Z2|

Pomyślmy o naszych liczbach na wejściu jako o punkcie w przestrzeni $\mathbb{R}^n$.

Nasze drzewo decyzyjne będzie miało na wierzchołkach kombinacje liniowe ciągu wejściowego, a krawędzie do synów będą odpowiadały relacjom <0, $=0$ i >0.

Zauważmy, że $\sum_{i=1}^{n} a_i \cdot x_i = 0$ tworzy nam hiperpłaszczyznę dzielącą naszą przestrzeń na 2 półprzestrzenie oraz hiperpłaszczyznę.

Lemat 1:

Każdy z tych zbiorów jest zbiorem wypukłym. Czyli

$\forall u,v \in C \quad \forall 0 \leq \alpha \leq 1 \quad \alpha u + (1-\alpha) \cdot v \in C.$

Dowód:

Weźmy dowolne u, v które trafiły do C. (jak na rysunku na miro)

To znaczy, że

$$\sum_{i=1}^{n} a_i^3 \cdot u_i > 0 \quad \wedge \quad \sum_{i=1}^{n} a_i^3 \cdot v_i > 0$$

$$\sum_{i=1}^{n} \alpha \cdot a_i^3 \cdot u_i > 0 \quad \wedge \quad \sum_{i=1}^{n} (1-\alpha) \cdot a_i^3 \cdot v_i > 0$$

$$+$$

$$\sum_{i=1}^{n} a_i^3 \cdot (\alpha u_i + (1-\alpha) \cdot v_i)) > 0$$

Czyli:

$$\alpha \cdot u + (1-\alpha) \cdot v \in C$$

Rozważmy taki ciąg liczb na wejściu:

L_1 L_2 L_3

$1, 3, \dots, n-1, 2n, -(2n+2), -(2n+4), \dots, -(2n+n-2)$

Zauważmy, że w takim ciągu nie ma takich 3 liczb, które sumują się do 0.

- gdy x, y z L_1 to $x+y < \text{abs}(\min(L_3))$
- gdy x, y z L_3 to $\text{abs}(x+y) > \max(L_1, L_2)$
- gdy x z L_1, y z L_2 a z z L_3 to $2|y+z$ a $2 \nmid x$ □

Permutując L_1, otrzymamy $(\frac{n}{2})!$ ciągów, z których każdy daje odp nie.

Lemat 2:

Żadne dwa z tych punktów wymiernych przez permutacje nie wylądują w tym samym liściu.

D-d:

Załóżmy nie wprost, że P_i i P_j są w jednym liściu i niech k będzie pierwszą współrzędną, na której te punkty się różnią. BSO niech $P_j(k) > P_i(k)$.

Skorzystajmy z lematu 1 i ustalmy $\alpha = \frac{1}{P_j(k) - P_i(k)}$. Wtedy punkt $p = \frac{1}{P_j(k) - P_i(k)} \cdot P_j + \frac{P_j(k) - P_i(k) - 1}{P_j(k) - P_i(k)} \cdot P_i$ również leży w tym liściu.

Ale zauważmy, że $p(k) = \frac{P_j(k) + P_j(k) \cdot P_i(k) - P_i(k) \cdot P_i(k) - P_i(k)}{P_j(k) - P_i(k)} = \frac{P_j(k) - P_i(k)}{P_j(k) - P_i(k)} +$

$+ \frac{P_i(k) \cdot P_i(k) - P_i(k) \cdot P_i(k)}{P_i(k) \cdot P_i(k)} = 1 + P_i(k)$. a to znaczy, że

algorytm dla p zwraca nie. A powinien tak, bo

$p(k) + 2n + (-(2n + p(k))) = 0$. Sprzeczność. ☐

Zatem pokazaliśmy, że mamy conajmniej $\frac{n}{2}!$ części.

Zatem $\frac{n}{2}! = 3^n$, gdzie h to wysokość drzewa.

Ze wzoru Stirlinga:

$$n! \approx \left(\frac{n}{e}\right)^n$$

$$\log_3 n! \geq n(\log_3 n - \log_3 e) \geq n \log_3 n - 0{,}91 n$$

Czyli wysokość drzewa a zarazem dolne ograniczenie na liczbę porównań to $O(n \log n)$

Zł)

Do tego zadania możemy zastosować podejście z adwersarzem.

Rola adwersarza będzie polegać na tym, aby zmusić gracza do zadania jak najdłuższej liczby zapytań.

Skonstruujmy teraz przestrzeń zdarzeń: (L_i - L_0 z zamienionymi elementami na i i $i+1$ pozycji:)

$L_0 = a_1, b_1, a_2, b_2, \dots, a_n, b_n$

$L_1 = b_1, a_1, a_2, b_2, \dots, a_n, b_n$

$L_2 = a_1, a_2, b_1, b_2, a_3, b_3, \dots, a_n, b_n$

$L_3 = a_1, b_1, b_2, a_2, a_3, b_3, \dots, a_n, b_n$

$\vdots$

$L_{n-1} = a_1, b_1, \dots, b_n, a_n$

Rozważmy teraz możliwe zapytania gracza: (zakładamy, że gracz nie zadaje pytań w stylu $a_i < a_j$?, bo to już wie, czyli interesuje nas tylko $a_i < b_j$, $a_i > b_j$?)

- $i > j+1$

  W takim przypadku adwersarz odpowie, że $a_i > b_j$. Zauważmy, że jest to prawda dla dowolnego L_i, czyli przestrzeń zdarzeń się nie zmniejszy.

- $j > i$

  Teraz adwersarz odpowie $a_i < b_j$ i ewa logicznie jak wyżej nie zmniejszy przestrzeni wydarzeń.

- $i = j$

  Tutaj również adwersarz odpowie $a_i < b_j$, ale przestrzeń wydarzeń zmniejszy się o 1, ponieważ ta sytuacja zachodzi przy L_i, w którym zamieniamy a_i i b_j miejscami

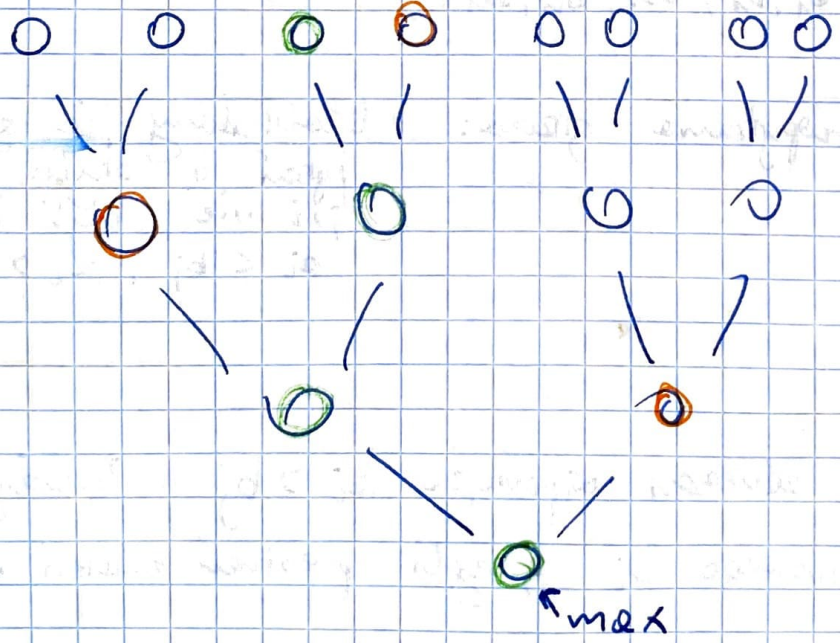
- $i < j+1$

Tutaj adwersarz odpowiada $a_i > b_j$, i usuwa jedno zdarzenie odpowiadające zamianie a_i i b_{i-1} miejscami.

Pokorabiśmy zatem, że dla dowolnego zapytania usuwamy max 1 zdarzenie, czyli musimy wykonać co najmniej $2n-1$ zapytań.

Z6)

- wystarczy $n + \lceil \log n \rceil - 2$:



$n-1$ porównań na maxa

$\lceil \log n \rceil - 1$ - na pozostałe maxa

czyli sumarycznie $n + \lceil \log n \rceil - 2$ porównań

• Potrzeba $n + \lceil \log n \rceil - 2$ porównań:

Niech M_1 to największy element, M_2 przed największy a S to zbiór $n-2$ pozostałych elementów wyznaczone przez algorytm.

Zauważmy, że elementy ze zbioru S musiały być mniejsze porównane albo od M_2 albo od innego elementu z S, czyli minimalnie algorytm wykonał $n-2$ porównań.

Zastanówmy się ile jeszcze porównań musiał wykonać algorytm, aby wyznaczyć M_1.

Zdefiniujmy funkcję $L(x)$, która mówi nam ile do tej pory algorytm nie elementach $\leq x$. Teraz rozważmy adwersarza, który gdy porównujemy elementy $x : y$, powie nam, że

$$\begin{cases} x < y, \text{ gdy } L(x) < L(y) & (1) \\ x > y, \text{ w p.p.} & (2) \end{cases}$$

po takim kroku $\begin{cases} L(y) = L(x) + L(y), & L(x)=0 \quad (1) \\ L(x) = L(x) + L(y), & L(y)=0 \quad (2) \end{cases}$

Zauważmy, że $L(x)$ po k porównaniach $x \leq 2^k$, bo za każdym razem $L(x)$ zwiększa się max dwukrotnie.

Wiemy, że po zakończeniu algorytmu $L(M_1) = n$, czyli

$$n \leq 2^k$$

$\lceil \log n \rceil \leq k$, czyli potrzeba jeszcze conajmniej $\lceil \log n \rceil$ kroków, co razem daje $n + \lceil \log n \rceil - 2$. □