzyskaniu słowa *to* otrzymujemy nagrodę n złotych. Ustaliliśmy, że aby gra była opłacalna, wartość nagrody n powinna być większa niż średnia liczba losowań, jakie trzeba wykonać, by otrzymać słowo *to*. Obliczyliśmy, że teoretyczna wartość tej średniej to $\bar{x} = \frac{13}{3} \cdot \frac{13}{4} = \frac{169}{12} \approx 14,08$.

Tym razem zajmiemy się innymi wariantami gry Hamleta. Na początek przyjmijmy, że naszym docelowym słowem jest *oo* i obliczmy średni czas oczekiwania na to słowo (tj. średnią liczbę losowań, jakie trzeba wykonać, by otrzymać *oo*). W tym przypadku gra Hamleta jest równoważna losowej wędrownicy pionka po planszy przedstawionej na rysunku 1 od pola początkowego x do pola końcowego oo ,



Rys. 1

a odpowiedni układ równań liniowych ma następującą postać:

$$\begin{cases} \bar{x} = 1 + \frac{4}{13}\bar{o} + \frac{9}{13}\bar{x} \\ \bar{o} = 1 + \frac{4}{13} \cdot 0 + \frac{9}{13}\bar{x} \end{cases}$$