Informatyczny kącik olimpijski (126):
Klocki z numerami i piłeczki na podłodze

W tym odcinku omówimy rozwiązania dwóch zadań z pierwszego etapu XIII Młodzieżowej Olimpiady Informatycznej.

Klocki z numerami: *Danych jest n klocków z zapisanymi na nich numerami. Ciąg $a = (a_1, a_2, \ldots, a_n)$ opisuje numery zapisane na kolejnych klockach. Mówimy, że kolekcja klocków jest imponująca, jeśli każdy naturalny numer x występuje x razy albo nie występuje wcale. Imponującymi kolekcjami są $(3,2,2,3,3)$ i $(4,4,1,4,4)$, zaś nie są: $(1,2,3,4)$ i $(2,2,3,3)$. Możemy zmienić numer dowolnego klocka na dowolny inny numer. Ile minimalnie zmian numerów należy dokonać, aby otrzymać imponującą kolekcję?*

Rozwiązanie $O(n^2)$
Nasze rozważania rozpocznijmy od prostej obserwacji:

W ostatecznym ponumerowaniu żaden klocek nie będzie miał numeru większego niż n.

Na mocy powyższej obserwacji możemy ograniczyć się do numerów nie większych niż n, gdyż tylko takie numery będą w ostatecznym ponumerowaniu. Zliczmy, dla każdego $1 \leqslant i \leqslant n$, liczbę klocków z numerem i (niech w_i oznacza tę wartość). Teraz problem możemy opisać następująco. Mamy n numerów, a i-ty numer ma wartość w_i. Intuicyjnie, wartość numeru oznacza, ile już klocków z tym numerem mamy. Naszym celem jest wybrać najbardziej wartościowy podzbiór numerów sumujących się do n. Wówczas liczba klocków, które należy przenumerować, to n pomniejszone o znalezioną wartość podzbioru.

Otrzymany problem jest podobny do problemu plecakowego (numery są przedmiotami). Z jedną różnicą, chcemy zapełnić plecak „po brzegi", tzn. wybrać podzbiór, którego numery sumują się do n. Opiszemy teraz, w jaki sposób znaleźć taki podzbiór.

Niech $T[j]$, dla $0 \leqslant j \leqslant n$, oznacza wartość najbardziej wartościowego podzbioru o sumie numerów równej j. Początkowo $T[0] = 0$ oraz $T[j] = -\infty$ dla $1 \leqslant j \leqslant n$. Następnie przeglądamy kolejne numery, które mogą tworzyć finalny podzbiór (numery od 1 do n). Załóżmy, że rozpatrujemy numer i o wartości w_i. Chcemy policzyć nowe wartości tablicy T (z uwzględnionym i-tym numerem), oznaczmy te wartości przez T'. Wówczas:

$$T'[j] = \max(T[j], \underbrace{T[j-i] + w_i}_{\text{jeśli } T[j-i] \neq -\infty})$$

($T[j]$ oznacza przypadek, kiedy do podzbioru o sumie j nie bierzemy i-tego oraz wyższych numerów, zaś $T[j-i] + w_i$ oznacza przypadek, kiedy i-ty numer włączamy do podzbioru). Po obliczeniu T', przepisujemy wartości T' do T.

Dla każdego z n przedmiotów przeglądamy $n+1$ elementową tablicę T, zatem otrzymujemy rozwiązanie w złożoności czasowej $O(n^2)$.

Piłeczki na podłodze: *Danych jest n punktów na płaszczyźnie, ponumerowanych od 1 do n, i-ty punkt ma współrzędne (x_i, y_i). Naszym zadaniem jest pokolorować te punkty (każdy punkt na czerwono albo niebiesko). Chcemy, aby odległość pomiędzy najbliższymi dwoma punktami w tym samym kolorze była jak największa.*

Rozwiązanie $O(n^2 \cdot \log(n))$
Stwórzmy graf pełny $G = (V, E)$, gdzie:

- $V = \{1, 2, \ldots, n\}$ – zbiór n wierzchołków,
- $E = \{(u, v) | 1 \leqslant u < v \leqslant n\}$ – zbiór $\frac{n(n-1)}{2}$ krawędzi nieskierowanych.

Niech i-temu punktowi odpowiada i-ty wierzchołek. Wagą krawędzi (u, v) niech będzie kwadrat odległości pomiędzy u-tym i v-tym wierzchołkiem, tj. $w(u, v) = (x_u - x_v)^2 + (y_u - y_v)^2$. Na potrzeby tego rozwiązania wprowadźmy nowe pojęcie: d-porządek, czyli takie przyporządkowanie kolorów wierzchołkom, że waga krawędzi pomiędzy dowolnymi dwoma wierzchołkami w tym samym kolorze wynosi przynajmniej d. Naszym zadaniem jest znaleźć takie największe d, dla którego istnieje d-porządek.

Zauważmy, że d-porządek jest również d'-porządkiem dla $d' < d$. Zatem wartość d możemy znaleźć za pomocą wyszukiwania binarnego. Pozostało nam więc opisać algorytm, który dla ustalonego d sprawdzi, czy istnieje d-porządek. Zbudujmy graf $G' = (V, E')$, gdzie $E' = \{(u, v) | (u, v) \in E \wedge w(u, v) < d\}$. Otrzymaliśmy graf G' z grafu G poprzez usunięcie krawędzi o wadze większej lub równej d. Nie musimy uwzględniać tych krawędzi, ponieważ ich końce mogą być pokolorowane na ten sam kolor. G będzie d-porządkiem, jeśli G' będzie grafem dwudzielnym. Aby sprawdzić, czy G' jest dwudzielny, należy sprawdzić, czy każda spójna tego grafu jest dwudzielna. Sprawdzenie, czy spójna jest dwudzielna, rozpoczynamy od wybrania dowolnego wierzchołka i pokolorowania go na czerwono. Następnie przeszukujemy graf (np. za pomocą algorytmu DFS) i odwiedzanym wierzchołkom przypisujemy kolor przeciwny do koloru poprzednika. Jeśli natrafimy na krawędź, która łączy dwa wierzchołki z przyporządkowanym tym samym kolorem, to wiemy, że graf nie jest dwudzielny. Jeśli natomiast nie wystąpi żadna kolizja, to znaczy, że graf jest dwudzielny.

Liczba krawędzi w G jest rzędu $O(n^2)$. Zatem algorytm wyszukiwania binarnego wykona $O(\log(n))$ faz. W każdej fazie budujemy graf rozmiaru $O(n^2)$ i w czasie liniowym od jego rozmiaru sprawdzamy, czy graf jest dwudzielny. Całkowita złożoność czasowa to $O(n^2 \log(n))$.

Bartosz ŁUKASIEWICZ