Informatyczny kącik olimpijski (70): Dwa zadania z Potyczek

W tym kąciku omówimy dwa zadania z Potyczek Algorytmicznych 2012. Pomimo krótkich rozwiązań, zadania te wymagały od zawodników niemałej dozy pomysłowości.

Treść zadania *Korniki* wygodnie nam będzie streścić w terminologii numizmatycznej: na stole w jednym rzędzie leży n monet o nominałach $a[1], a[2], \ldots, a[n]$. Dwóch graczy na przemian wykonuje ruchy. Ruch polega na zabraniu monety z lewego końca, z prawego końca lub z obu końców rzędu naraz. Wiedząc, że każdy z graczy gra tak, by zmaksymalizować sumę wartości zabranych przez siebie monet, należy wyznaczyć wynik gry (różnicę między zyskiem pierwszego i drugiego gracza).

Przykładowo, dla ciągu czterech monet o nominałach $5, 2, 9, 3$ optymalnym ruchem pierwszego gracza jest zabranie monety o nominale 5. Drugi gracz w swoim ruchu zabiera monety 2 i 3.

Z inną wariacją „gry w monety" Czytelnicy spotkali się w artykule *Numizmatyka dla zachłannych* w numerze 12/2012.

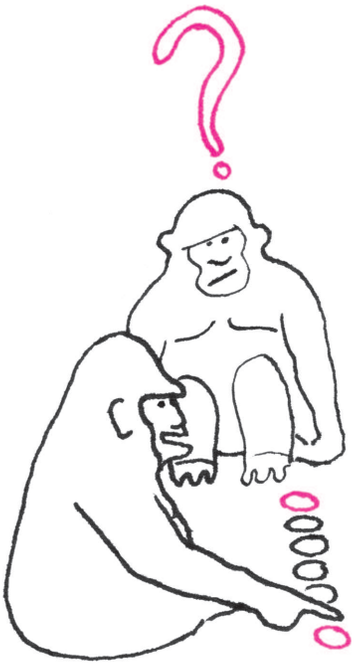
Symulując wprost reguły gry, dostajemy algorytm działający w czasie $O(n^2)$. Wypełniamy kwadratową tablicę, gdzie $d[i, j]$ oznacza wynik gry dla ciągu monet na pozycjach o numerach $i, i+1, \ldots, j$ (zatem $d[1, n]$ jest odpowiedzią do zadania). Jeśli pominiemy warunki brzegowe, rekurencja będzie następująca:

$$(*) \qquad d[i,j] = \max(a[i] + a[j] - d[i+1, j-1], a[i] - d[i+1, j], a[j] - d[i, j-1]).$$

Istnieje jednak efektywniejsze rozwiązanie. Załóżmy najpierw, że n jest nieparzyste. Oznaczmy sumę nominałów monet na pozycjach nieparzystych oraz parzystych przez

$$s_N = a[1] + a[3] + \ldots + a[n], \quad s_P = a[2] + a[4] + \ldots + a[n-1].$$

Zauważmy, że pierwszy gracz ma strategię, która pozwala mu na zabranie wszystkich monet na pozycjach nieparzystych (niezależnie od ruchów drugiego gracza – w swoim pierwszym ruchu zabiera dwie monety, a w każdym kolejnym ruchu kopiuje ostatni ruch drugiego gracza). Gwarantuje mu to uzyskanie wyniku nie mniejszego niż s_N. Z kolei gracz drugi ma strategię, która pozwala mu na zabranie wszystkich monet na pozycjach parzystych (również kopiuje ruchy przeciwnika), co gwarantuje mu uzyskanie wyniku nie mniejszego niż s_P. Zatem obaj gracze uzyskają właśnie tyle, a wynikiem gry będzie $s_N - s_P$.



Dla n parzystego nie widać, jak to zrobić równie prosto. Zatem pozostaje nam skorzystać ze wzoru $(*)$. Ograniczymy się jednak do wyznaczenia wartości $d[i, n+1-i]$ dla $i = 1, 1, \ldots, \frac{n}{2}$. Zauważmy, że przy obliczaniu każdej z nich pierwszy wyraz pod maksimum jest również tej postaci, natomiast dwa pozostałe dotyczą fragmentów długości nieparzystej, z którymi umiemy już sobie łatwo radzić. Rzeczywiście, jeśli stablicujemy sumy prefiksowe ciągu z naprzemiennym znakiem, możemy obliczyć wartość gry dla dowolnego takiego fragmentu w czasie stałym. Ostatecznie otrzymujemy rozwiązanie o złożoności czasowej $O(n)$.

* * *

W zadaniu *Podatek* mamy dany graf nieskierowany G reprezentujący sieć dróg pomiędzy miastami. Każda droga ma przypisaną stawkę podatku. Jadąc z miasta do miasta, musimy w każdym mieście pośrednim uiścić opłatę równą maksimum ze stawki podatku dla drogi, którą wjechaliśmy do miasta, i dla drogi, którą z miasta wyjedziemy. Dla miasta pierwszego i ostatniego rozważamy tylko jedną stawkę. Przykładowo, jadąc trasą v_0, v_1, v_2, v_3, jeśli przez s_i oznaczymy stawkę podatku na drodze pomiędzy v_{i-1} a v_i, zapłacimy

$$s_1 + \max(s_1, s_2) + \max(s_2, s_3) + s_3.$$

Należy podać minimalną opłatę przejazdu pomiędzy dwoma ustalonymi miastami.

Skorzystamy ze wzoru

$$\max(a, b) = \frac{1}{2}(a + b + |a - b|).$$

Możemy wtedy zapisać dwukrotność opłaty, którą uiścimy w naszym przykładzie, jako

$$\begin{aligned} &2s_1 + (s_1 + s_2 + |s_1 - s_2|) + (s_2 + s_3 + |s_2 - s_3|) + 2s_3 = \\ &\qquad = 2(s_1 + s_2 + s_3) + |0 - s_1| + |s_1 - s_2| + |s_2 - s_3| + |s_3 - 0|. \end{aligned}$$

Tworzymy teraz nowy graf G'. Dla każdego wierzchołka v z grafu G, do którego wchodzą krawędzie o wagach $s_1 \leqslant s_2 \leqslant \ldots \leqslant s_k$, tworzymy w grafie G' wierzchołki $(v, s_0 = 0), (v, s_1), (v, s_2), \ldots, (v, s_k)$ oraz łączymy (v, s_i) i (v, s_{i+1}) krawędzią o wadze $s_{i+1} - s_i$. Ponadto, dla każdej krawędzi o wadze s łączącej w G wierzchołki v i u dodajemy w G' krawędź o wadze $2s$ pomiędzy wierzchołkami (v, s) i (u, s).

Pozostaje zauważyć istnienie bijekcji pomiędzy ścieżkami z v do u w grafie G a ścieżkami z $(v, 0)$ do $(u, 0)$ w grafie G', w której dwukrotność opłaty za ścieżkę w G jest równa zwykłej wadze ścieżki w G'. Graf G' ma $O(m)$ wierzchołków i $O(m)$ krawędzi, więc uruchomienie na nim algorytmu Dijkstry zajmie czas $O(m \log m)$.

Tomasz IDZIASZEK