Lista 4 z AiSD

Zadanie 8

Kamil Banaś 308262

12 maja 2020

1 Treść

Na każdym polu szachownicy o wymiarach $4 \times n$ znajduje się jedna liczba naturalna. Ułóż algorytm, który umieszcza na szachownicy kamyki w taki sposób, że:

- na każdym polu znajduje się co najwyżej jeden kamień,
- jeśli na polu P znajduje się kamyk, to na polach mających wspólny bok z P nie ma kamyków,
- suma liczb z pól, na których leżą kamyki, jest maksymalna

2 Teoria

Na początku zauważmy, że dla prostokąta 4×1 mamy łącznie 8 możliwości umieszczenia kamyków w sposób zgodni z treścią zadania. Możemy wypisać wszystkie te możliwości i utożsamić każdą z nich z jedną cyfrą w następujący sposób:

0	1	2	3	4	5	6	7
	X				X	X	
		X					X
			X		X		
				X		X	X

gdzie X to pole zajęte przez kamień. W rozwiązaniu zadania możemy dynamicznie przechodzić od lewej strony naszej tablicy i dla każdej kolejnej kolumny obliczyć maksymalne wyniki dla 8 możliwości.

Rozpiszmy dla każdego typu kolumny, jak obliczamy dla niej maksymalny wynik, czyli jakie typy kolumny bierzemy pod uwagę oraz jakie pola z nowej tabeli bierzemy pod uwagę. Przyjmujemy, że wartości w tabeli oznaczamy przez odpowiednio V[0], V[1], V[2], V[3].

- $0 \leftarrow \max(0, 1, 2, 3, 4, 5, 6, 7)$
- $1 \leftarrow \max(0, 2, 3, 4, 7) + \mathrm{V}[0]$
- $2 \leftarrow \max(0, 1, 3, 4, 5, 6) + \mathrm{V}[1]$
- $3 \leftarrow \max(0, 1, 2, 4, 6, 7) + \mathrm{V}[2]$
- $4 \leftarrow \max(0, 1, 2, 3, 5) + \mathrm{V}[3]$
- $5 \leftarrow \max(0, 2, 4, 7) + \mathrm{V}[0] + \mathrm{V}[2]$
- $6 \leftarrow \max(0, 2, 3) + \mathrm{V}[0] + \mathrm{V}[3]$
- $7 \leftarrow \max(0, 1, 3, 5) + \mathrm{V}[1] + \mathrm{V}[3]$

Rozwiązaniem będzie korzystanie z powyższych zależności dla obliczenia maksymalnych wyników idąc od lewej dla każdej kolumny. Widzimy także, że dla obliczeń na danej kolumnie potrzebujemy tylko wyników z poprzedniej kolumny; nie musimy pamiętać wyników dla wcześniejszych kolumn.

3 Algorytm

W algorytmie numerujemy kolumny od 1 do n. Ponadto będziemy korzystać z dwóch procedur, które działają w następujący sposób:

- $newmaxes(i)$ dla $i-$tej kolumny obliczamy maksymalne wyniki dla każdej z 8 możliwości zgodnie z powyższymi zależnościami. Dla kolumny $i-$tej korzystamy z wyników tylko z kolumny $i-1$, natomiast dla kolumny 1 liczymy sumę wartości dla pól zajętych przez kamyki,
- $max(i)$ dla $i-$tej kolumny obliczamy maksymalny wynik, czyli bierzemy maksimum z 8 wyników.

---

procedure DP()
    for $i = 1; i \leqslant n; i \leftarrow i + 1$ **do**
        $newmaxes(i)$
    end for
    return $max(n)$
end procedure

---

4 Uzasadnienie poprawności

Niezmiennikiem w zadaniu będzie fakt, że po i-tym kroku pętli w algorytmie dla każdego typu kolumny mamy obliczony maksymalny wynik dla prefiksu tablicy $4 \times i$.
Dowód: Dla $i = 1$ liczymy same sumy wartości pól zajętych przez kamyki, więc dostajemy poprawne wyniki. Natomiast dla większych i zakładamy, że dla prefiksu $4 \times i - 1$ mamy

obliczone maksymalne wyniki w kolumnie $i-1$. Wobec tego w procedurze $newmaxes(i)$ rozpatrujemy wszystkie możliwości, jaki dany typ kolumny może przyjąć, czyli nie jest możliwe otrzymanie innego wyniku. Zatem dla kolumny i poprawnie obliczymy maksymalne wyniki.

5 Złożoność czasowa i pamięciowa

Widzimy, że procedury $newmaxes(i)$ oraz $max(i)$ działają w czasie stałym, ponieważ wysokość każdej kolumny jest równa 4 przez co wykonujemy stałą liczbę obliczeń i przypisań. Wobec tego złożoność czasowa algorytmu to $O(n)$, gdyż iterujemy po szerokości szachownicy.
Ze względu na to, że wykorzystujemy tylko 16 zmiennych dla trzymania wyników podczas algorytmu to dodatkowa pamięć dla zadania to $O(1)$