Lista 4 zadanie 8

Wiktor Hamberger
308982

11 maja 2020

Zacznijmy od opisania maskami bitowymi, jak kamienie mogą być ułożone w jednej kolumnie, okazuje się że możliwości jest 8 (1 oznacza pole z kamieniem, 0 pole bez kamienia):

$$maski[] = \begin{cases} 0000 \\ 1000 \\ 0100 \\ 0010 \\ 0001 \\ 1001 \\ 1010 \\ 0101 \end{cases} \tag{1}$$

Musimy jeszcze zagwarantować, że kamienie nie stykają się między kolumnami. Zauważmy, że jeżeli kamienie się ze sobą stykają pomiędzy kolumnami A i B oraz kolumny są opisane jak wyżej, to wtedy $A\&\&B \neq 0$. Znając te dwa warunki jedyne co pozostaje to wypełnić tabelkę $legalne[maska]$, która dla każdej maski bitowej z (1) będzie zawierała pozostałe maski z którymi dana maska && się do 0. Mając już to zrobione, wystarczy napisać prostego dynamika w którym $dp[i][j]$ będzie oznaczać maksymalną wartość pierwszych i kolumn, przy ułożeniu i-tej kolumny zgodnie z j-tą maską. Dla ułatwienia zdefiniujmy jeszcze funkcję $val(i, j)$, która zwraca sumę z pól zajętych przez kamienie w i-tej kolumnie, przy ustawieniu kamieni zgodnie z j-tą maską.

$$dp[i][j] = \begin{cases} val(i,j) \quad i = 0 \\ max_{k \in legalne[j]} t(i-1,k) \ + \ val(i,j) \quad w\ p.\ p. \end{cases}$$

Wynikiem będzie $max_{j\in(1)}\ dp[n-1][j]$.

```
Data: legalne[], val(), maski[] –jak wyżej, dp[][] - wypełniona zerami
Result: maksymalna suma liczb z pól na których leżą kamyki
Function fun() is
    foreach maska in maski[] do
        dp[0][maska]=val(0, maska);
    end
    for (i=1;i<n;i++) do
        foreach maska in maski[] do
            foreach legalna in legalne[maska] do
                dp[i][maska]:=
                    max(dp[i][maska], dp[i-1][legalna]+val(i, maska));
            end
        end
    end
    return max_{j∈maski[]}(dp[n − 1][j])
end
```

Algorytm wykonuje pętlę od 1 do n-1, a w każdym z obrotów tej pętli wykonuje 8 razy (ilość różnych możliwych masek) co najwyżej 7 (ilość pozostałych, możliwych masek) porównań. Algorytm wykonuje więc $7 * 8 * n$ operacji, więc jego złożoność to $O(n)$.

Dowód poprawności

Przez indukcję. Dla $n = 1$ algorytm wykona tylko pierwszą pętlę i policzy maksa z wszystkich możliwych ustawień kamieni w jednej kolumnie – działa. Załóżmy, że nasz algorytm działa dla wszystkich plansz wielkości $4 \times n$. Teraz po wykonaniu kolejnego algorytmu w kolumnie $dp[n + 1]$ będziemy mieli, zgodnie z założeniem indukcyjnym, maksymalne wartości plansz dla ustawień kamieni w n+1 kolumnie zgodnie z odpowiednią maską. Więc teraz wyciągnięcie maksa w tej kolumny da nam maksymalna suma liczb z pól na których leżą kamyki na planszy $4 \times n + 1$.