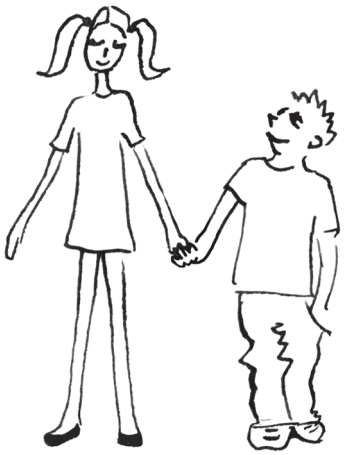
Zliczamy inwersje

Michał ADAMASZEK



Autorem przedstawionego tutaj algorytmu zliczania inwersji jest prof. Wojciech Rytter z Instytutu Informatyki Uniwersytetu Warszawskiego.

Będziemy badać permutacje zbioru $\{1,\ldots,n\}$, czyli różnowartościowe ciągi długości n o wyrazach z tego właśnie zbioru. Na przykład ciąg:

$$[a_1,a_2,a_3,a_4,a_5,a_6]=[6,3,4,2,1,5]$$

jest permutacją zbioru $\{1,2,3,4,5,6\}$.

Spójrzmy teraz na dowolną parę pozycji w takim ciągu. Niektóre z nich są w „dobrej" kolejności, tzn. $a_i<a_j$ dla $i<j$, a niektóre wręcz przeciwnie. Pary (a_i,a_j), które są w złej kolejności, czyli dla $i<j$ zachodzi $a_i>a_j$, nazywamy *inwersjami*. W przykładowej permutacji zapisanej powyżej w inwersji są pary $(6,3)$, $(6,4)$, $(6,2)$, $(6,1)$, $(6,5)$, $(3,2)$, $(3,1)$, $(4,2)$, $(4,1)$ oraz $(2,1)$. Jest ich razem 10. Właśnie ta wartość – łączna liczba inwersji – jest ważną charakterystyką permutacji, którą chcemy nauczyć się obliczać.

Czytelnicy o bardziej algorytmicznym podejściu do życia mogą poczuć się bardziej usatysfakcjonowani inną definicją. Przypuśćmy, że traktujemy powyższy zapis permutacji jako tablicę, którą chcemy posortować rosnąco. Łączna liczba inwersji w permutacji to wówczas *liczba zamian elementów, jakich dokona algorytm sortowania bąbelkowego*. Przypomnijmy bowiem, że algorytm ten zamienia zawsze sąsiednie elementy, likwidując przy tym dokładnie jedną inwersję.

Zanim przystąpimy do zliczania inwersji, mała obserwacja wstępna. Dla danych dwóch liczb naturalnych x,y zdefiniujmy ciąg:

$$(x_0,y_0):=(x,y)\quad\text{oraz}\quad(x_k,y_k)=\left(\left\lfloor\frac{x_{k-1}}{2}\right\rfloor,\left\lfloor\frac{y_{k-1}}{2}\right\rfloor\right)\quad\text{dla}\quad k\geqslant 1.$$

Na przykład, startując od pary $(59,52)$ otrzymamy kolejno:

$$(59,52)\to(29,26)\to(14,13)\to(7,6)\to(3,3)\to(1,1)\to(0,0)\to\ldots$$

Bez problemu zauważamy, że jeśli $x>y$, to najpierw będziemy otrzymywać pary (x_k,y_k), w których $x_k>y_k$, następnie pary, w których $x_k=y_k$, aż wreszcie od pewnego momentu stale parę $(0,0)$. Zastanówmy się, co dzieje się w *ostatnim* kroku l, w którym $x_l>y_l$.

Przede wszystkim, jeśli $x\geqslant y+2$, to $\lfloor\frac{x}{2}\rfloor>\lfloor\frac{y}{2}\rfloor$. Wobec tego w rozważanym kroku l musimy mieć dokładnie $x_l=y_l+1$. Przypuśćmy teraz, że y_l jest nieparzyste, czyli $y_l=2p+1$, $x_l=2p+2$. Wówczas:

$$x_{l+1}=p+1,\qquad y_{l+1}=p,$$

a zatem wciąż $x_{l+1}>y_{l+1}$, co jest sprzeczne z wyborem l jako ostatniego indeksu o takiej własności. Z kolei jeśli y_l jest parzyste ($y_l=2q$, $x_l=2q+1$), to $x_{l+1}=q=y_{l+1}$, czyli wszystko gra. Reasumując, możemy stwierdzić, że:

• *jeśli $x>y$, to istnieje dokładnie jeden indeks l, dla którego $x_l=y_l+1$ oraz y_l jest parzyste.*

Teraz algorytm zliczania inwersji w permutacji mamy już prawie za darmo. Zademonstrujmy go na przykładzie permutacji

$$[6,3,4,2,1,5].$$

Znajdujemy wszystkie pary (a_i,a_j), w których $i<j$, $a_i=a_j+1$ oraz a_j jest parzyste. W naszym przypadku jest jedna taka para, $(3,2)$. Dzielimy wszystkie wyrazy ciągu przez 2 (całkowitoliczbowo):

$$[3,1,2,1,0,2]$$

i powtarzamy wyszukiwanie par: tym razem są cztery pary $(3,2)$, $(3,2)$, $(1,0)$ oraz $(1,0)$. Dzielimy przez 2:

$$[1,0,1,0,0,1].$$

Tutaj mamy 5 par postaci $(1,0)$. W kolejnym kroku dostaniemy same zera i zakończymy pracę. Zatem wszystkich inwersji jest $5+4+1=10$. Poprawność tego algorytmu wynika oczywiście z ostatniej obserwacji: każdą parę, która jest inwersją, zliczymy dokładnie raz, mianowicie w ostatnim kroku, w którym między wyrazami na rozważanych pozycjach wciąż ma miejsce nierówność ostra.

Pozostaje pytanie, jak zaimplementować pojedynczą iterację algorytmu, to znaczy zliczanie w zadanym ciągu $[a_i]$ wszystkich par $i<j$, dla których $a_i=a_j+1$ oraz a_j jest parzyste. Wystarczy w tym celu, przeglądając tablicę od lewej, zliczać w osobnej tablicy wystąpienia kolejnych wartości, a napotykając parzyste a_j sprawdzać dodatkowo, ile razy pojawiła się dotychczas liczba a_j+1. Jedna faza wymaga więc $O(n)$ operacji, a faz jest dokładnie $\lceil\log_2 n\rceil$.

W tym eleganckim i pomysłowym algorytmie wykorzystywaliśmy własności liczb całkowitych. Można zapytać, jak wygląda algorytm znajdowania liczby inwersji w dowolnym ciągu porównywalnych elementów, na przykład liczb rzeczywistych lub napisów. Innymi słowy pytamy, jak znaleźć liczbę inwersji ciągu w modelu obliczeń, w którym dopuszczalne są jedynie pytania postaci „czy $a_i>a_j$?". Można to zrobić także w czasie $O(n\log n)$, co pozostawiamy Czytelnikowi jako zadanie. Może się przydać struktura danych, która pozwala wstawić nowy element oraz znaleźć liczbę elementów większych od danego w czasie $O(\log n)$.