Zadanie: CZE
Czekolada

Warsztaty ILO, grupa olimpijska, dzień 17. Dostępna pamięć: 128 MB.

Rozwiązanie wzorcowe $O(log\ n)$

Zauważmy, że opisaną w zadaniu grę można sprowadzić do gry NIM na czterech słupkach. Powiedzmy, że pole oznaczone krzyżykiem jest w pewnym sensie centrum czekolady. I narysujmy cztery słupki wychodzące z tego pola w każdym z czterech kierunków (góra, dół, lewo, prawo) aż osiągniemy brzeg czekolady. Następnie zauważmy, że każde przełamanie czekolady skraca nam dokładnie jeden z czterech słupków. Przykładowo, przełamanie w poziomie ponad polem oznaczonym skraca nam górny słupek, a przełamanie w pionie na prawo od oznaczonego pola skraca nam prawy słupek. Ponadto, ponieważ możemy przełamać czekoladę w dowolnym miejscu, możemy każdy słupek skrócić o dowolną długość. Dowodzi to, że ta gra to tak naprawdę NIM na czterech słupkach.

Przypomnijmy, że gracz rozpoczynający grę w NIM ma strategię wygrywającą wtedy i tylko wtedy, gdy wartość xor wysokości wszystkich słupków jest niezerowa. Przyjmijmy zatem, że wysokości odpowiednio lewego, górnego, prawego i dolnego słupka oznaczymy przez zmienne l, g, p, d.

Wówczas zauważmy, że niezależnie od wyboru pola oznaczonego, musi zachodzić $l + p + 1 = m$ oraz $g + d + 1 = n$. Jest to prawdziwe, ponieważ l i p reprezentują słupki lewe i prawe, a zatem całą szerokość czekolady (z wyjątkiem pola oznaczonego). Analogicznie w pionie.

Dla uproszczenia zmniejszmy n i m o 1 i teraz szukamy takich czwórek l, p, g, d, że $l + p = m$, $g + d = n$ oraz $l \oplus p \oplus g \oplus d = 0$.

Aby to obliczyć, zastosujemy programowanie dynamiczne.

Będziemy tworzyć zapisy binarne liczb l, p, g, d od najmniej do najbardziej znaczących bitów, tak aby ostatecznie dały nam one poprawny sumy i xory.

Będziemy sobie utrzywać trójwymiarową tablicę $dp[log\ n][2][2]$, taką że $dp[i][v][h]$ mówimy nam, na ile sposobów możemy stworzyć ostatnie i bitów liczb l, p, g, d, tak żeby xor tych i bitów się zerował, a suma $l + p$ dawała na ostatnich i bitach dokładnie takie bity, jakie ma liczba m oraz dawała ona przeniesienie (lub nie, zmienna h to ustala), a suma $g + d$ analogicznie dawała na ostatnich i bitach bity liczby n, a zmienna v mówi, czy ta suma daje przeniesienie do $i + 1$-szego bitu.

Jak tworzyć przejścia w naszym programowaniu dynamicznym. Załóżmy, że mamy obliczone pewne $dp[i][v][h]$. Będziemy chcieli teraz obliczyć kolejny, $i + 1$-szy bit każdej z liczb l, p, g, d. Zatem rozważmy każdy możliwy wariant $i+1$-szego bitu każdej z nich (łącznie mamy 16 wariantów). Niech a, b, c, d będą wartościami $i+1$-szego bitu liczb l, p, g, d. Wiemy, że $(a+b+h)\ mod\ 2$ będzie nowym bitem sumy l i p a $(a+b+h)/2$ będzie wartością przeniesienia do kolejnej pozycji. Analogicznie dla zmiennych pionowych. Dodatkowo musimy zapewnić, że $a \oplus b \oplus c \oplus d$ jest równe 0. Zatem możemy wyznaczyć wszystkie osiągalne stany ze stanu (i, v, h) ustalając $i + 1$-sze bity.

Złożoność jest $O(log\ n)$, ponieważ iterujemy się po liczbie bitów ustalonych, a pozostała wymiary mają stały rozmiar i liczba przejść z każdego stanu jest stała.