AiSD Lista 4
wtorek, 4 maja 2021 22:51

Zad 7

7. (1pkt) Rozważmy następujący problem *3-podziału*. Dla danych liczb całkowitych $a_1, \ldots, a_n \in \langle -C..C \rangle$ chcemy stwierdzić, czy można podzielić zbiór $\{1, 2, \ldots, n\}$ na trzy rozłączne podzbiory I, J, K, takie, że
$$\sum_{i \in I} a_i = \sum_{j \in J} a_j = \sum_{k \in K} a_k.$$

Niech $S_J = \sum_{i \in I} a_i$, $S_J = \sum_{j \in J} a_j$, $S_K = \sum_{k \in K} a_k$. Zauważmy, że wtedy

$$S_I + S_J + S_K = S = \sum_{i=1}^{n} a_i .$$

Jeśli chcemy, by $S_I = S_J = S_K$, to $S = 3 S_J \Rightarrow S_J = S_I = S_K = \frac{S}{3}$.

Skoro mamy n elementów $a_k \in [-C, C]$, to $-nC \leq S \leq nC$, zatem

$$\frac{-nC}{3} \leq S_J \leq \frac{nC}{3}.$$

Dodatkowo zdefiniujmy $Ind = \{1, 2, \ldots, n\}$ – zbiór indeksów.

Chcemy sprawdzić, czy da się wyznaczyć takie rozłączne podzbiory I, J, K, że $S_I = S_J = S_K = \frac{S}{3}$.

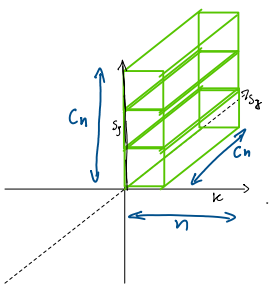
Zanim wejdziemy na kilka obserwacji.

1) Jeśli mamy wyznaczone wartości $S_I = S_J = \frac{S}{3}$, wtedy mamy też wyznaczone S_K, bo $S_K = S - S_I - S_J = S - \frac{2S}{3} = \frac{S}{3}$.
2) Widząc, że mamy skonstruować taki podział dla n elementów, możemy sprawdzić, czy jest to tzw. możliwe dla $n+1$ elementów. Ustalmy dodać $n+1$ element do którejś z sum i sprawdzić, czy dalej spełniony jest warunek.
3) Jeśli liczba elementów to 0, to możemy skonstruować taki podział $Ind = \emptyset \cup \emptyset \cup \emptyset$. Zbiory puste są rozłączne, bo $\emptyset \cap \emptyset = \emptyset$.
4) Jeśli $3 \nmid S$, to nie da się skonstruować takich podzbiorów indeksów.

Zauważmy, że zachodzi następująca zależność rekurencyjna:

$$T[x][y][k] = \begin{cases} \text{true}, & k = x = y = 0 \\ \text{false}, & k = 0 \wedge (x \neq 0 \vee y \neq 0) \\ T[x - a_k][y][k-1] \vee T[x][y - a_k][k-1] \vee T[x][y][k-1] \end{cases}$$

Zadanie można rozwiązać dynamicznie wykorzystując powyższą zależność rekurencyjną i obserwacje. Stworzymy tablicę trójwymiarową wielkości podanej wcześniej współrzędne będą indeksowane po wartościach sum S_I i S_J, a także po indeksie z Ind. Zanim plastry tablicy będziemy wypełniać zerami, co oznacza początek złotego pustego. Będziemy sprawdzać, czy istnieje rozwiązanie dla sum $S_I = x$, $S_J = y$ i $ind = k \in Ind$, jeśli istnieje rozwiązanie dla indeksu $k-1$ i którejś z sum pomniejszonej o element a_k, wtedy dla indeksu k moglibyśmy dodać do tych sum a_k otrzymując rozwiązanie.



Będziemy przechodzić przez tablicę od $sum = 0$ dla $C \cdot k$, zgodnie z powyższą zależnością rekurencyjną. Odpowiedź będziemy mogli odczytać z ostatniej kwadratowej części. $T[Cn][Cn][n]$.

Musimy przejść po każdym polu tablicy trójwymiarowej i wykonać o wiele stałą ilość operacji (zdefiniowanych w zależności rekurencyjnej), zatem złożoność czasowa algorytmu to $O(n^3 C^2)$, bo rozmiar tablicy to $2Cn \times 2Cn \times n$ – mniej więcej sumy, przez które n indeksów. Złożoność pamięciowa to $O(C^2 n^3)$.

Algorytm (nieprawidłowy – można pamiętać tylko dwie płaszczyzny tablicy)

$T[0][0][0] \leftarrow$ tablica o rozmiarach $2nC \times 2nC \times n$ wypełniona na indeksach $T[0][0][0]$.
$T[k] \leftarrow$ tablica tych samych rozmiarów wypełniona fałszami
dla kolejnego $k \in \{1, \ldots, n\}$:
  dla kolejnego $x \in [-ci, ci]$:
    dla kolejnego $y \in [-ci, ci]$:

$T[x][y][k] = T[x][y][k-1] \vee T[x - a_k][y][k-1] \vee T[x][y - a_k][k-1]$

$T[k][x][y] = T[k-1][x][y] \vee (L[k-1] \wedge a[x][y])$

$T[k-1] = T[k]$ // prog increment