Kdist – descrierea solutiei

stud. Andrei Pârvu – Universitatea "Politehnica" Bucuresti
stud. Andrei Ciocan – Universitatea "Politehnica" Bucuresti

Solutie O(N^2) - 30-40 de puncte

Grupam nodurile dupa culori, si pentru fiecare pereche de noduri de aceeasi culoare calculam distanta dintre ele, folosind un algoritm de determinare al celui mai de jos stramos comun.

Solutie O(N*sqrtN) – 60-70 de puncte – Mugurel Andreica

Se impart culorile in 2 tipuri: cele cu numar mic de noduri (<= sqrt(N)) si cele cu numar mare de noduri (>sqrt(N) noduri). Pentru culorile "mici" facem brute force pt fiecare (luam oricare 2 noduri din acea culoare, calculam LCA-ul in O(1) si pe baza lui calculam distanta dintre ele ; pentru a calcula LCA-ul in O(1) putem face o preprocesare in O(N*logN) initial). Pentru fiecare culoare "mare" rezolvam problema in timp O(N) (dinamica pe arbore).

Complexitatea totala este:

- pentru culorile mici: suma din $x(i)^2$ (x(i) = numarul de noduri cu culoarea i) ; cum toate x(i)-urile sunt <= sqrt(N), valoarea maxima posibila a acestei sume se obtine cand avem N/sqrt(N) culori mici, fiecare avand sqrt(N) noduri => O((N/sqrt(N)) * $sqrt(N)^2$)=O(N*sqrt(N))
- pentru culorile mari: sunt maxim N/sqrt(N) culori mari, fiecare rezolvata in timp O(N) => O(N * N/sqrt(N)))=O(N*sqrt(N))

Solutie - 60-70 de puncte – Adrian Panaete

Se realizează o descompunere a arborelui in drumuri de lungime cât mai mare (longest path decomposition) şi se realizează o comasare a mai multor noduri de aceeaşi culoare în unul singur . Descompunerea se realizează formând initial un cel mai lung drum care pleacă din rădăcină spre o frunză apoi plecând de la succesorii nodurilor de pe drum se alege cel mai lung drum care pleaca de la un astfel de succesor spre o frunză şi asa mai departe în aşa fel încat în cele din urma fiecare drum cu excepţia celui iniţial se va lega (se va subordona) la un alt drum mai lung. Pentru fiecare drum vom reţine drumul la care se subordonează şi nivelul la care se realizeaza conexiunea.

Pentru fiecare drum se memorează într-un multiset nodurile sortate dupa criteriul culoare – nivel – greutate. Prin greutatea unei culori într-un nod intelegem câte noduri având acea culoare sunt comasate în nodul corespunzător. Pe fiecare drum vom procesa nodurile în ordinea culorii şi în cadrul aceleiaşi culori în ordinea descrescătoare a distanţei faţă rădăcină. Există două situaţii când procesăm o culoare

1. Avem pe un drum un nod de culoarea procesata şi mai avem unul pe acelaşi drum. Atunci "mutăm" nodul de jos şi îl comasăm cu cel situat deasupra şi care conţine aceeasi culoare.

2. Pe drum a rămas un singur nod pe culoarea procesată. Atunci "mutăm" nodul în drumul la care se conectează drumul curent la un alt drum.

În ambele cazuri la soluţia culorii respective se aduna o valoare egala cu produsul între greutatea culorii în nodul de plecare cu diferenţa intre greutatea totală a culorii respective cu acest număr şi cu numarul de nivele pe care s-a realizat mutarea.
Excepţie face cazul în care un nod are greutatea pe culoarea procesată este egală cu greutatea totală a culorii situaţie în care de fapt am obţinut rezultatul final pentru culoarea respectivă.

Solutie O(NlogN) – 100 de puncte

Solutia pleaca de la calculul distantei intre doua noduri: se aduna adancimile celor doua noduri si se scade 2 * adancimea celui mai de jos stramos comun. La fel se poate proceda si pentru T noduri (toate de aceeasi culoare in cazul problemei noastre): se aduna adancimile tuturor nodurilor si se scad stramosii comuni pentru oricare doua. Totusi, se observa ca foarte multi stramosi comuni coincid pentru perechile formate din cele T noduri, asa ca incercam sa determinam fiecare stramos de cate ori trebuie scazut.
Pentru a face acest lucru, procesam nodurile in ordinea unei parcurgeri DFS, calculam stramosii comuni pentru oricare doua noduri succesive, si sortam acesti stramosi comuni descrescator dupa adancimea lor (dorim sa procesam stramosii de jos in sus). Vom mentine pentru fiecare stramos comun cele doua noduri ce l-au generat. Pe masura ce procesam stramosii, trebuie sa unim multimile nodurilor ce au generat un stramos dat (este clar ca daca unul din nodurile ce a generat stramosul este implicat in calculul altui stramos si scaderea adancimii acestuia, atunci si celalat nod va fi implicat, iar adancimea stramosului va trebui scazuta si pentru el). Astfel, cand intalnim un stramos x ce este generat de y si z, trebuie sa scadem 2 * adancime(x) * cardinal_multime(y) * cardinal_multime(z) si dupa aceea sa unim multimea lui y cu multimea lui z.
Bineinteles, operatiile pe multimi trebuie realizate intr-un mod eficient, folosind structuri de multimi disjuncte, intr-o complexitate de O(Nlog*N).
Complexitatea finala a algoritmului iese din calulul stramosilor comuni, pentru care este nevoie de o preprocesare de NlogN.

Se poate observa ca se stiu dinainte perechile de noduri pentru care se doreste calcularea stramosului comun, asa ca se poate folosi algoritmul lui Tarjan, de o complexitate O(Nlog*N), care reduce rezolvarea problemei la o complexitate totala de O(Nlog*N). Totusi, aceasta optimizare nu este necesara pentru obtinerea punctajului maxim.