AiSD L5

Maurycy Borkowski

12.05.2021

zadanie 1.

Spisane na kartce.

zadanie 2.

Spisane na kartce.

zadanie 3.

Ile mamy hiperpłaszczyzn z odpowiedzią "TAK"?
Wszędzie indziej jest odpowiedź "NIE".

$$\binom{n}{3} = \frac{n!}{(n-3)! \cdot 3!} = \Theta(n^3)$$

Na ile obszarów dzieli przestrzeń n-wymiarową, m-hiperpłaszczyzn (dowód)?

$$\sum_{i=0}^{n} \binom{m}{i}$$

Oszacujemy:

$$\binom{m}{i} = \frac{m!}{(m-i)! \cdot i!} \leqslant \frac{m^i}{i!} \leqslant \frac{m^i}{i}$$

Stąd:

$$\sum_{i=0}^{n} \binom{m}{i} \leqslant \sum_{i=0}^{n} \frac{m^i}{i} \leqslant n \cdot \frac{m^n}{n} = m^n$$

Mamy więc co najwyżej m^n obszarów.

Stąd, nie możemy podzielić przestrzeni wejść na więcej niż
$\Theta(n^3)^n = 2^{\Theta(n \log n)}$ maxymalnych obszarów (liści) z tą samą odpowiedzią "NIE".

zadanie 4.

Adwersarz przygotowuje sobie $2n$ ciągów w następujący sposób (w każdym ciągu zmienia dokładnie jedną parę sąsiednich elementów):

$$
\begin{aligned}
A_0 &= a_1, b_1, a_2, b_2, a_3, b_3, \ldots \\
A_1 &= b_1, a_1, a_2, b_2, a_3, b_3, \ldots \\
A_2 &= a_1, a_2, b_1, b_2, a_3, b_3, \ldots \\
A_3 &= a_1, b_1, b_2, a_2, a_3, b_3, \ldots \\
A_4 &= a_1, b_1, a_2, a_3, b_2, b_3, \ldots \\
&\qquad\qquad\qquad\qquad\qquad\quad \ldots
\end{aligned}
$$

Teraz pokażemy, jak adwersarz może odpowiadać tak, by za każdym razem odrzucał co najwyżej jeden z ciągów. Oczywiście nie będziemy zadawać *głupich* pytań tj. o relacje wewnątrz a_i lub b_j tylko pomiędzy nimi. Rozpatrzmy przypadki:

1. $i > j + 1$
   w każdym z $2n$ ciągów możemy odpowiedzieć, że $a_i > b_j$ nie odrzucając żadnego z nich
2. $i < j$
   w każdym z $2n$ ciągów możemy odpowiedzieć, że $a_i < b_j$ nie odrzucając żadnego z nich
3. $i = j$
   w każdym oprócz jednego ciągu $a_i < b_i$, stąd adwersarz musi odrzucić jeden ciąg
4. $i = j + 1$
   w każdym oprócz jednego ciągu $a_i > b_i$, stąd adwersarz musi odrzucić jeden ciąg

Widzimy, więc, że każde zapytanie do adwersarza odrzuca co najwyżej jeden ciąg. Stąd musimy wykonać $2n - 1$ zapytań (porównań).

zadanie 6.

Wystarcza

Możemy zbudować sobie drzewo *turniejowe*, aby wyłonić zwycięzce musimy wykonać $n-1$ porównań (liczba wierzchołków w drzewie binarnym bez liści). Aby wyłonić srebrnego medaliste, trzeba sprawdzić z wszystkich, którzy przegrali ze zwycięzcą. Ich jest: $\lceil \log n \rceil$ więc żeby wyłonić drugie miejsce musimy wykonać jeszcze $\lceil \log n \rceil - 1$ porównań.
Ostatecznie wykonamy: $n + \lceil \log n \rceil - 2$ porównań.

Potrzeba

Algorytm wyłania:

- element największy, a
- element drugi największy, b
- zbiór pozostałych elementów, R (oczywiście $|R| = n-2$)

Żeby stwierdzić czy dany element ma należeć do R musimy wiedzieć, że jest on mniejszy od b lub od innego elementu z R, stąd trzeba wykonać conajmniej $n-2$ porównania z elementami ze zbioru $\{b\} \cup R$.

Oznaczmy przez $L(x)$ liczbę elementów o których wiemy, że *przegrała* z x.
Adwersarz, gdy porównujemy elementy x, y wybierze ten gdzie daje nam mniej informacji:

$$\begin{cases} x < y \text{ ,gdy } L(x) < L(y) \\ x > y \text{ w p.p.} \end{cases}$$

Wtedy zbiory elementów mniejszych będą się zmieniały następująco:

$$\begin{cases} L(y) = L(x) + L(y) \wedge L(x) = 0 \text{ ,gdy } L(x) < L(y) \\ L(x) = L(x) + L(y) \wedge L(y) = 0 \text{ w p.p.} \end{cases}$$

W najgorszym przypadku (dla adwersarza) po udzieleniu odpowiedzi $L(x)$ zwiększy się dwukrotnie, stąd po k porównaniach, $L(x) \leqslant 2^k$, pod koniec algorytmu $L(a) = n$ stąd:

$$n = L(a) \leqslant 2^k \implies \lceil \log n \rceil \leqslant k$$

czyli potrzeba jeszcze conajmniej $\lceil \log n \rceil$ (kroków=porównań z a).

Porównania potrzebne do wyłonienia a ($\lceil \log n \rceil$) nie dają informacji nic o tym jak postąpić z resztą elementów. Są rozłączne z $(n-2)$ porównaniami, tworzącymi R te porównania stwierdzają mniejszość względem b lub innego elementu z R. Stąd te dwa zbiory porównań są rozłączne.

Ostatecznie potrzeba: $n + \lceil \log n \rceil - 2$ porównań.