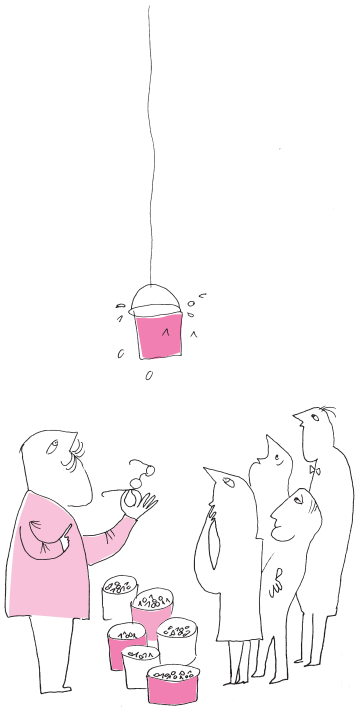
2018 SYMPOZJUM MŁODYCH NAUKOWCÓW WYDZIAŁU FIZYKI 17–18 Września

W dniach 17–18 września 2018 roku na Wydziale Fizyki Uniwersytetu Warszawskiego odbędzie się czwarta edycja Konferencji Krajowej Sympozjum Młodych Naukowców Wydziału Fizyki. Celem konferencji jest umożliwienie studentom studiów I i II stopnia przedstawienia wyników własnych badań naukowych w zakresie fizyki oraz jej zastosowań w pokrewnych dziedzinach. W ramach Sympozjum przewidziane są sesje plenarne z udziałem sześciorga zaproszonych gości, sesje tematyczne wystąpień studentów oraz sesja plakatowa.

Organizatorami Sympozjum są: Studenckie Koło Nanotechnologii „Nanorurki", Samorząd Studentów Wydziału Fizyki UW oraz Wydział Fizyki UW zaś uczestnikami studenci z całej Polski. Szczegółowy program Sympozjum można znaleźć na stronie internetowej Sympozjum `smn.fuw.edu.pl`.

Serdecznie zapraszamy do udziału w Sympozjum!



Informatyczny kącik olimpijski (119): *Problem* (MAX, +) *kontratakuje*

Tym razem wyjątkowo nie omówimy żadnego nowego zadania. Wrócimy za to do problemu przedstawionego w Δ^4_{13} w artykule *Złośliwy problem* (MAX, +) *i kubełkowe struktury danych*. Problem ten wygląda następująco: Dany jest ciąg złożony z n liczb całkowitych $c_1, c_2, \ldots, c_n$. Na tym ciągu chcemy wykonywać dwa typy operacji: (1) przypisanie wartości k każdemu elementowi ciągu o indeksie z przedziału $[i..j]$ i wartości mniejszej niż k (operacja **update**(i, j, k)) oraz (2) zapytanie o sumę elementów ciągu o indeksach z przedziału $[i..j]$ (operacja **quary**(i, j)).

W oryginalnym artykule przedstawiona była struktura danych, która m takich operacji potrafi wykonać w czasie $\mathcal{O}(n + m\sqrt{n}\log n)$. Autor tamtego opracowania (Wojciech Śmietanka) zorganizował wówczas konkurs na poprawę tej złożoności, który udało mi się wtedy wygrać, uzyskując wynik $\mathcal{O}(n\log^2 n + m\log^3 n)$. Niedawno w algorytmicznej społeczności pojawił się jeszcze lepszy pomysł, który przedstawię w dzisiejszym kąciku. Nie dość, że rozwiązuje on nasz problem w czasie $\mathcal{O}((n + m)\log n)$, to jeszcze korzysta wyłącznie ze zwykłego drzewa przedziałowego! (Choć to, jakie konkretnie informacje będziemy przechowywać w każdym przedziale bazowym, jest – według mnie – dość nieintuicyjne.)

Przejdźmy do sedna. W każdym wierzchołku drzewa przedziałowego będziemy utrzymywać następujące informacje o elementach zawartych w jego przedziale bazowym: `min` – minimalna wartość wśród elementów z przedziału; `count_min` – liczba elementów o wartości równej `min`; `second_min` – najmniejsza wartość elementu różna od `min` lub nieskończoność, jeśli wszystkie wartości w przedziale są równe; `sum` – suma wartości elementów; `lazy` – wartość k, o jaką należy zaktualizować dzieci przedziału bazowego, aby wartości w nich przechowywane stały się aktualne, lub **null**, gdy wartości w dzieciach są aktualne.

Po pierwsze, zachęcamy Czytelnika do przekonania się, że z aktualnych wartości dla dzieci przedziału bazowego możemy odzyskać wartości dla ojca. Następnie razem przeanalizujmy, w jaki sposób aktualizujemy konkretny przedział bazowy o nową wartość k. Jeśli $k \leqslant$ `min`, to nie robimy nic. Nawet najmniejsza wartość nie zostanie zmieniona na k. Jeśli `min` $< k <$ `second_min`, to wszystkie elementy o wartości `min` zamienią się na elementy o wartości k, czyli przypisujemy `min` $:= k$ oraz `sum` $:=$ `sum` $+$ `count_min` $\cdot (k -$ `min`$)$. Przypisujemy również `lazy` $:= k$, być może nadpisując mniejszą wartość `lazy`. Jeśli `second_min` $\leqslant k$, to rekurencyjnie aktualizujemy dzieci przedziału bazowego i w czasie stałym odzyskujemy z nich informacje o ojcu.

Otrzymując zapytanie o przedział $[i..j]$, rekurencyjnie rozkładamy go na przedziały bazowe. Jeśli w rekurencji natrafimy na przedział z wartością `lazy` $\neq$ **null**, to – jak wyżej – aktualizujemy jego dzieci. Następnie można odzyskać sumę albo zaktualizować owe przedziały bazowe i odzyskać wartości ich przodków w drzewie.

Sama poprawność algorytmu jest dość oczywista. Jednak skąd bierze się tak dobra złożoność czasowa? Nietrudno skonstruować przykład, w którym jedna operacja **update**(i, j, k) zajmie czas $\mathcal{O}(n)$. Problemem są zejścia rekurencyjne, gdy `second_min` $\leqslant k$. Oznaczmy ich liczbę przez x. Poza nimi wszystko działa jak w zwykłych drzewach przedziałowych, więc nasz algorytm działa w czasie $\mathcal{O}(x + n + m\log n)$.

Przejdźmy do dokładniejszej analizy algorytmu. Weźmy przedział bazowy P. Jeśli w jakimś nadprzedziale przypisaliśmy wartość `lazy` $:= k$, to w nim całym była tylko jedna wartość nie większa niż k, w szczególności k było mniejsze niż `second_min` dla P. Dowodzi to, iż wartości `second_min` zawsze pozostają aktualne. Z tego powodu przy propagowaniu wartości `lazy` nie natrafimy na przypadek `second_min` $\leqslant k$. Całe x pochodzi więc z aktualizacji przedziałów bazowych o nowe wartości k.

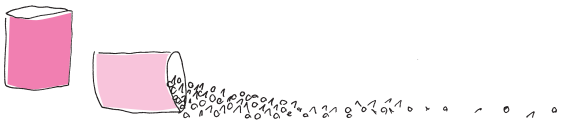
Aby oszacować wartość x w terminach n oraz m, wprowadzimy pojęcie potencjału. Dla danego przedziału bazowego możemy policzyć, ile jest w nim różnych wartości. Potencjałem dla ciągu nazwijmy sumę tych wartości, liczoną po wszystkich przedziałach bazowych. Potencjał ten będzie największy, jeśli wszystkie wartości w ciągu będą różne. Będzie on wtedy wynosił tyle, ile suma długości przedziałów bazowych, czyli $\mathcal{O}(n \log n)$.

Najpierw zastanówmy się, kiedy nasz potencjał może się zwiększyć za względu na dany przedział bazowy P. Przy operacji **update**(i, j, k) zbiór wartości w P może się powiększyć co najwyżej o jeden element – o element o wartości k. Aby tak się stało, musi zachodzić $k >$ `min`, w przeciwnym razie żaden element w P nie otrzyma nowej wartości k. Ponadto przedział $[i..j]$ nie może zawierać całego P, w przeciwnym razie usuniemy wartość `min` ze zbioru wartości z P i potencjał się nie zwiększy. Liczba przedziałów bazowych, które przecinają się z $[i..j]$, a nie są zawarte w $[i..j]$, wynosi $\mathcal{O}(\log n)$, a są to dokładnie te przedziały, które odwiedzamy rekurencyjnie, rozkładając $[i..j]$ na przedziały bazowe. Ze względu na każdy z tych $\mathcal{O}(\log n)$ przedziałów potencjał zwiększy się o co najwyżej jeden.

Nasz potencjał na początku wynosił co najwyżej $\mathcal{O}(n \log n)$, a podczas każdej z m operacji wzrośnie o co najwyżej $\mathcal{O}(\log n)$, a więc może też zmaleć co najwyżej $\mathcal{O}(n \log n) + m \cdot \mathcal{O}(\log n) = \mathcal{O}((n + m) \log n)$ razy.

Wróćmy do naszego problematycznego przypadku, czyli gdy w aktualizowanym o wartość k przedziale zachodzi `second_min` $\leqslant k$. Wszystkie elementy o wartościach `min` oraz `second_min` będę po aktualizacji miały wartość równą k, czyli liczba różnych wartości w aktualizowanym przedziale bazowym zmaleje i zmaleje też potencjał. W ten sposób ograniczyliśmy x przez $\mathcal{O}((n + m) \log n)$, a cały algorytm, tak jak postulowaliśmy, zadziała w czasie $\mathcal{O}((n + m) \log n)$.

Marcin SMULEWICZ



Redaguje Łukasz BOŻYK

Zadania

M 1576. Mając daną siatkę czworościanu, skonstruować punkty styczności sfery wpisanej w ten czworościan do jego ścian.
Rozwiązanie na str. 14

W kolejnych dwóch zadaniach rozważamy trójkąt równoboczny ABC o boku n podzielony na n^2 trójkątów równobocznych o boku 1. Każdy punkt, który jest wierzchołkiem co najmniej jednego z tych n^2 trójkątów, nazwijmy *węzłem*.

M 1577. Wyznaczyć liczbę równoległoboków o wierzchołkach w węzłach, których dwa boki są równoległe do AC, a dwa do BC.
Rozwiązanie na str. 17

M 1578. Wyznaczyć liczbę trójkątów równobocznych o wierzchołkach w węzłach (ale bokach niekoniecznie równoległych do boków ABC).
Rozwiązanie na str. 15

Przygotował Andrzej MAJHOFER

F 959. Struny leżącej poziomo gitary uginają się nieznacznie pod wpływem siły ciężkości. Oszacuj, o ile obniży się środek struny G o częstości podstawowej $f = 196\,\mathrm{Hz}$. Przyspieszenie ziemskie $g \approx 10\,\mathrm{m/s^2}$.
Rozwiązanie na str. 3

F 960. Jak zmieni się wysokość tonu piszczałki organowej (tzw. piszczałka otwarta), dającej dźwięk o częstości $f_0 = 196\,\mathrm{Hz}$ w temperaturze $T_0 = 20°C$, gdy temperatura w kościelnej nawie spadnie do $T_1 = 0°C$?
Rozwiązanie na str. 3