Descrierea soluțiilor

1 Problema Caii

Distribuția punctelor în acestă problemă este construită astfel încât să existe multe subprobleme independente relativ ușoare însă care nu pot fi implementate cu ușurință împreună datorită diferențelor masive dintre condițiile acestora.

1.1 Pentru punctaje parțiale

Pentru fiecare dintre subprobleme oferim o scurtă descriere a soluției.

Majoritatea soluțiilor se folosesc de următoarele 2 componente:

Găsirea celui de al doilea nod pe un drum

Dându-se 2 noduri A și B să se găsească al doilea nod pe drumul de la A la B. Această operație se poate implementa în $\log N$ prin indexarea fiecărui nod în ordinea parcurgerii în pre-ordine. După această reindexare toate direcțiile mai puțin una corespund unui interval de indecși și putem căuta binar.

Subarborele nodurilor active

Definiție 1 (Nod activ)**.** *Numim un nod* activ *drept un nod în care pot exista cai după toate updateu-rile până la momentul curent dacă presupunem că inițial fiecare nod avea un cal .*

Observație 1. *Setul de noduri active este un subarbore conex.*

Soluțiile subproblemelor

Mai jos se găsesc soluțiile subproblemelor:

1. **Subproblema 1:** În cazul a doar 6 cai, putem reține poziția fiecărui cal și manipula mișcarea fiecărui cal independent în cazul unui update.
2. **Subproblema 2:** În acest caz putem avea o soluție în $O(NQ)$ care simulează complet mișcarea fiecărui cal în arbore.
3. **Subproblema 3:** În acest caz, putem folosi o abordare similară ca în subproblema 1 și itera prin fiecare poziție care conține un cal.
4. **Subproblema 4:** Întrebarea se reduce la a verifica dacă un nod este *activ* sau nu. Este suficient să menținem subarborele nodurilor active. Putem observa că în urma unei operații CJ, toate frunzele curente mai puțin maxim una vor dispărea, iar celelalte element vor rămâne în continuare. Este suficient să avem la dispoziție o structură de date ce ne spune frunzele curente și poate elimina frunze. Acest lucru se poate implementa cu ușurință prin menținerea fiecărui grad și a listelor de adiacență.
5. **Subproblema 5:** În cazul în care arborele este un lanț atunci putem rezolva problema utilizând o structură de date ce suportă următoarele operații:
   (a) Elimină element la o poziție arbitrară.
   (b) Modifică element la o poziție arbitrară.
   (c) Returnează ce element se găsește la o poziție arbitrară.

   Există multe structuri care pot implementa aceste operații, cea mai ușor de utilizat fiind cea de Treap.
6. **Subproblema 6:** În acest caz observăm putem descompune arborele în lanțuri maximale (nu putem reduce trivial numărul de lanțuri prin combinarea a două lanțuri adiacente). Fiecare lanț trebuie să implementeze următoarele operații:
   (a) Mută toate elementele cu o poziție la stânga.
   (b) Elimină element la o poziție arbitrară.

(c) Modifică element la o poziție arbitrară.

(d) Returnează ce element se găsește la o poziție arbitrară.

Putem apoi itera prin lanțurile ce încă conțin noduri active și să le facem operațiile aferente pe lanțuri. Motivul pentru care această soluție se amortizează este deoarece fiecare lanț procesat garantează reducerea numărului de noduri active cu 1. În soluția completă acestă idee o să fie generalizată mai departe.

1.2 Pentru punctaj complet

Putem împărți arborele în lanțuri folosind tehnica heavy-path. Apoi o să împărțim lanțurile de lungime mai mare ca $L = \sqrt{N}$ în lanțuri de lungime maxim $L = \sqrt{N}$.

În cadrul fiecărui lanț exista un număr de noduri în care pot exista caii. Numim un lanț *activ* drept un lanț care să conțină cel puțin un nod *activ*. Pentru fiecare lanț activ numim *sublanțul activ* sublanțul de noduri pe care există doar noduri active.

Definim funcția de potențial: $\phi = \phi_{noduri} + \phi_{lanțuri}$ unde ϕ_{noduri} este numărul de noduri active, iar $\phi_{lanțuri}$ este numărul de lanțuri active.

Pentru fiecare operație CJ o să actualizăm informația reținută în fiecare lanț activ. Fie ϕ^{-}_{noduri} schimbarea în potențial dată de operația CJ.

Există multiple tipuri de interacțiuni posibile între o operație CJ și un lanț:

1. Nodul de operație CJ este parte dintr-un lanț A. În acest caz putem modifica informația pe A în $O(L)$. Acest caz se poate întampla o singură dată.
2. Cel mai apropiat nod din sublanțul activ de nodul de update este unul din capetele acestuia:
   (a) Următorul nod după capătul lanțului nu face parte dintr-un alt lanț activ: Acest lucru se întamplă o singură dată și nu afectează funcția de potențial. Putem modifica informația lui A în $O(1)$.

(b) Următorul nod după capătul sublanțului activ face parte dintr-un alt lanț activ B, iar acest următor nod se găsește în capătul sublanțului activ al lui B: Acest lucru se poate întâmpla de maxim $\frac{N}{L} + \phi^{-}_{noduri} \cdot logN$. Putem modifica informația lui A și B în $O(1)$.

(c) Următorul nod după capătul sublanțului activ face parte dintr-un alt lanț activ B, iar acest următor nod se găsește strict în interiorul sublanțului activ al lui B: Acest lucru se întamplă de maxim ϕ^{-}_{noduri} ori și putem modifica informația lui A și B în $O(1)$.

3. Cel mai apropiat nod din sublanțul activ de nodul de operație CJ este strict în interiorul sublanțului activ: Acest caz se întâmplă de maxim ϕ^{-}_{noduri} ori și evaluăm în $O(L)$ schimbarea pe lanț.

Un lanț trebuie să suporte următoarele operații în mod relativ eficient:

1. Shiftează toate valorile pe lanț. Complexitate $O(1)$.
2. Elimină o valoare din mijlocul lanțului. Complexitate $O(L)$.
3. Elimină o valoarea de la capătul sublanțului activ. Complexitate $O(1)$.
4. Extrage valoarea de la o anumită poziție din lanț . Complexitate $O(1)$.
5. Modifică valoarea de la o anumită poziție din lanț. Complexitate $O(1)$.

La final obținem o soluție de complexitate amortizată $O(N\sqrt{N})$.