Szachy

PREOI 2025
Dzień 6 – 30 stycznia 2025

Kod zadania: **coi**

Podzadanie 1 - $1 \leq n \leq 7$, $1 \leq m \leq 20$

Przeiterujmy się po wszystkich $n!$ kombinacjach rang graczy. Dla każdej sprawdźmy czy jest poprawna. Dla poprawnej kombinacji, zanotujmy dla każdego gracza jego możliwy poziom siły. Jeżeli sumarycznie jakiś gracz, ma dokładnie jedną możliwą rangę to znamy jego prawdziwą rangę. Możemy puścić taki algorytm m razy dla każdego prefiksu meczy i dla każdego gracza poznać minimalny mecz, po którym jego ranga jest znana.

Podzadanie 2 - $1 \leq n \leq 100$, $1 \leq m \leq 400$

Wykorzystamy obserwację: da się określić że szachista x ma niższą rangę od szachisty y wtedy i tylko wtedy, gdy istnieje ciąg szachistów $x\ a_1\ \ldots\ a_n\ y$, że pomiędzy każdą parą sąsiadujących szachistów a, b, został rozegrany mecz między nimi i a przegrał.
Utwórzmy graf o n wierzchołkach, gdzie krawędź miedzy i i j oznacza istnienie meczu między graczem i i j, który wygrał gracz j. Na tak utworzonym grafie, DFS puszczony z wierzchołka i odwiedza wszystkich wierzchołki reprezentujące silniejszych szachistów (i szachistę i). Jeżeli odwrócimy ten graf to analogiczny DFS odwiedzi słabszych szachistów. Zgodnie z obserwacją szachista i ma określoną rangę gdy oba dfs odwiedzą sumarycznie $n-1$ wierzchołków (nie licząc wierzchołka i). Dla każdego prefiksu krawędzi i dla każdego wierzchołka potrafimy w ten sposób liniowo stwierdzić czy rangę da się określić. Umiemy policzyć wynik w złożoności $O(m*n*(n+m))$.

Podzadanie 3

Dla każdego wierzchołka chcąc policzyć wynik, możemy się wyszukiwać binarnie po wyniku. Daje to złożoność $O(log(m)*n*(n+m))$.

Podzadanie 4

Utwórzmy graf jak w podzadaniach 2 i 3 zawierający krawędzie reprezentujące wszystkie mecze, wykonajmy na nim sortowanie topologiczne. Otrzymamy pewną kolejność $a_1 \ldots a_n$. Z definicji, jeżeli istniał mecz między szachistami i i j (wygrany przez j) to j pojawia się później w naszym ciągu niż i. Przenumerujmy wszystkie wierzchołki, tak że i-temu wierzchołkami przypisujemy jego pozycję w ciągu. Teraz możemy założyć, że dla każdego meczu i, j (wygrany przez j) $i < j$.
Dla wierzchołka x oznaczmy $a[x]$ jako najmniejszy numer szachisty, który wygrał mecz z szachistą x. Analogicznie $b[x]$ jako największy numer szachisty, który przegrał z szachistą x.

LEMAT: x ma określoną rangę wtedy i tylko wtedy, gdy dla każdego $y < x$ zachodzi $a[y] \leq x$ oraz dla każdego $y > x$ zachodzi $b[y] \geq x$ (dowód niżej).

Niech $c[x]$ to liczba takich szachistów, którzy nie spełniają warunku z lematu, zatem liczba y, że $y < x$ i $a[y] > x$ lub $y > x$ i $b[y] < x$. Szachista ma określoną rangę, wtedy kiedy $c[x]$ staje się zerem. Załóżmy że szachistów numerujemy od 1 do n. Zainicjujmy tablice a jako $n+1$, a b jako 1, zgodnie z definicją c powinno być wypełnione $n-1$. Rozpatrujmy mecze po kolei, niech aktualny mecz między i i j, $i < j$. W tablicy a aktualny mecz wpływa potencjalnie na wartość i, jeżeli j jest nowym najgorszym szachistą, który wygrał z i. $a[i]$ staje się j, zatem przestaje być większe od liczb $j \ldots (a[i]-1)$. Zatem na takim przedziale należy zmniejszyć tablicę c o 1 i zaktualizować $a[i]$. W tablicy b sprawa wygląda podobnie, mecz potencjalnie wpływa na wartość $b[j]$, wtedy zwiększylibyśmy wartość w $b[j]$ na i. Należałoby zmniejszyć wartość w c o 1 na przedziale $(b[j]+1) \ldots i$ i zaktualizować $b[j]$. W ten sposób zaktualizowaliśmy tablicę c, każdy indeks na którym właśnie

pojawiło się 0, dotyczy szachisty którego ranga stała się właśnie znana.

Potrzebujemy struktury danych pozwalającej na odejmowanie 1 na przedziale, oraz zwracanie listy wszystkich indeksów, gdzie wartością stało się 0. Możemy rozwiązać ten problem drzewem przedział-przedział umożliwiającym dodanie wartości na przedziale i odczytaniem minimum na przedziale. W naszym algorytmie odejmowanie 1 przeprowadzamy standardowo. Gdy chcemy uzyskać listę indeksów które stały się zerami: Póki minimum w korzeniu drzewa wynosi 0, schodzimy do odpowiedniego syna (tego w którym poddrzewie minimum wynosi 0) i w ten sposób znajdujemy liścia - szukany numer szachisty - zapisujemy go. Nie chcemy ponownie wracać do tego wierzchołka, zatem ustawiamy wartość w nim na INT_MAX. Pamiętamy o zaktualizowaniu minimum w całym drzewie i powtarzamy algorytm. złożoność $O((m+n)log(n))$.

Dowód lematu.
Ustalmy x, pokażę, że jeśli dla każdego $y < x$ zachodzi $a[y] \leq x$ to dla każdego $y < x$ na pewno y słabsze od x. Analogicznie można udowodnić, że z warunku na $b[x]$ wynika, że wszyscy szachiści o większym numerze są silniejsi, zatem ranga x jest znana.
Indukcyjnie dla $y = x - 1$ musi zachodzić $a[y] = x$, bo $a[y] > y$, zatem y przegrał bezpośrednio z x - jest od niego słabszy.
Krok indukcyjny: zakładam że każdy z szachistów o numerze $i \ldots (x-1)$ jest słabszy od x. Zauważmy że $a[i-1]$ do tego ciągu lub jest równe x, bo $a[i-1] > i$, ale też $a[i-1] \leq x$. Zatem $i-1$ jest słabszy od szachisty słabszego od x lub jest słabszy bezpośrednio od x.
Pokazałem że warunek z lematu jest dostateczny, udowodnię że jest konieczny. Załóżmy że mam $y < x$ oraz $a[y] > x$ (dowód z b jest analogiczny). Zauważmy że między $y+1$ i x (włącznie z końcami), żaden szachista nie wygrał bezpośrednio z szachistą y. Okazuje się, że możemy w aktualnej kolejności wierzchołków wstawić y bezpośrednio po x i wciąż uzyskamy poprawne numerowanie. Czyli x może mieć zarówno rangę x i $x-1$ - nie jest ona znana.