AiSD ćwiczenia lista 4

Oskar Bujacz

22 maja 2020

Zadanie 8

Na każdym polu szachwnwicy 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 kamień, to na polach mających wspólny bok z P nie ma kamyków,
- suma liczb z pól, na których leżą kamyki jest maksymalna.

Rozwiązanie

W zadaniu będziemy zakładać, że rozpatrywana szachownica posiada 4 wiersze i n kolumn. W celu rozpatrzenia możliwych kombinacji ułożeń kamyków w poszczególnych kolumnach posłużymy się maskami bitowymi. Istnieje ich 8. 1 oznacza obecność kamyka, 0 jego brak.

nr maski	1	2	3	4	5	6	7	8
wiersz 1	1	0	0	0	1	0	1	0
wiersz 2	0	1	0	0	0	1	0	0
wiersz 3	0	0	1	0	1	0	0	0
wiersz 4	0	0	0	1	0	1	1	0

Rozpatrzmy, które maski mogą występować w kolejnych kolumnach. Stworzymy do tego celu funkcje $prev(i)$, gdzie i to rozważany indeks kolumny. Wartościami będą możliwe maski w kolumnie $i-1$.

nr maski	zbiór wartości
1	{2,3,4,6,8}
2	{1,3,4,5,7,8}
3	{1,2,4,6,7,8}
4	{1,2,3,5,8}
5	{2,4,6,8}
6	{1,3,5,8}
7	{2,3,8}
8	{1,2,3,4,5,6,7,8}

Algorytm

```
wczytaj szachownice do S[4][n]
zainicjalizuj tablicę T[8][n] zerami
for i=0:8 do
        T[i][1] = sum(maska_i(S[:][1]))
for j=0:n do
        for i=0:8 do
                T[i][j] = max_{x∈prev(i)}(T[x][j-1]) + sum(maska_i S[:][j])
return max_{k∈[1,...8]} T[k][n]
```

Poprawność

Będziemy chcieli pokazać, że to co zwraca podany algorytm faktyczenie jest optymalnym rozwiązaniem. Nazwijmy je $OPT(i, j)$, gdzie i to liczba kolumn, a j oznacza zastosowaną maskę w ostatniej kolumnie. Pokażemy zależność.

$$OPT(i,j) = \begin{cases} sum(maska_i(S[:][1])) & \text{dla } i = 1 \\ max_{x \in next(i)}(OPT[x][j-1]) + sum(maska_i S[:][j]) & \text{dla } i > 1 \end{cases}$$

W zależności widać, że problem znalezienie $OPT(i, j)$ można sprowadzić do problemu $OPT(i, j-1)$, zatem problem spełnia kryterium optymalności. Przeprowadzimy dowód indukcyjny.

- Baza indukcji, j=1
  Dla każdego i nakładamy tylko odpowiednią maskę i sumujemy wyrazy w pierwszej kolumnie, zatem jest to rozwiązanie optymalne dla danych $(i, 1)$.

- Krok indukcyjny
  Zakładamy, że $j > 1$ oraz, że wartości $OPT(k, j)$ są optymalne dla $k < m$. Pokażemy, że dla kolumny m rozwiązania również są optymalne. Weźmy dowolne j. Wybrana maska zagwarantuje poprzez funkcję $prev$, że będziemy szukać maksimum w kolumnie $m-1$ tylko po takich polach, które są dozwolone. Zastosujemy maskę j na m i dodamy tylko te wartości, które są dozwolone.
  Pokazuje to, że w $OPT(j, m)$ będzie wynik optymalny. Oczywiście, rozumowanie powtarzamy dla każdego j.

Na koniec algorytm szuka maksimum w ostatniej kolumnie, zatem uwzględnia przy której masce jest największa wartość. Algorytm zwróci poprawny (maksymalny) wynik.

Wybrane pola

W zadaniu mamy odpowiedzieć również na pytanie, jakie pola mamy wybrać przy optymalnym rozwiązaniu. W tym celu stworzymy tablicę $Choice[8][n-1]$ oraz zmienna $LastChoice$, w której będziemy umieszczać informację, dla jakiego x w wewnętrznej pętli algorytmu osiągneliśmy maksimum oraz przed zwróceniem wartości. Wymaga to tylko drobnej modyfikacji.

Wybrane pola możemy od razu wypisywać, przy czym będą one podane od ostatniej kolumny, bądź stworzyć tablicę do ich przechowywania.

Załóżmy teraz, że dla maksimum w kolumnie n została wybrana maska y. Wiemy już, że należy wybrać wyrazy w kolumnie n zgodnie z maską y, a w $Choice[y][n-1]$ mamy zapisane jaką maska

została wybrana w poprzedniej kolumnie. Oznaczmy ją przez y'. Stosujemy maskę y' dla kolumny $n-1$, zapisujmy które pola należą do optymalnego rozwiązania, oraz szukamy w $Choice[y'][n-2]$ kolejnej maski. Postępujemy tak, az nie wrócimy do pierwszej kolumny, czyli wypisania wszystkich wybranych pól.

Złożoność

Złożoność czasowa:
Pierwsza pętla w algorytmie daje nam $O(n)$. Pętla podwójna również daje $O(n)$, ponieważ ilość iteracji wewnętrznej pętli (8) jest niezależna od rozmiaru danych. Tak samo wygląda sytuacja z czasem wykonania funkcji *max* i *sum*, który możemy ograniczyć przez pewną stałą.
Podobnie wygląda sytuacja z czasem odtworzenia wybranych pól. Wykonujemy stałą liczbę operacji dla każdej kolumny tablicy *Choice* o liczbie kolumn równej $n-1$, zatem mamy $O(n)$. Sumując otrzymujemy $O(n)$.
Złożoność pamięciowa
Do przechowania danych z zadania potrzebujemy tablicy o rozmiarze $4 \times n$. Używamy również tablicy rozmiaru $8 \times n$ do pamiętania sumy po maskach, tablicy $8 \times n-1$ do pamiętnia wyborów, ewentualnie tablicę $4 \times n$ do odtworzenia rozwiązania. Suma to $O(n)$.