Informatyczny kącik olimpijski (123): *Or*

Tym razem omówimy zadanie *Or*, które pojawiło się w 2018 roku na *Junior Balkan Olympiad in Informatics* w Timisoarze (Rumunia).

Zadanie: *Dana jest liczba całkowita p oraz kwadratowa macierz A rozmiaru $n \times n$. W każdym polu macierzy znajduje się jedna liczba całkowita. Z ilu pól składa się najmniejsza prostokątna podmacierz macierzy A, której or bitowy wszystkich elementów wynosi p?*

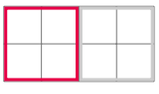
Dla uproszczenia, wartością podmacierzy będziemy nazywali wartość równą *or*'owi wszystkich liczb w tej podmacierzy.

Rozwiązanie $O(n^6)$
Zaczniemy od najbardziej intuicyjnego pomysłu. Rozwiązanie polega na niezależnym sprawdzeniu każdej podmacierzy i wybraniu tej, która ma wartość p oraz jest najmniejsza. Wszystkich podmacierzy jest $O(n^4)$ (lewy górny róg możemy wybrać na $O(n^2)$ sposobów, podobnie prawy dolny róg możemy wybrać na $O(n^2)$ sposobów). Naiwne obliczenie wartości podmacierzy (iterowanie po wszystkich elementach) działa w czasie $O(n^2)$. Zatem całe rozwiązanie działa w czasie $O(n^6)$.

Szybkie obliczanie wartości podmacierzy
Skonstruujemy strukturę danych, która będzie umożliwiała obliczanie wartości dowolnej podmacierzy.

W naiwnym rozwiązaniu iterujemy po każdym polu podmacierzy. Oczywiście takie rozwiązanie jest wolne, dlatego je przyspieszymy. Chcemy, w pewnym sensie, niektóre pola zliczać hurtowo. Zatem obliczymy wartość wszystkich podmacierzy o rozmiarze $a \times b$, gdzie a, b są liczbami postaci 2^k dla $k \in \mathbb{N}$. Zauważmy, że takich podmacierzy w całej macierzy A będzie $O(n^2 \log^2(n))$, ponieważ w każdym polu jest zaczepionych $\log^2(n)$ podmacierzy $(1 \times 1, 1 \times 2, 1 \times 4, \ldots, 2 \times 1, 2 \times 2, 2 \times 4, \ldots, 4 \times 1, 4 \times 2, 4 \times 4, \ldots)$.

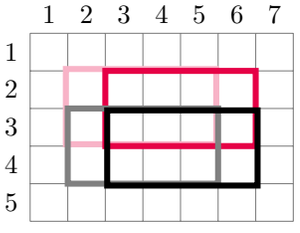
Wartości wyżej opisanych podmacierzy będziemy obliczali od tych, które mają najmniejszą liczbę pól, do tych, które mają największą liczbę pól. Wartość podmacierzy o wymiarach 1×1 to liczba zapisana w tym polu. Wartość podmacierzy o wymiarach $a \times b$ to *or* wartości dwóch mniejszych podmacierzy o wymiarach $a \times \frac{b}{2}$, jeśli $b > 1$ lub *or* wartości dwóch mniejszych podmacierzy o wymiarach $\frac{a}{2} \times b$ w przeciwnym przypadku. Przykładowo: wartością podmacierzy o wymiarach 2×4 jest *or* wartości dwóch mniejszych podmacierzy o wymiarach 2×2 każda.



Konstrukcja struktury zajmuje $O(n^2 \log^2(n))$ pamięci i odbywa się w czasie $O(n^2 \log^2(n))$.

Pozostało nam jeszcze opisanie, w jaki sposób znaleźć wartość dowolnej podmacierzy. Nazwijmy ją M i niech ma wymiary $a \times b$. Jej wartość obliczymy na podstawie znanych wartości innych podmacierzy. Ustalmy takie największe $c = 2^i$ i największe $d = 2^j$, że $i, j \in \mathbb{N}$, $c \leqslant a$ oraz $d \leqslant b$. Weźmy takie cztery podmacierze M o wymiarach $c \times d$, że każde pole M będzie należało do przynajmniej jednej z tych czterech podmacierzy. Wówczas wartością M będzie *or* wartości tych czterech podmacierzy.

Przykład: chcemy obliczyć wartość podmacierzy, której lewe górne pole to $(2, 2)$, a prawe dolne to $(4, 6)$. Zauważmy, że tę podmacierz możemy pokryć czterema podmacierzami o wymiarach 2×4, dla których wyniki znamy. Zatem wartością tej podmacierzy jest *or* wartości czterech mniejszych podmacierzy.



Rozwiązanie $O(n^4)$
Zauważmy, że dzięki powyższej strukturze danych potrafimy w czasie $O(1)$ obliczyć wartość dowolnej podmacierzy. Zatem w naturalny sposób możemy przyspieszyć rozwiązanie z $O(n^6)$ do $O(n^4)$. Wystarczy przejrzeć wszystkie $O(n^4)$ podmacierzy macierzy A i spośród tych o wartości p wybrać najmniejszą.

Rozwiązanie $O(n^3)$
W tym rozwiązaniu iterujemy po każdym przedziale wierszy. W ustalonym przedziale wierszy szukamy najmniejszej podmacierzy o wartości p i wysokości równej liczbie wierszy w tym przedziale. Od teraz możemy o tym myśleć jak o problemie jednowymiarowym, ponieważ wysokość jest ustalona. Szukamy najkrótszego fragmentu, który będzie miał wartość p. Ten problem możemy rozwiązać za pomocą techniki o nazwie „gąsienica". Na początku ustawiamy dwa wskaźniki (początek i koniec gąsienicy) w pierwszej kolumnie. Następnie symulujemy ruch gąsienicy. Jeśli aktualna wartość jest mniejsza od p i może jeszcze osiągnąć p, wtedy rozszerzamy gąsienicę o kolejną kolumnę. W przeciwnym przypadku skracamy gąsienicę. W każdym stanie, za pomocą powyżej opisanej struktury, w czasie $O(1)$ obliczamy wartość aktualnie rozpatrywanej podmacierzy. Dla każdego rozpatrywanego przedziału wierszy gąsienica wykona liniowo wiele ruchów. Wszystkich przedziałów wierszy do rozpatrzenia jest $O(n^2)$, zatem całe rozwiązanie działa w czasie $O(n^3)$.

Bartosz ŁUKASIEWICZ