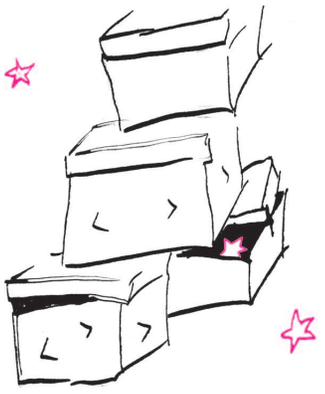
Informatyczny kącik olimpijski (30): Pudełka



Przyjrzyjmy się zadaniu, które pojawiało się już na wielu konkursach w różnych odmianach, sprowadzających się jednak zwykle do tego samego spostrzeżenia. Pierwszą, o której mi wiadomo, była wersja zatytułowana *Stacking Boxes*, którą rozwiązywali uczestnicy zawodów TopCoder w sierpniu 2005 (Single Round Match 251).

Mamy zbiór n pudełek tej samej wielkości, z których każde jest wykonane z innego materiału. Dla każdego z nich znamy jego masę w gramach m_i oraz tzw. wytrzymałość w_i. Ta ostatnia to maksymalna liczba gramów, które można postawić na górze tego pudełka, aby się nie zniszczyło. Chcemy ułożyć z pudełek jak najwyższy stos, tzn. taką konstrukcję, że pudełka stoją jedno na drugim, żadne z nich nie zostaje zgniecione i jest ich jak najwięcej.

Zadanie jest ciekawe chociażby dlatego, że nie wydaje się na pierwszy rzut oka trudne. Po dłuższym zastanowieniu okazuje się rzecz zgoła odmienna: ciężko wymyślić jakiekolwiek rozwiązanie o złożoności wielomianowej względem n.

Zacznijmy więc od jakiegokolwiek poprawnego rozwiązania. Będziemy chcieli dla każdego podzbioru pudełek A obliczyć $W(A)$ – maksymalną liczbę gramów, które można postawić na szczycie jakiegoś stosu złożonego ze wszystkich pudełek z A tak, żeby żadne pudełko z A nie ucierpiało. Zakładamy, że $W(\emptyset) = \infty$. Dla niepustych A musimy wybrać pudełko $a \in A$, które będzie najwyżej w stosie, takie że z pozostałych da się ustawić stos, który się nie zawali, gdy postawimy a na wierzch. Daje to równość:

$$W(A) = \max_{a \in A}\{\min(W(A \setminus \{a\}) - m_a, w_a)\}.$$

Jeśli $W(A) < 0$, to niezależnie od wyboru a,

$$W(A \setminus \{a\}) - m_a < 0,$$

a więc w ogóle nie da się ustawić stosu ze wszystkich pudełek z A. W przeciwnym przypadku mamy żądany wynik. W ten sposób możemy obliczyć wartości $W(A)$ dla wszystkich A będących podzbiorami naszego zbioru pudełek, a następnie znaleźć zbiór A o największej mocy, taki że $W(A) \geqslant 0$, co daje nam końcowy wynik. Takie rozwiązanie ma złożoność czasową $O(2^n \cdot n)$ i zużywa $O(2^n)$ pamięci.

Jak poprawić złożoność? Prawdopodobnie każdy rozwiązujący to zadanie wpadł wcześniej czy później na pomysł, że opłaca się stawiać wytrzymałe pudełka na dole, a te kruche – u góry. Albo, że lepiej jest ciężkie pudełka ustawić w pobliżu dołu stosu, a lekkie wyżej. Żadne z tych zdań nie jest do końca prawdą, można znaleźć kontrprzykłady. Jeśli jednak te zdania połączymy w jedno, niewątpliwie otrzymujemy zdanie prawdziwe: mając pudełka a i b, które chcemy wykorzystać w stosie, takie że $m_a > m_b$ i $w_a > w_b$, pudełko a na pewno lepiej położyć niżej w stosie niż b. Co jednak z przypadkami, gdy jedno z pudełek jest zarówno lżejsze, jak i bardziej wytrzymałe?

Weźmy jakiś zbudowany już stos i pewne dwa sąsiednie pudełka z tego stosu, i oraz j. Niech i będzie niżej w stosie. Zastanawiamy się, kiedy nie można ich zamienić miejscami. Dla wszystkich pozostałych pudełek nie ma znaczenia, czy wyżej jest i, czy j. Niech M oznacza masę wszystkich pudełek ponad j. Skoro i może być niżej, a j nie, to: $w_i \geqslant m_j + M$ oraz $w_j < m_i + M$, czyli

$$w_i + m_i \geqslant m_j + M + m_i > w_j + m_j.$$

Nazwijmy wartość $w_i + m_i$ *współczynnikiem* pudełka i. Stąd: sąsiednich pudełek nie możemy zamienić tylko wtedy, gdy to wyższe ma niższy współczynnik. Jest to kluczowe spostrzeżenie.

Weźmy bowiem najwyższy możliwy do zbudowania stos. Jeśli gdziekolwiek występują dwa sąsiednie pudełka, takie że wyższe z nich ma większy współczynnik, to zamieniamy je miejscami. Takimi zamianami doprowadzamy do stanu, w którym stos jest posortowany według współczynników (duże współczynniki na dole stosu), składa się z tych samych pudełek (a więc nadal jest najwyższy) i dalej się nie zawala. Udowodniliśmy, że istnieje optymalne rozwiązanie, w którym pudełka są posortowane według współczynników.

Jak teraz rozwiązać całe zadanie? Posortujmy nasz zbiór pudełek niemalejąco według współczynników. Udowodniliśmy, że pewien najwyższy stos jest podciągiem tak otrzymanego ciągu pudełek. Chcemy zatem znaleźć najdłuższy podciąg, który jest dobrym stosem. Niech $t(i, d)$ oznacza masę najlżejszego dobrego podciągu, który zawiera się w i pierwszych elementach ciągu i ma długość d (przyjmujemy $t(i, d) = \infty$, jeśli takiego podciągu nie ma). W takim razie $t(0, 0) = 0$, $t(0, d) = \infty$ dla $d > 0$ oraz

$$t(i,d) = \begin{cases} t(i-1,d) & \text{jeśli } w_i < t(i-1,d-1), \\ \min(t(i-1,d-1) + m_i, t(i-1,d)) & \\ & \text{jeśli } w_i \geqslant t(i-1,d-1) \end{cases}$$

dla $i > 0$.

Rozwiązaniem zadania jest największe d, takie że $t(n, d) < \infty$. Złożoność czasowa takiego rozwiązania to $O(n^2)$ przy takim samym zużyciu pamięci. Koszt pamięciowy można jeszcze zredukować do $O(n)$, jeśli zauważymy, że wzór na $t(i, d)$ korzysta tylko z wartości $t(i-1, \cdot)$, a na koniec potrzebujemy tylko wartości $t(n, \cdot)$, tak więc wystarczy zawsze pamiętać dwie ostatnio obliczone „kolumny” tablicy $t(\cdot, \cdot)$.

Tomasz KULCZYŃSKI