Problema 2- Drumuri

Autor: stud. Alexandru Cazacu, Universitatea București

Soluție 50 puncte - complexitate O (N * (N + M))

Pentru 50% din teste, putem să calculăm pentru fiecare nod în parte, nodurile în care se poate ajunge plecând din el. Deasemnea, făcând încă o parcurgere pe graful transpus, putem vedea nodurile din care se ajunge in nodul curent.

Solutia 1(100 de puncte) Alex Cazacu - complexitate O(N)

Se observă că dacă un nod este în mulțimea soluțiilor, atunci toate nodurile din componenta sa tare conexă respectă această proprietate. Putem lucra pe graful aciclic al componentelor tare conexe. În continuare, începem să parcurgem nodurile ca la sortarea topologică, pornind din cele exterioare (care au gradul interior 0). Dacă la un moment dat avem un singur nod exterior, atunci acest nod este soluție. În momentul în care avem mai multe astfel de noduri, este evident că nu se poate ajunge de la unul la altul. Următorul nod din soluție, este cel în care se unesc aceste "ramuri" care pleaca din nodurile exterioare. Este necesară o tăiere simultană a lor. Procedam astfel : calculăm pentru fiecare nod dișanta celui mai lung drum care pleacă din el. La un pas eliminăm nodurile cu gradul de intrare 0, care au distanța maximă. Când ajungem sa avem la un pas, un singur nod exterior, il adăugam la soluție.

Solutia 2(100 de puncte) Adrian Panaete - complexitate O(N)

Lucrăm din nou pe graful aciclic al componentelor tare conexe. Fie T o sortare topologică a grafului. Pentru ca un nod X situat pe pozitia P în sortarea topologică să fie popular este nevoie ca X sa fie accesibil din fiecare nod situat pe o pozitie < P în sortare. Analog, este nevoie ca fiecare nod situat pe o pozitie > P în sortare sa fie accesibil din X.

Astfel, ne vine ideea sa ținem pentru fiecare nod X valoarea nr_pred[X] = numarul de predecesori ai lui X în sortare cu proprietatea că X este accesibil din predecesorul respectiv. Această valoare nu poate fi calculată in timp liniar pentru fiecare nod, deoarece un algoritm simplu de tip programare dinamica va număra anumiți predecesori de mai multe ori. Putem însă realiza un algoritm simplu care calculeaza valorile nr_pred[] in mod corect doar pentru nodurile candidate la solutie. Mai exact, sa spunem ca avem deja calculata valoarea nr_pred[X]. Vom aduna aceasta valoare la primul vecin al lui X care îl urmeaza în

sortarea topologică. Astfel știm sigur ca valoarea nr_pred[] va fi corectă pentru fiecare nod care este accesibil din toate nodurile precedente. Argumentăm prin faptul că singurul caz în care un anumit nod poate "rata" un update de la un predecesor este cel în care nodul nu este accesibil din predecesorul respectiv. Astfel, el nu poate face parte din soluție.

Pentru a completa soluțtia, vom aplica același algoritm și pe graful transpus pentru a calcula numărul de succesori pentru fiecare nod. Nodurile care au atat numarul necesar de predecesori cat si numarul necesar de succesori vor fi nodurile din solutie.