**Descrierea soluţiei ­– raspandaci**

prof. Emanuela Cerchez

Colegiul Naţional "Emil Racoviţă" Iaşi

Modelăm problema ca un graf orientat: vârfurile grafului sunt cei n elevi; dacă elevul x transmite mesaje elevului y vom avea arc de la x la y.

Fie x un vârf din graf. Să notăm acc(x)= mulţimea vârfurilor accesibile din x=mulţimea vârfurilor y pentru care există drum de la x la y.

Trebuie să determinăm numărul minim de vârfuri din graf cu proprietatea că reuniunea mulţimilor vârfurilor accesibile din aceste vârfuri este egală cu mulţimea vârfurilor grafului (practic aceste vârfuri „acoperă” graful).

Ţinând cont de restricţiile din enunţul problemei, vom reprezenta graful prin liste de adiacenţă alocate dinamic (liste simplu înlănţuite sau vectori din STL).

*Pasul I. Descompunem graful în componente tare conexe.*

Din nou, ţinând cont de restricţiile din enunţul problemei, este necesar un algoritm eficient, ca urmare vom utiliza algoritmul *Kosaraju-Sharir*.

Pe scurt, algoritmul este:

1. Se parcurge graful în adâncime şi se numerotează vârfurile grafului în postordine (vârful x este numerotat după ce toţi succesorii săi au fost numerotaţi); în vectorul postordine se memorează ordinea vârfurilor.
2. Se determină graful transpus GT
3. Se parcurge graful transpus în adâncime, considerând vârfurile în ordinea inversă a vizitării lor în parcurgerea DFS a grafului iniţial.
4. Fiecare subgraf obţinut în parcurgerea DFS a grafului transpus reprezintă o componentă tare-conexă a grafului iniţial.

Complexitatea algoritmului Kosaraju-Sharir de descompunere în componente tare-conexe O(n+m).

În cadrul unei componente tare-conexe există drum de la x la y şi drum de la y la x pentru oricare două vârfuri x, y.

*Pasul II. Construim graful condensat (graful componentelor tare-conexe).*

În graful condensat există câte un vârf pentru fiecare componentă tare-conexă din graful iniţial.

Dacă există arc de la un vârf din componenta tare-conexă x la un vârf din componenta tare-conexă y, va exista un arc de la x la y în graful condensat.

Evident, graful condensat nu mai conţine circuite.

Construim un vector d cu nrc elemente (nrc este numărul de componente tare-conexe), unde d[x]=numărul de arce care au ca extremitate finală un vârf din componenta tare-conexă x.

*Pasul III. Determinare rezultat*

Dacă d[x] este nenul, evident niciun vârf din această componenă tare-conexă nu va fi răspandac (aceste vârfuri vor primi mesaje de la alte vârfuri din alte componente tare-conexe.

Răspândacii vor fi obligatoriu vârfuri din componente tare-conexe x pentru care d[x]=0 (deoarece acestea nu primesc mesaje de la niciun alt vârf).

Deci soluţia este numărul de elemente nule din vectorul d, deoarece este suficient să alegem un singur vârf din fiecare componentă tare-conexă cu această proprietate.

*Observaţie*

Punctaje parţiale se pot obţine pe algoritmi mai ineficienţi (de exemplu, descompunere în componente tare-conexe bazată pe matricea drumurilor).

**Descrierea soluţiei ­– zaruri**

lect. dr. Paul Diac

Facultatea de Informatică, Universitatea „*Alexandru Ioan Cuza*” Iaşi

Calculăm numărul de aruncări a n zaruri cu suma s, pe care îl reținem într-o matrice pos[n][s], ce se determină în funcție de numărul de aruncări a n-1 zaruri cu sume mai mici. Luând în considerare toate cele 6 valori posibilie pentru zarul n obținem formula de mai jos, pe care o calculăm folosind programare dinamică:

pos[n][s] =

Pentru a calcula eficient numărul de aruncări a n zaruri cu suma între st și dr, putem precalcula în matricea poss[n][s] suma elementelor din matricea pos[][] de pe linia n, și coloane până la s. Matricea poss[][] este o matrice de sume parțiale, calculată folosind o altă formulă de programare dinamică, mai simplă:

poss[n][s]=pos[n][1]+pos[n][2] + … + pos[n][s]=poss[n][s-1]+pos[n][s]

Răspunsul la o întrebare de forma n st dr este:

pos[n][st] + pos[n][st+1] + … + pos[n][dr] = poss[n][dr] - poss[n][st-1]

Putem calcula elementele matricii poss[][] o singură dată în ordinea crescătoare a liniilor și să răspundem la întrebări, dacă sortăm întrebările după valorile n. Alternativ, putem reține pentru fiecare valoare posibilă a lui n întrebările ce au n egal cu valoarea respectivă. În ambele variante trebuie să avem grijă ca la final să afișăm răspunsurile întrebărilor în ordinea lor din fișierul de intrare.

Pentru a nu depăși limita de memorie, ne folosim de faptul că o linie depinde doar de linia anterioară, astfel putem reduce matricile pos[][] și poss[][] la doar două linii, reducând complexitatea spaţiului de memorie de la pătratică la liniară.

Soluțiile ce generează toate posibilitățile prin *backtracking* ar trebui să se încadreze în timp pentru testele cu n,q < 10.