

## Descrierea soluției – raspandaci

prof. Emanuela Cerchez  
Colegiul Național "Emil Racoviță" Iași

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  $\text{acc}(x)$  = mulțimea vârfurilor accesibile din  $x$  = mulțimea vârfurilor  $y$  pentru care există drum de la  $x$  la  $y$ .

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

Ținând cont de restricțiile din enunțul problemei, vom reprezenta graful prin liste de adiacență alocate dinamic (liste simplu înlănțuite sau vectori din STL).

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

Din nou, ținând cont de restricțiile din enunțul problemei, este necesar un algoritm eficient, ca urmare vom utiliza algoritmul Kosaraju-Sharir.

Pe scurt, algoritmul este:

1. Se parcurge graful în adâncime și se numerotează vârfurile grafului în postordine (vârful  $x$  este numerotat după ce toți succesorii săi au fost numerotați); în vectorul *postordine* se memorează ordinea vârfurilor.
2. Se determină graful transpus  $G^T$
3. Se parcurge graful transpus în adâncime, considerând vârfurile în ordinea inversă a vizitării lor în parcurgerea DFS a grafului inițial.
4. Fiecare subgraf obținut în parcurgerea DFS a grafului transpus reprezintă o componentă tare-conexă a grafului inițial.

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  $y$  și 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 inițial.

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 conține circuite.

Construim un vector  $d$  cu  $n_{rc}$  elemente ( $n_{rc}$  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 soluția este numărul de elemente nule din vectorul  $d$ , deoarece este suficient să alegem un singur vârf din fiecare componentă tare-conexă cu această proprietate.

*Observație*

Punctaje parțiale se pot obține pe algoritmi mai ineficienți (de exemplu, descompunere în componente tare-conexe bazată pe matricea drumurilor).

## Descrierea soluției – zaruri

lect. dr. Paul Diac

Facultatea de Informatică, Universitatea „Alexandru Ioan Cuza” Iași

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 posibile pentru zarul  $n$  obținem formula de mai jos, pe care o calculăm folosind programare dinamică:

$$pos[n][s] = \sum_{z=1}^{\min(6,s)} pos[n-1][s-z]$$

Pentru a calcula eficient numărul de aruncări a  $n$  zaruri cu suma între  $st$  și  $dr$ , putem precalcuła 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:

$$poss[n][s] = pos[n][1] + pos[n][2] + \dots + pos[n][s] = poss[n][s-1] + pos[n][s]$$

Răspunsul la o întrebare de forma  $n \ st \ dr$  este:

$$pos[n][st] + pos[n][st+1] + \dots + pos[n][dr] = poss[n][dr] - poss[n][st-1]$$

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 spațiului de memorie de la pătratică la liniară.

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