Architekci i algorytmy (2)

W marcowym numerze *Delty* postawiliśmy następujące pytanie: w jaki sposób uszeregować n budynków o wysokościach danych ciągiem $b_1, b_2, \ldots, b_n$, aby suma różnic wysokości sąsiadujących ze sobą budynków $\sum_i |b_i - b_{i+1}|$ była jak największa. Przyjmijmy dla uproszczenia, że liczba budynków n jest parzysta i równa co najmniej 4. Algorytm znajdujący optymalne uszeregowanie można opisać tak:

Krok 0. Uporządkuj budynki od najniższego do najwyższego. Wysokości budynków utworzą niemalejący ciąg $b_1, b_2, \ldots, b_{2k}$, gdzie $n = 2k$.

Krok 1. „Przełam" ciąg (b_i) między wyrazami b_k i b_{k+1}, otrzymując dwa podciągi: $b_1, b_2, \ldots, b_{k-1}, b_k$ oraz $b_{k+1}, b_{k+2}, \ldots, b_{2k-1}, b_{2k}$.

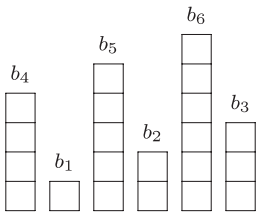
Krok 2. „Przełóż" wyrazy podciągów naprzemiennie, tak że otrzymasz ciąg

$$b_{k+1}, b_1, b_{k+2}, b_2, \ldots, b_{2k}, b_k.$$

Krok 3. Ustaw *w dowolnej kolejności* wyrazy stojące na miejscach nieparzystych (z wyjątkiem pierwszego), tj. wyrazy $b_{k+2}, b_{k+3}, \ldots, b_{2k}$.

Krok 4. Ustaw *w dowolnej kolejności* wyrazy stojące na miejscach parzystych (z wyjątkiem ostatniego), tj. wyrazy $b_1, b_2, \ldots, b_{k-1}$.

Wynik działania algorytmu dla przykładowego ciągu zilustrowano na rysunku 1. Dodajmy pewne uwagi.



Rys. 1. Uporządkowanie generowane przez algorytm dla $n = 6$ budynków o wysokościach $b_i = i$.

Po pierwsze: instrukcja „ustaw w dowolnej kolejności" oznacza, że w wyniku wykonania algorytmu możemy uzyskać $(k-1)! \cdot (k-1)!$ optymalnych uszeregowań. Oczywiście wygenerowane uszeregowania będą wszystkie różne tylko wówczas, gdy wszystkie wyrazy podciągów, o których mowa w krokach 3–4 algorytmu, są różne.

Po drugie: rozwiązań jest tak naprawdę dwa razy więcej (chyba że wszystkie wysokości budynków są jednakowe), gdyż po zapisaniu wyrazów ciągu, wygenerowanego przez algorytm, w odwrotnej kolejności, również otrzymamy optymalne uszeregowanie.

W efekcie zastosowania kroków 0–2 dla ciągu 5, 10, 10, 15, 25, 25, 30 i 45 otrzymamy optymalny ciąg 25, 5, 25, 10, 30, 10, 45, 15. Wszystkich optymalnych rozwiązań, które uzyskujemy wykonując kroki 3–4, jest w sumie 36 (dlaczego?).

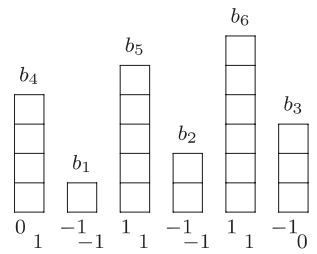
Jak uzasadnić, że przedstawiony wyżej algorytm daje optymalne rozwiązanie, czyli takie uszeregowanie budynków, że suma różnic wysokości sąsiednich budynków będzie największa z możliwych?

Zauważmy, że jeśli dwa budynki b_i i b_j sąsiadują (przy czym $b_i \geqslant b_j$), to ich wkład w sumę różnic wynosi $b_i - b_j$. Zatem dla każdej pary wkład to różnica wysokości wyższego budynku i wysokości niższego budynku. Mamy $n-1$ takich par, a każdy budynek (oprócz skrajnych) należy do dwóch takich par. Zatem sumę różnic można zapisać jako

$$S = \sum_{i=1}^{n} (b_i \cdot x_i + b_i \cdot y_i),$$

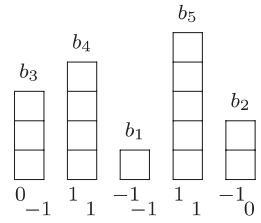
gdzie wśród wartości $x_1, \ldots, x_n$ oraz $y_1, \ldots, y_n$ dokładnie dwie są równe 0, dokładnie $n-1$ wartości jest równych 1 i dokładnie $n-1$ wartości jest równych -1. Można zauważyć, że dla $b_1 \leqslant b_2 \leqslant \ldots \leqslant b_n$, suma S przyjmie maksymalną wartość wtedy, gdy $x_1 \leqslant y_1 \leqslant x_2 \leqslant y_2 \leqslant \ldots \leqslant x_n \leqslant y_n$.

b_i	1	2	3	4	5	6
x_i	−1	−1	−1	0	1	1
y_i	−1	−1	0	1	1	1

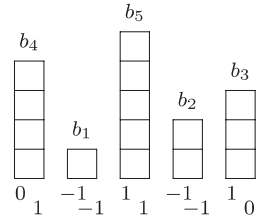


Rys. 2. Pod budynkiem b_i zapisano odpowiadające mu wartości x_i i y_i.

b_i	1	2	3	4	5
x_i	−1	−1	0	1	1
y_i	−1	**0**	**−1**	1	1



b_i	1	2	3	4	5
x_i	−1	−1	**1**	**0**	1
y_i	−1	−1	0	1	1



Rys. 3. Przykładowe porządkowania dla nieparzystej liczby budynków.

Pozostaje wykazać, że istnieje uszeregowanie, które realizuje tę sumę. Łatwo się przekonać, że jest to, na przykład, uporządkowanie uzyskane w kroku 2 algorytmu. Wówczas wartości x_i i y_i dla $i \leqslant k-1$ są równe -1, wartość $x_k = -1$, wartości y_k i x_{k+1} są równe 0, wartość $y_{k+1} = 1$ oraz wartości x_i i y_i dla $i \geqslant k+2$ są równe 1 (rys. 2). Tak więc

$$S = (b_{k+1} - b_1) + (b_{k+2} - b_1) + (b_{k+2} - b_2) + \ldots + (b_{2k} - b_k).$$

Zauważmy również, że permutacje, które wykonujemy w krokach 3–4 algorytmu, nie zmieniają wartości x_i i y_i dla indeksów $i \leqslant k-1$ oraz $i \geqslant k+2$, zatem nie psują optymalności rozwiązania.

Jako zadanie dla Czytelników pozostawiamy uzasadnienie, że algorytm generuje wszystkie możliwe optymalne uszeregowania budynków, czyli że wyrazy podciągów, o których mowa w kroku 1, muszą występować naprzemiennie. (Można tu skorzystać z obserwacji, że permutacje z kroków 3–4 są jedynymi, które nie zmieniają wartości x_i i y_i, i popatrzeć, co się dzieje z sumą S, jeśli któreś z wartości zostaną zmienione.)

Czytelnik Wnikliwy może zapytać: a co w przypadku, gdy n jest liczbą nieparzystą? Czy przedstawiony wyżej algorytm można w prosty sposób zaadaptować dla takiej sytuacji? Okazuje się, że tak! W jaki sposób to zrobić?

Gdy $n = 2k+1$, dla pewnego naturalnego $k \geqslant 1$, mamy $2k$ par sąsiadujących budynków, a każdy budynek (oprócz skrajnych) należy do dwóch takich par. Postępując jak poprzednio, sumę różnic można zapisać jako

$$S = \sum_{i=1}^{2k+1} (b_i \cdot x_i + b_i \cdot y_i),$$

gdzie wśród wartości $x_1, \ldots, x_{2k+1}$ oraz $y_1, \ldots, y_{2k+1}$ dwie są równe 0, $2k$ wartości jest równych 1, a $2k$ wartości jest równych -1. Znowu dla niemalejącego ciągu (b_i), suma S przyjmie maksymalną wartość wtedy, gdy $x_1 \leqslant y_1 \leqslant x_2 \leqslant y_2 \leqslant \ldots \leqslant x_{2k+1} \leqslant y_{2k+1}$. Można się jednak przekonać, że dla żadnego k nie istnieje takie uszeregowanie, że wartości x_i i y_i dla $i \leqslant k$ są równe -1, wartości x_{k+1} i y_{k+1} są równe 0, a wartości x_i i y_i dla $i \geqslant k+2$ są równe 1, gdyż wartości 0 muszą być przypisane skrajnym, a więc różnym, budynkom (inaczej oznaczałoby to, że budynek o wysokości b_{k+1} jest skrajnym zarówno z lewej, jak i prawej strony). Kiedy zatem suma S będzie największa? Są trzy możliwości.

(1) Jeśli $b_{k+1} - b_k < b_{k+2} - b_{k+1}$, to maksymalna suma będzie odpowiadać zamianie wartości $y_k = 0$ i $y_{k+1} = -1$, a więc np. uszeregowaniu
$$b_{k+1}, b_{k+2}, b_1, b_{k+3}, b_2, \ldots, b_{2k+1}, b_k.$$

(2) Jeśli $b_{k+1} - b_k > b_{k+2} - b_{k+1}$, to maksymalna suma będzie odpowiadać zamianie wartości $x_{k+1} = 1$ i $x_{k+2} = 0$, a więc np. uszeregowaniu
$$b_{k+2}, b_1, b_{k+3}, b_2, \ldots, b_{2k+1}, b_k, b_{k+1}.$$

(3) Jeśli $b_{k+1} - b_k = b_{k+2} - b_{k+1}$, to oba wyżej wymienione uszeregowania będą związane z maksymalną sumą (rys. 3).

Przeprowadzając analogiczne rozumowanie jak w przypadku n parzystego, można pokazać, że wykonując pewne permutacje, możemy otrzymać więcej optymalnych uszeregowań budynków. Gdy wszystkie wysokości budynków są różne, to ich liczba jest równa $2 \cdot k! \cdot (k-1)!$ (lub jeszcze dwa razy większa, jeśli mamy do czynienia z trzecią możliwością z powyższej listy).

Przy okazji: „przełamywanie" ciągu i naprzemienne „przekładanie" wyrazów podciągów może Czytelnikowi przypominać tasowanie kart w grach karcianych. To celne spostrzeżenie! Więcej na temat teorii tasowania kart (i nie tylko) Czytelnik znajdzie w książce Iana Stewarta pt. *Jak pokroić tort i inne zagadki matematyczne*, wydanej w języku polskim w 2012 roku.

Małą Deltę przygotowali Paweł PEREKIETKA i Tomasz IDZIASZEK