Algorytm magicznych piątek

*Oskar SKIBSKI**

* Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

Wszyscy wiemy, że magia istnieje. Naukowcy starają się tłumaczyć większość zjawisk matematycznymi równaniami lub wymyślają nowe definicje, aby udawać, że to, czego nie da się opisać równaniami, też ma sens. Wiemy jednak dobrze, że udaje im się przekonać tylko tych już przekonanych. Jeżeli królika nie było w kapeluszu, a magik go z niego wyciągnął, to nie ma równania, które by to opisało. Jeden to jeden, a zero to zero.†

† Zawarte w artykule treści są jedynie (błędnymi) poglądami autora i nie muszą odzwierciedlać (i nie odzwierciedlają) poglądów redakcji (przyp. red.).

Jako że w *Delcie* konsekwentnie udajemy, że magia nie istnieje, w tym artykule omówimy algorytm, który wydaje się magiczny, a magiczny nie jest. Możemy porównać go do sztuczki magicznej, w której magik po prostu wyciągnął królika z kapelusza, ale wcześniej nie pokazał widowni, że kapelusz jest pusty. Przez chwilę może nas to zaskoczyć („O! Królik!"). Wnikliwa analiza i logiczne wnioskowanie pozwalają jednak wytłumaczyć tę sztuczkę: królik musiał siedzieć magikowi na głowie przez całe przedstawienie, a przy zdejmowaniu kapelusza magik zręcznie podważył mu łapki i dzięki temu miał co z kapelusza wyciągnąć.

Sztuczka magiczna: Mamy dany kapelusz z n króli... nie, no bądźmy choć trochę poważni! Mamy dany zbiór A zawierający n liczb (liczby mogą się powtarzać). Jeden z widzów podaje dowolną liczbę k od 1 do n. Magik w *czasie liniowym* odpowiada, jaka jest k-ta co do wielkości liczba w zbiorze A.

Jak wykonać sztuczkę: Dla niektórych wartości k zadanie to jest bardzo proste, np. dla $k = 1$ nasz problem sprowadza się do szukania najmniejszego elementu w zbiorze, co łatwo zrobić w czasie liniowym. W ogólności (np. jeżeli $k = \lceil n/2 \rceil$) nie jest jednak jasne, jak możemy to zrobić. Przypadek $k = \lceil n/2 \rceil$ jest zresztą bardzo ważny dla statystyków, gdyż dotyczy tzw. *mediany*, która potrafi mieć dla nich nawet więcej uroku niż średnia. Naturalnym pomysłem jest posortowanie wszystkich liczb i potem wskazanie tej z pozycji k-tej. Najszybsze algorytmy sortowania działają jednak w czasie $O(n \log n)$, a nasz problem wydaje się dużo prostszy niż problem sortowania – jak rozwiązać go w czasie liniowym?

Kiedy piszemy, że czas działania $f(n)$ to $O(g(n))$, to mamy na myśli, że f jest co najwyżej rzędu g, czyli dla dużych wartości n funkcja $f(n)$ rośnie nie szybciej niż $g(n)$. Formalnie: istnieją takie stałe c oraz n_0, że dla $n \geqslant n_0$ zachodzi $f(n) \leqslant c \cdot g(n)$. Jest to zatem oszacowanie górne, ale zwykle nie najlepsze, jakie da się znaleźć.

Pomysł na rozwiązanie pożyczymy ze sztuczki magicznej polegającej na przepiłowaniu asystentki na pół, albo, jak kto woli, z *algorytmu Hoare'a*. Weźmy losowy element m ze zbioru A i podzielmy cały nasz zbiór na dwa niepuste zbiory tak, aby w pierwszym zbiorze znalazły się tylko elementy mniejsze bądź równe m (zbiór $A_{\leqslant}$), a w drugim większe bądź równe m (zbiór $A_{\geqslant}$). Aby zbiory były niepuste, możemy np. do pierwszego z nich wrzucić wszystkie elementy mniejsze bądź równe m i jeżeli drugi zbiór okaże się pusty, przerzucić do niego m. Teraz jeżeli $A_{\leqslant}$ ma co najmniej k elementów, to szukany element musi być w $A_{\leqslant}$ – rekurencyjnie szukamy go zatem w tym zbiorze. Jeżeli z kolei $A_{\leqslant}$ ma mniej niż k elementów, to szukany element jest w zbiorze $A_{\geqslant}$: musimy zatem rekurencyjnie znaleźć tam element $k - |A_{\leqslant}|$ co do wielkości.

Oczytany Czytelnik może mieć (bardzo trafne) skojarzenie z algorytmem QuickSort. W algorytmie tym, aby posortować zbiór liczb, dzielimy go na elementy mniejsze i większe od pewnego elementu, a potem rekurencyjnie sortujemy obie części. Nie jest przypadkiem, że autorem algorytmu QuickSort jest właśnie Tony Hoare.

Pomysł jest prosty, jednak jego skuteczność nie musi być zbyt dobra – jeżeli będziemy pechowo losować zawsze najmniejszy lub największy element, to co krok nasz zbiór będzie się zmniejszał tylko o jedną liczbę. A skoro każdy krok zajmuje czas liniowy, to pesymistyczny czas działania naszego algorytmu będzie kwadratowy: $O(n^2)$. To gorzej, niż gdybyśmy użyli sortowania!

Nasz algorytm możemy jednak poprawić, zmieniając liczbę, względem której dzielimy zbiór. Aby znaleźć wskazaną przez widza kartę, magik zwykle nie zdaje się na los, ale starannie przekłada talię, tak aby kontrolować, gdzie ta karta jest. My zrobimy tak samo – trochę przetasujemy liczby i wyciągniemy taką, która zagwarantuje nam, że żadna z części nie będzie zbyt duża.

Tak właśnie działa *algorytm magicznych piątek*. W dowolny sposób podzielmy nasz zbiór na piątki elementów (ostatnia piątka może być niepełna) i dla każdej znajdźmy jej medianę. Teraz, korzystając rekurencyjnie z naszego algorytmu, znajdźmy medianę median: oznaczmy ją przez m. Podzielmy nasz zbiór na trzy części: elementy mniejsze od m (oznaczane $A_{<}$), elementy równe m (oznaczane $A_{=}$) oraz elementy większe od m (oznaczane $A_{>}$). Teraz jeżeli $|A_{<}| \geqslant k$, to rekurencyjnie szukamy w nim elementu k-tego co do wielkości.

Rozwiązanie zadania F 1076.
Jeśli zaniedbamy efekty brzegowe, to po naładowaniu okładek pojawi się między nimi jednorodne pole elektryczne prostopadłe do powierzchni okładek. Jednorodne też będą gęstości ładunków. Przyjmijmy, że całkowita powierzchnia każdej z okładek wynosi S, wówczas gęstości ładunków wyniosą odpowiednio: $\sigma_1 = Q_1/S$ i $\sigma_2 = Q_2/S$. Na podstawie prawa Gaussa stwierdzamy, że wewnątrz kondensatora każda z okładek jest źródłem pola elektrycznego o natężeniu

$$E_i = \frac{\sigma_i}{2\varepsilon_0}.$$

W powyższym wzorze ε_0 oznacza przenikalność dielektryczną próżni. Pole elektryczne skierowane jest „od okładki", jeżeli jej ładunek jest dodatni, i „do okładki" – jeśli jest ujemny. Wypadkowe pole elektryczne wewnątrz kondensatora jest sumą pól od obu okładek i wynosi:

$$E = E_1 - E_2 = \frac{\sigma_1 - \sigma_2}{2\varepsilon_0} = \frac{Q_1 - Q_2}{2\varepsilon_0 S},$$

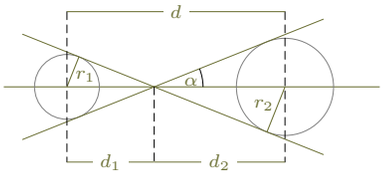
a wartość różnicy potencjałów:

$$U = Ed = (E_1 - E_2)d = \frac{(Q_1 - Q_2)d}{2\varepsilon_0 S} = \\ = \frac{Q_1 - Q_2}{2C}.$$

Rozwiązanie zadania F 1075.
Obserwacje z kierunków pomiędzy dwiema stycznymi do powierzchni gwiazd (rysunek) związane są z pojawianiem się zaćmień. Te styczne przecinają się w punkcie dzielącym odcinek d na odcinki d_1 i d_2, $d = d_1 + d_2$. Odpowiada to zakresowi kątów obserwacji α, dla których

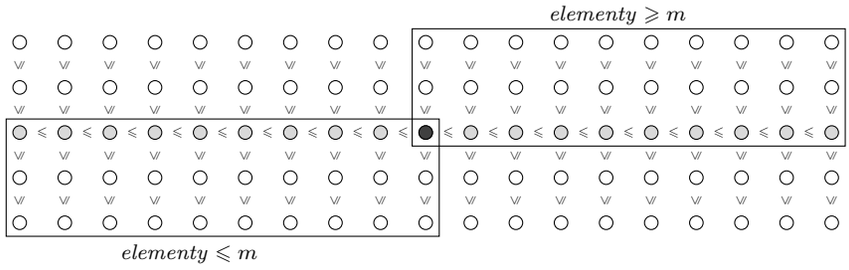
$$|\sin\alpha| \leqslant \frac{r_1}{d_1} = \frac{r_2}{d_2} = \frac{r_1 + r_2}{d}.$$

Rozważany w zadaniu model opisuje najprostszy przypadek układu podwójnego. W ogólnym przypadku wzajemne oddziaływania grawitacyjne mogą prowadzić do odkształceń gwiazd od kształtu kulistego do elipsoidalnego lub nawet do przepływu materii pomiędzy nimi.



Jeżeli $|A_<| < k$, ale $|A_<| + |A_=| \geqslant k$, to znaczy, że k-ty co do wielkości element to po prostu m. Jeśli zaś $|A_<| + |A_=| < k$, to naszego elementu musimy szukać rekurencyjnie w zbiorze $A_>$ – jest on tam $(k - |A_<| - |A_=|)$-ty co do wielkości. Tadam!

Wyjaśnienie sztuczki: No dobrze, ale skąd pewność, że żadna z części nie będzie zbyt duża, a czas działania liniowy? Zauważmy, że w połowie piątek mediana jest mniejsza bądź równa m. W każdej takiej piątce co najmniej połowa elementów jest mniejsza bądź równa m. Wynika z tego, że co najmniej $1/4$ wszystkich elementów jest mniejsza bądź równa m. To oznacza, że elementów większych od m jest nie więcej niż $3/4$ wszystkich. Analogicznie, elementów mniejszych od m jest też nie więcej niż $3/4$ wszystkich. Oznacza to, że nieważne, który przypadek wystąpi, rekurencyjnie wywołamy nasz algorytm na zbiorze zmniejszonym o co najmniej 25%.



Czy to wystarcza, aby nasz algorytm działał w czasie liniowym? Okazuje się, że tak! Oznaczmy czas wykonania naszego algorytmu przez $T(n)$. Podzielenie na piątki i wybranie z każdej mediany wykonujemy w czasie liniowym – $O(n)$. Wybranie mediany median zajmie nam czas $T\left(\lceil \frac{n}{5} \rceil\right)$. Ostatnim krokiem jest wywołanie rekurencyjne dla którejś z części. Jak już uzasadniliśmy, żadna z części nie jest większa niż $3/4$ całości, a zatem czas to $T\left(\lceil \frac{3n}{4} \rceil\right)$. Dostajemy więc następujące oszacowanie:

$$T(n) \leqslant O(n) + T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\left\lceil \frac{3n}{4} \right\rceil\right).$$

Jak się okazuje, z oszacowania tego wynika, że czas naszego algorytmu jest liniowy! Kluczowa jest tu nierówność $\frac{1}{5} + \frac{3}{4} = \frac{19}{20} < 1$, która zapewnia, że $T(n)$ nie rośnie zbyt szybko.

Żeby to zobaczyć, weźmy najpierw za n pewną potęgę liczby 20, aby ładnie dzieliła się przez 4 i 5. Wiemy, że czas potrzebny na podzielenie na piątki i wybranie median możemy oszacować z góry przez cn dla pewnego c naturalnego. Łatwo pokazać przez indukcję, że zachodzi: $T(n) \leqslant 20cn$:

$$T(n) \leqslant cn + T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) \leqslant cn + 20cn\left(\frac{1}{5} + \frac{3}{4}\right) = 20cn.$$

Z oszacowania dla potęg 20 dostajemy natychmiast oszacowanie dla innych liczb: Weźmy dowolne n i dobierzmy k tak, aby $20^k < n \leqslant 20^{k+1}$. Skoro T jest funkcją niemalejącą, to możemy oszacować $T(n)$ przez $T(20^{k+1})$. Ale wiemy, że $n > 20^k$, więc

$$T(n) \leqslant T(20^{k+1}) \leqslant 20c \cdot 20^{k+1} \leqslant 20^2 cn.$$

Skoro istnieje taka stała 20^2c, że dla dowolnego n zachodzi $T(n) \leqslant 20^2c \cdot n$, to oznacza, że nasz algorytm działa w czasie $O(n)$.

Na koniec pozostaje zadać pytanie – czemu akurat piątki? Naturalne wydaje się wzięcie liczby nieparzystej (aby istniały wartości środkowe), ale czy nie mogliśmy wziąć trójek albo siódemek? Albo jedenastek?

Okazuje się, że trójek wziąć nie mogliśmy: musielibyśmy najpierw znaleźć medianę $1/3$ wszystkich liczb, a problem zredukowalibyśmy znowu o $1/4$. Ponieważ $1/3 + 3/4 > 1$, to odpowiednia rekurencja dałaby czas $O(n \log n)$. Moglibyśmy natomiast wziąć siódemki, dziewiątki itd.: Szukanie mediany wśród $1/7$ czy $1/9$ liczb byłoby nawet szybsze niż szukanie ich wśród $1/5$ liczb. Wzrósłby nam jednak koszt wyznaczania median (czyli to enigmatyczne c w powyższym dowodzie) i implementacja byłaby bardziej złożona. W algorytmie użyte są więc piątki, bo pięć jest najmniejszą liczbą nieparzystą większą niż cztery. Ot, cała magia.