Informatyczny kącik olimpijski (35): Cukierki

W tym kąciku spróbujemy rozwiązać zadanie z Olimpiady Informatycznej Środkowej Europy z roku 2004, które to zawody odbyły się w Rzeszowie. Tytuł zadania to *Sweets*, co można przetłumaczyć na polskie *Cukierki*.

Zgodnie z treścią zadania Jaś ma $n \leqslant 10$ słojów z cukierkami, zawierających kolejno $m_1, m_2, \ldots, m_n$ sztuk tychże. Chłopiec chce zjeść co najmniej a, a co najwyżej b cukierków, ale nie wie, ile dokładnie i ile którego rodzaju. Naszym zadaniem jest obliczyć, na ile sposobów może on podjąć decyzję, i podać resztę z dzielenia wyniku przez 2004.

Najprostszym podejściem jest zdecydować osobno, ile wziąć cukierków z każdego ze słojów, a następnie sprawdzić, czy w sumie wzięło się ich między a a b sztuk. W ten sposób uzyskuje się złożoność czasową $O(m_1 m_2 \ldots m_n)$.

Takie rozwiązanie nas nie satysfakcjonuje, możemy więc spróbować je przyspieszyć. Podstawowym pomysłem jest zastosowanie programowania dynamicznego. Niech $t_k[x]$ oznacza liczbę sposobów wzięcia x cukierków z pierwszych k słojów. Wiemy, że $t_0[0] = 1$ i $t_0[x] = 0$ dla $x > 0$. Dla $k > 0$ mamy natomiast zależność rekurencyjną

$$t_k[x] = \sum_{i=0}^{\min(m_k, x)} t_{k-1}[x-i].$$

Zauważmy, że wynikiem jest reszta z dzielenia liczby $\sum_{i=a}^{b} t_n[i]$ przez 2004. Zakładając, że wszystkie działania po drodze będziemy wykonywać modulo 2004, otrzymamy rozwiązanie o złożoności czasowej do $O(b \cdot (m_1 + m_2 + \ldots + m_n))$.

Możemy to jeszcze poprawić, jeśli zauważymy, że sumy potrzebne do obliczenia $t_k[x]$ mają konkretną postać. Chodzi o to, że sumujemy kolejne pola tablicy $t_{k-1}[*]$. W takim razie możemy zastosować tzw. sumy prefiksowe, tzn. po obliczeniu tablicy $t_{k-1}[*]$ obliczyć dodatkowo tablicę $s_{k-1}[*]$ określoną następująco:

$$s_{k-1}[x] = \sum_{i=0}^{x} t_{k-1}[i].$$

Aby to zrobić szybko, korzystamy ze wzorów: $s_{k-1}[-1] = 0$ oraz

$$s_{k-1}[x] = s_{k-1}[x-1] + t_{k-1}[x] \quad \text{dla } x \geqslant 0.$$

Używając takiej tablicy pomocniczej, otrzymujemy wzór

$$t_k[x] = s_{k-1}[x] - s_{k-1}[x - \min(m_k, x) - 1],$$

który pozwala na zmniejszenie złożoności czasowej do $O(bn)$.

Wciąż nie jesteśmy zadowoleni – chcielibyśmy lepiej wykorzystać fakt, że n jest bardzo małe. Po pierwsze, zauważmy, że $\{a, a+1, \ldots, b\} = \{0, 1, \ldots, b\} \setminus \{0, 1, \ldots, a-1\}$, a zatem wystarczy umieć obliczać wynik, gdy chcemy wybrać co najwyżej r cukierków, i dwukrotnie skorzystać z tej metody, dla $r = b$ oraz $r = a - 1$. Przypuśćmy w dalszych rozważaniach, że chcemy wybrać z n słojów łącznie co najwyżej r cukierków i dopuszczamy przy tym wybieranie liczb cukierków przekraczających ograniczenia (m_i). Niech A_i oznacza zbiór tych wyborów cukierków, które łamią ograniczenie m_i na liczbę cukierków wyciągniętych z i-tego słoja. Interesującą nas wartością jest liczba wyborów, które *nie* należą do zbioru $A_1 \cup A_2 \cup \ldots \cup A_n$. Zgodnie z zasadą włączeń i wyłączeń jest to dokładnie:

$$|A| - \sum_{i=1}^{n} |A_i| + \sum_{1 \leqslant i < j \leqslant n} |A_i \cap A_j| - \ldots + (-1)^n |A_1 \cap A_2 \cap \ldots \cap A_n|,$$

gdzie A jest zbiorem wszystkich wyborów co najwyżej r cukierków. Powyższa suma zawiera 2^n składników; zastanówmy się teraz, jak je wszystkie powyznaczać.

Weźmy najpierw zbiór A. Chcemy wybrać co najwyżej r cukierków z n słojów, nie uwzględniając ograniczeń (m_i). Można zauważyć, że jest to równoważne ustawieniu w rzędzie $n + r$ obiektów, z których n jest nierozróżnialnymi ścianami, a r – nierozróżnialnymi kropkami. Mając takie konkretne ustawienie, wybieramy tyle cukierków z pierwszego słoja, ile jest kropek przed pierwszą ścianą, tyle z drugiego, ile kropek między pierwszą a drugą ścianą, itd. Kropki za ostatnią, n-tą ścianą symbolizują cukierki, których nie wybraliśmy z żadnego słoja (jeśli wybieramy mniej niż r). Ustawień ścian i kropek jest $\binom{n+r}{n}$, więc taka jest też moc zbioru A.

Podobne rozumowanie można przeprowadzić dla pozostałych przecięć zbiorów. Jeśli chcemy obliczyć $|A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_k}|$, chodzi nam o wybory r cukierków z n słojów, które łamią ograniczenia dla słojów o numerach $1 \leqslant i_1 < i_2 < \ldots < i_k \leqslant n$. Musimy więc ze słoja i_1 wziąć co najmniej $m_{i_1} + 1$ cukierków, ze słoja i_2 co najmniej $m_{i_2} + 1$, itd. Pozostałe cukierki wybieramy dowolnie, a więc jak wyżej dla A. W ten sposób otrzymujemy wzór

$$|A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_k}| = \binom{r + n - \sum_{j=1}^{k} (m_{i_j} + 1)}{n},$$

przy czym trzeba zauważyć, że gdy $\sum_{j=1}^{k} (m_{i_j} + 1) > r$, to zarówno lewa, jak i prawa strona równania są zerami, a zatem równość zachodzi także w takim patologicznym przypadku.

Ostatecznie sprowadziliśmy problem do obliczenia 2^n symboli Newtona, z których każdy jest postaci $\binom{m}{n}$, przy czym m może być duże. Pozostaje pytanie, jak obliczać takie symbole, a konkretnie ich reszty z dzielenia przez 2004.

Mamy dwie możliwości. Możemy wykonywać operacje na dużych liczbach (co wymaga w językach używanych na zawodach własnej implementacji arytmetyki na takich liczbach) i skorzystać z zależności

$$\binom{m}{n} = \frac{m(m-1)(m-2)\ldots(m-n+1)}{n!}.$$

Zauważmy, że w tym przypadku nie wystarczy pamiętać reszty z dzielenia licznika i mianownika przez 2004 – nie są one wystarczające do obliczenia reszty z dzielenia ilorazu przez 2004. (Inaczej byłoby, gdyby zawody odbywały się w roku 2003 – dlaczego?) Zakładając, że koszt czasowy operacji na liczbach rzędu b jest stały, uzyskujemy rozwiązanie o złożoności czasowej $O(2^n n^2)$.

Nie chcąc wykonywać tak żmudnych prac jak implementacja własnej arytmetyki, możemy zamiast tego spostrzec, że

$$\binom{2k}{n} = \sum_{i=0}^{n} \binom{k}{i} \binom{k}{n-i},$$

co ma łatwe uzasadnienie kombinatoryczne: chcąc wybrać n obiektów spośród $2k$, musimy, dla pewnego i, wybrać i z nich z pierwszej połowy, a $n - i$ z drugiej. Dokładając do tego znaną tożsamość $\binom{2k+1}{n} = \binom{2k}{n} + \binom{2k}{n-1}$, uzyskujemy sposób obliczania wszystkich wartości $\binom{m}{0}, \binom{m}{1}, \ldots, \binom{m}{n}$ w czasie $O(n^2 \log m)$, przy czym operujemy tutaj na samych resztach z dzielenia przez 2004, dzięki czemu nie potrzeba własnej arytmetyki. Daje to ostatecznie rozwiązanie o złożoności czasowej $O(2^n n^2 \log b)$.

Tomasz KULCZYŃSKI