Construcția unui graf cu secvența de grade dată

Necesitatea *construcției unei model de tip rețea în care pentru fiecare nod se impun restricții asupra numărului de legături directe care trebuie stabilite din fiecare nod apare în domenii precum: c****himie*** *– studiul structurii posibile a unor compuși cu formula chimică dată (formula chimică = secvența de grade),* ***proiectare de rețele****,* ***biologie*** *– rețele metabolice, de interacțiuni între gene/proteine,* ***studii epidemiologice*** *– în care prin chestionare anonime persoanele declară numărul de persoane cu care au interacționat, în studii bazate pe simulări de rețele. În anumite cazuri este nevoie de a simula un model aleatoriu de rețea cu gradele nodurilor date sau de mai multe modele de rețea, sau de modele cu alte proprietăți precum: rețele conexe, rețele aciclice.*

Arbori. Codificarea arborilor

Ideea de a găsi o reprezentare a unui arbore cu mulțimea vârfurilor {1,2,…,n} sub formă de vector de lungime cât mai mică își are ***originea*** *în probleme de* ***numărare****. Pentru a număra arbori cu anumite proprietăți legate spre exemplu de grade este util să asociem fiecărui arbore un vector (un cod) cu elemente din mulțimea {1,2,…,n} , astfel încât această asociere să fie bijectivă, problema reducându-se astfel la numărarea de vectori cu anumite proprietăți, care este mai simplă. Astfel, au fost propuse modalități de codificări ale arborilor sub formă de vectori de lungime n – 2 (! nu n cât necesită vectorul de tați) cu elemente în mulțimea {1,2,…,n}. Una dintre cele mai vechi si mai cunoscute codificări, introdusă pentru a demonstra Teorema lui Cayley, este codificarea Prüfer.*

Codificările arborilor (sub formă de vectori) sunt utile însă și în informatică, spre exemplu în ***programarea evolutivă*** *în codificarea cromozomilor (care în general sunt reprezentați sub formă vectorială), în generarea de arbori aleatori, în transmiterea de informații reprezentate prin structuri arborescente.*

Probleme

1. La încheierea anului universitar studenții dintr-o grupă completează un chestionar în care declară numărul de colegi cu care au colaborat la proiecte de-a lungul anului. Pentru a verifica dacă datele colectate sunt corecte, dar și posibile modele ale rețelei de colaborare profesională stabilită de studenți de-a lungul anului universitar, se dorește construirea unui model corespunzător datelor corectate. Se citesc din fișierul `date.in` următoarele informații: numărul de studenți n urmat de n numere d_1, d_2, …, d_n, unde d_i reprezintă numărul de colegi cu care a declarat al i-lea student că a lucrat la proiecte.

a) Construiți, dacă se poate, un model al relațiilor de colaborare stabilite între studenți care să corespundă rezultatelor din chestionar. Se vor afișa perechile de studenți care au colaborat.

b) Construiți, dacă se poate, un model arborescent al relațiilor de colaborare stabilite între studenți care să corespundă rezultatelor din chestionar.

c) Construiți, dacă se poate, un model conex corespunzător datelor.

Modelare

Se citesc din fișierul `date.in` următoarele informații: un număr natural n urmat de n numere naturale d_1, d_2, …, d_n. Să se verifice dacă există un graf G cu secvența gradelor s(G) = { d_1, d_2, ... , d_n } și, în caz afirmativ, să se afișeze **muchiile** grafului G. Se vor considera pe rând cazurile în care G este:

a) graf neorientat – **$O(n^2)$** **(2,5p)**

date.in	**date.out (nu este unica solutia)**
6 2 3 2 3 2 2	1 2 2 3 2 4 4 5 4 6 1 3 5 6

b) arbore – **O(n)** **(1,5p)**

date.in	**date.out (nu este unica solutia)**
6 1 3 1 2 2 1	1 2 2 3 2 4 4 5 5 6

c) graf neorientat conex (suplimentar)

http://campion.edu.ro/arhiva/index.php?page=problem&action=view&id=885

2. **Codificare Prüfer**. Gigel este administrator al unei rețele de calculatoare. Între anumite perechi de noduri ale rețelei comunicarea se poate face direct, prin conexiuni bidirecționate. Între celelalte noduri comunicarea se poate face prin noduri intermediare. Gigel vrea să facă o revizie a conexiunilor din rețea, dar nu are timp să le revizuiască pe toate. De aceea, el asociază numere de la 1 la n nodurilor din rețea și îi trimite prietenului său Ionel un fișier `retea.in` care conține numărul de noduri ale rețelei și perechi de numere reprezentând conexiunile directe care există între nodurile rețelei și îl roagă pe acesta să îi găsească un număr minim de conexiuni care trebuie revizuite astfel încât între oricare două noduri din rețea comunicarea să se poată face prin legături revizuite.

   a) Ionel, care știe algoritmi de grafuri (dar nu are timp să îi implementeze), înțelege că Gigel are nevoie de fapt de muchiile unui arbore parțial al rețelei și vă roagă să scrieți un program care determină un arbore parțial al rețelei trimise de Gigel și scrie în fișierul `rezultat.out` doar n – 2 numere din care prietenul său să își poată reconstrui muchiile arborelui. Aceste n – 2 numere vor reprezenta codul Prüfer asociat arborelui determinat (**O(n+m)** pentru determinarea arborelui parțial, **O(n)** codificarea Prüfer). (**1+2,5p**)

   b) **Scrieți un alt program (diferit de cel de la a) care să îl ajute pe Gigel să reconstituie conexiunile din rețea care trebuie revizuite pe baza fișierului `rezultat.out` primit de la prietenul său; conexiunile vor fi scrise în fișierul `conexiuni.out` (decodificarea Prüfer)** - O(n) (**2,5p**)

retea.in	**rezultat.out (nu este unic)**
8 1 3 1 8 3 6 5 8 5 7 4 7 4 8 2 7 2 5	7 7 3 1 8 4
	conexiuni.out (nu neaparat in aceasta ordine)
	2 7 5 7 6 3 3 1 1 8 8 4 4 7

3. **Generarea aleatorie a unui arbore cu secvența gradelor dată**. Scrieți un program care, dată o secvență { d_1, d_2, ... , d_n }, genereză aleatoriu codul Prüfer al unei arbore cu această secvență de grade și să afișeze muchiile arborelui (suplimentar)