



**HAL**  
open science

# The Tree of Biconnected Components and Critical Nodes

Mugurel Ionut Andreica

► **To cite this version:**

Mugurel Ionut Andreica. The Tree of Biconnected Components and Critical Nodes. GInfo (Informatics Gazette), vol. 16 (5), May 2006. (ISSN: 1222-7129), 2006, 16 (5), pp.11-17. hal-00338301

**HAL Id: hal-00338301**

**<https://hal.science/hal-00338301>**

Submitted on 12 Nov 2008

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**Arborele  
componentelor biconexe  
si al  
nodurilor critice**

**ANDREICA MUGUREL IONUT**

*student Universitatea Politehnica Bucuresti*

**Mai 2006**

# Cuprins

- 1. Termeni “de baza”**
- 2. Determinarea nodurilor critice si a muchiiilor critice ale unui graf neorientat & conex**
- 3. Determinarea componentelor biconexe ale unui graf neorientat & conex**
- 4. Arborele componentelor biconexe si al nodurilor critice**
- 5. Descompunerea arborescenta a unui graf**
- 6. Cateva probleme rezolvabile folosind arborele componentelor biconexe si al nodurilor critice**
- 7. Bibliografie & referinte**

## 1. Termeni “de baza”

Un **graf**  $G$  este o pereche  $(V, E)$ , unde  $V$  este multimea nodurilor grafului, iar  $E$  este multimea muchiilor grafului. Nodurile grafului se considera etichetate cu etichete distincte (de cele mai multe ori vom folosi etichetele  $1, 2, \dots, N$ , unde  $N$  este  $|V|$ ). O muchie  $(u, v)$  este o pereche de noduri distincte ale grafului,  $u$  si  $v$ . Un **graf orientat** este un graf in care muchiile sunt perechi ordonate ( $(u, v)$  este diferit de  $(v, u)$ ), iar un **graf neorientat** este un graf in care muchiile se considera a fi perechi neordonate ( $(u, v)$  este identic cu  $(v, u)$ ). In continuare vom considera numai grafuri neorientate.

Vom folosi formularea “nodul  $x$  *‘apartine’* grafului  $G$ ”, daca  $G=(V,E)$  si  $x$  apartine lui  $V$ . In mod similar vom folosi formularea “muchia  $(u,v)$  *‘apartine’* grafului  $G$ ”. Formularea “o muchie *‘este adiacenta’* cu un nod” implica faptul ca nodul respectiv este unul din cele doua noduri din perechea ce reprezinta muchia.

Un **subgraf**  $SG$  al unui graf  $(V,E)$  este un graf  $(V', E')$ , unde  $V'$  este o submultime de noduri inclusa in  $V$ , iar  $E'$  contine o parte dintre acele muchii din  $E$  in care ambele noduri din pereche sunt incluse in  $V'$ . Un **subgraf indus**  $SG$  al unui graf  $(V,E)$  este un graf  $(V', E')$ , unde  $V'$  este o submultime de noduri inclusa in  $V$ , iar  $E'$  contine **toate** muchiile din  $E$  in care ambele noduri din pereche sunt incluse in  $V'$

Un **drum**  $D$  intr-un graf este o succesiune de noduri ale grafului  $(n[1], n[2], \dots, n[k])$ , avand proprietatea ca exista muchiile  $(n[i], n[i+1])$ . Se considera si cazul cand  $k = 1$ . Un drum de la  $x$  la  $y$  este un drum in care  $n[1]=x$  si  $n[k]=y$ .

O **componenta conexa** a unui graf  $G$  este un subgraf indus maximal al lui  $G$ , cu proprietatea ca intre oricare doua noduri ce apartin componentei exista cel putin un drum. O componenta conexa este maximala daca orice alt nod al grafului am adauga multimii nodurilor componentei, proprietatea mentionata nu se mai respecta.

Orice graf poate fi partitionat in mod unic in componente conexe (orice nod si orice muchie fac parte din exact o componenta conexa). Un graf se numeste conex daca el este partitionat intr-o singura componenta conexa. In continuare vom discuta numai despre grafuri conexe (si, dupa cum am mentionat mai sus, neorientate).

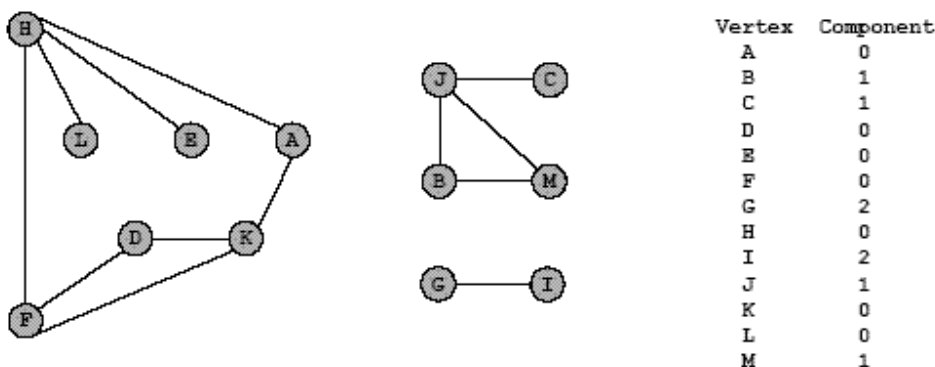


Figura 1. Partitionarea unui graf in componente conexe

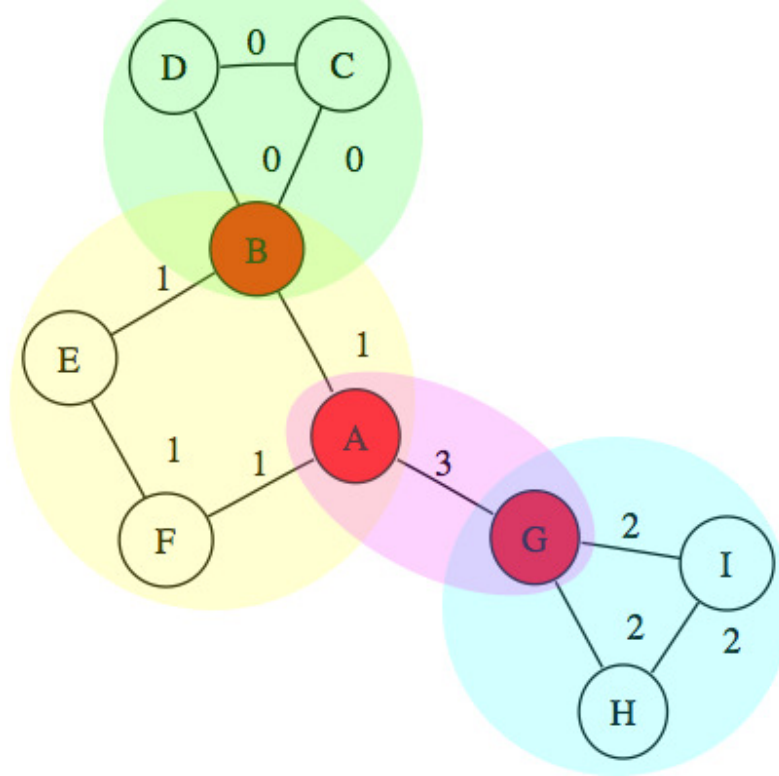
Un **nod critic** al unui graf  $G$  (numit si **punct de articulatie**) este un nod al grafului care, daca este scos din graf, impreuna cu toate muchiile adiacente, graful obtinut nu mai ramane conex. O **muchie critica** a unui graf  $G$  este o muchie a grafului care, daca este scoasa din graf (fara a elimina si cele doua noduri cu care este adiacenta), graful nu mai ramane conex.

O **componenta biconexa** a unui graf  $G$  este un subgraf indus maximal conex cu proprietatea ca orice nod ce apartine subgrafului nu este nod critic relativ la subgraful respectiv. Componenta biconexa este maximala in acelasi sens precizat anterior (la definitia unei componente conexe).

**Observatie:** Se poate verifica usor ca subgraful indus format din doua noduri legate intre ele printr-o muchie critica a unui graf  $G$  este o componenta biconexa a lui  $G$ .

Orice graf (neorientat & conex) poate fi partitionat in mod unic in componente biconexe (in asa fel incat fiecare muchie a grafului sa faca parte din exact o componenta biconexa). Unele noduri ale grafului pot face parte

din mai multe componente biconexe. Mai exact, fiecare nod critic face parte din cel puțin două componente, iar orice nod care nu este critic face parte din exact o componentă.



**Figura 2. Partitionarea unui graf in componente biconexe**

Un nod  $y$  este **vecin** cu un nod  $x$ , dacă există muchia  $(y,x)$ .

**Gradul** unui nod  $x$  este definit ca fiind numărul de noduri  $y$  cu care  $x$  este vecin.

Un **drum simplu** este un drum în care fiecare nod al grafului apare cel mult o dată în cadrul succesiunii de noduri.

Un **ciclu** este un drum ce conține cel puțin 3 noduri, iar primul și ultimul nod sunt legate printr-o muchie.

Un **ciclu simplu** este un ciclu în care fiecare nod al grafului apare cel mult o dată.

Un **arbore** este un graf (neorientat & conex) care nu conține cicluri.

Un arbore poate fi considerat ca având un nod special, numit **radacina**. Nodurile arborelui pot fi apoi distribuite pe niveluri, considerând că nodul radacina are nivelul 1, vecinii nodului radacina au nivelul 2, vecinii vecinilor nodului radacina au nivelul 3 s.a.m.d. Dacă un nod are nivelul  $K$  ( $K > 1$ ), atunci el este legat printr-o muchie de un singur nod de pe nivelul  $K-1$ . Vom numi acest nod **parintele** (sau **tatal**) nodului respectiv. Dacă un nod are nivelul  $K$  ( $K \geq 1$ ), orice vecin al acestuia de pe nivelul  $K+1$  se numește **fiu** al nodului respectiv. Un nod al unui arbore cu radacina fixată se numește **frunza** dacă el nu are nici un fiu.

Un **arbore partial** al unui graf  $G(V,E)$  este un arbore  $A(V,E')$ , cu proprietatea că  $E'$  este inclus în  $E$ .

Un **graf bipartit** este un graf  $G(V_1 \cup V_2, E)$  unde fiecare muchie are un capăt în  $V_1$  și celălalt în  $V_2$ . Orice arbore este un graf bipartit.

O **parcursere DF** a unui graf  $G$  constă într-o vizitare a tuturor nodurilor grafului, ce poate fi descrisă cu următorul algoritm recursiv (apelat pentru un nod oarecare din graf).

**DF (Nod  $x$ )**

1) marchează  $x$  ca fiind vizitat

2) pentru fiecare vecin  $y$  al lui  $x$ , care nu a fost marcat ca fiind vizitat, apelează DF( $y$ )

Complexitatea algoritmului de parcurgere DF a unui graf este de ordinul  $O(N+M)$ , unde  $N$  este numărul de noduri ale grafului, iar  $M$  este numărul de muchii.

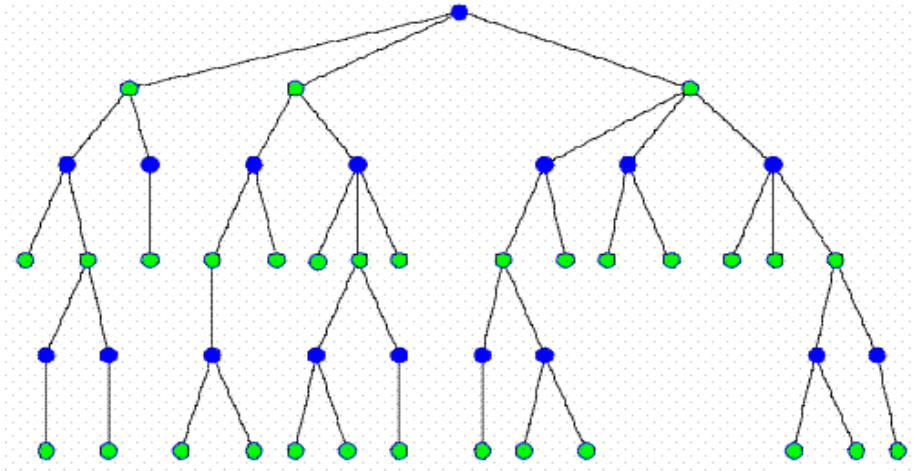


Figura 3. Un arbore

## 2. Determinarea nodurilor critice și a muchiiilor critice ale unui graf neorientat & conex

Nodurile și muchiiile critice ale unui graf se pot determina în mod eficient folosindu-ne de proprietățile parcurgerii DF.

Se face o parcurgere DF dintr-un nod oarecare al grafului, pe care îl vom numi în continuare *nod radacina*. Știm că o parcurgere DF determină un arbore parțial al grafului, având radacina în *nodul radacina*. În cadrul parcurgerii, vom calcula nivelul pe care se află fiecare nod (nivelul radacinii este 1, iar nivelul fiecărui nod este egal cu nivelul nodului tată plus 1).

Ca urmare a unei parcurgeri DF, muchiile grafului pot fi împartite în două categorii : muchii ce fac parte din arborele DF (*muchii înainte* sau *muchii directe*) și muchii ce nu fac parte din acest arbore (*muchii de întoarcere*, *muchii înapoi* sau *muchii inverse*). Pentru fiecare nod vom calcula, pe lângă nivelul acestuia, și nivelul minim la care poate ajunge nodul respectiv mergând numai pe muchii directe din subarborele sau din cadrul arborelui DF și folosind ca ultima muchie o muchie inversă. Această valoare, pe care o vom numi nivelul minim accesibil este egală cu minimul dintre următoarele 3 valori :

- nivelul nodului curent
- minimul dintre nivelurile nodurilor cu care este legat nodul curent printr-o muchie inversă
- minimul dintre nivelurile minime accesibile ale fiilor nodului curent din cadrul arborelui DF

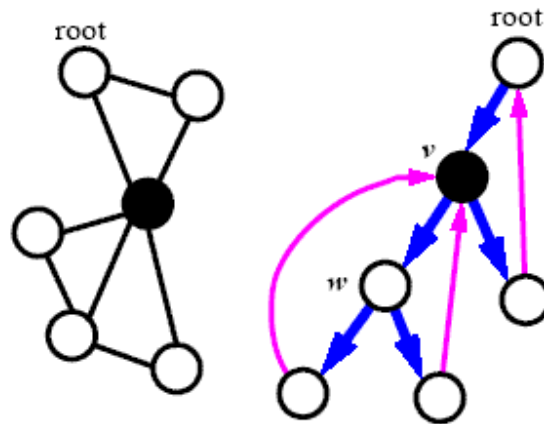
Se poate observa ușor că un nod (diferit de radacina) este critic dacă are cel puțin un fiu pentru care nivelul minim accesibil este mai mare sau egal cu nivelul nodului. Radacina este nod critic dacă are cel puțin 2 fii.

Orice muchie critică face parte din orice arbore DF. Condiția pentru ca o muchie  $(u,v)$  să fie critică este ca  $u$  să fie tatăl lui  $v$  în arborele DF, iar nivelul minim accesibil al nodului  $v$  să fie strict mai mare decât nivelul lui  $u$ .

Determinarea nodurilor și a muchiiilor critice se poate realiza, astfel, cu o complexitate  $O(N+M)$ , unde  $N$  este numărul de noduri ale grafului, iar  $M$  este numărul de muchii ale grafului.

## 3. Determinarea componentelor biconexe ale unui graf neorientat & conex

Pentru a determina componentele biconexe ale unui graf folosim tot o parcurgere DF, precum și o stivă de muchii. Algoritmul în pseudocod este următorul:



**Figura 4. Arborele DF, muchii directe si muchii inverse. Determinarea ca un nod este critic**

DF(Nod x)

1)marcheaza x ca fiind vizitat

2)pentru fiecare y vecin al lui x

2.1)daca y NU este tatal lui x

2.1.1)daca y a fost deja vizitat si nivelul lui y este mai mic decat nivelul lui x

2.1.1.1)pune muchia (x,y) in stiva

2.1.2)daca y NU a fost deja vizitat

2.1.2.1)memoreaza ca tatal lui y este x

2.1.2.2)pune muchia (x,y) in stiva

2.1.2.3)DF(y)

2.1.2.4)daca x este nod critic relativ la y (mai exact, daca nivelul minim accesibil al lui y este mai mare sau egal cu nivelul lui x)

2.1.2.4.1)incepe o componenta biconexa noua

2.1.2.4.2)cat timp in varful stivei nu este muchia (x,y)

2.1.2.4.2.1)scoate muchia din varful stivei si marcheaz-o ca facand parte din componenta biconexa curenta

2.1.2.4.2.1)scoate din varful stivei muchia (x,y) si marcheaz-o ca facand parte din componenta biconexa curenta

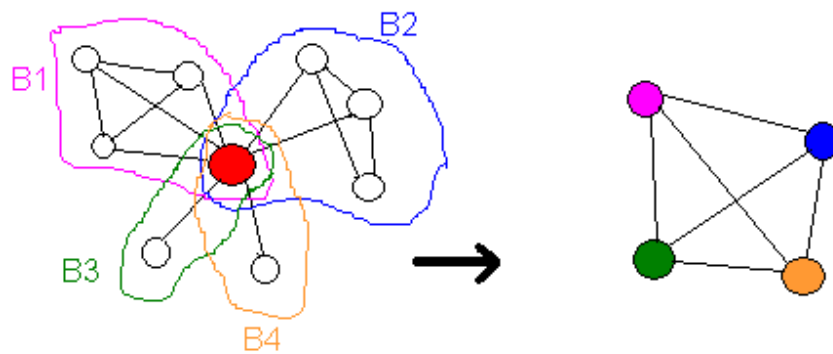
## 4. Arborele componentelor biconexe si al nodurilor critice

Intuitiv, privind partitionarea unui graf in componente biconexe, observam un fel de structura arborescenta. Totusi, cand incercam sa o formalizam, constatam cateva dificultati.

### 4.1. Graful componentelor biconexe

Consideram graful  $G'$  asociat unui graf  $G$ . Fiecare nod al lui  $G'$  corespunde unei componente biconexe a lui  $G$ . Exista muchie între 2 noduri din  $G'$  daca cele 2 componente biconexe corespunzatoare au un nod critic in comun. Aceasta idee corespunde unei partitionari de genul : componente biconexe = noduri , noduri critice = muchii. Totusi, privind figura 5, observam ca nu obtinem rezultatul dorit.

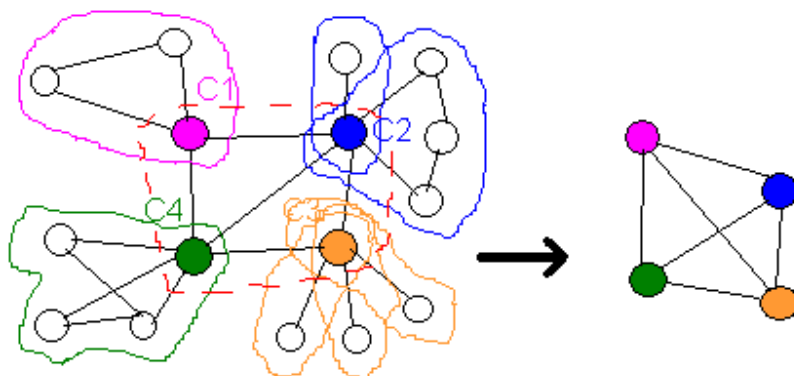
Intrucat mai multe componente biconexe pot avea in comun acelasi nod critic, un nod critic nu devine o singura muchie in  $G'$ , ci poate deveni o clica, ceea ce nu este de dorit, deoarece clicile nu sunt, in general, structuri cu care se lucreaza usor.



**Figura 5. Graful componentelor biconexe**

#### 4.2. Graful nodurilor critice

Consideram graful  $G'$  asociat unui graf  $G$ . Fiecare nod al lui  $G'$  corespunde unui nod critic din  $G$ . Exista muchie între 2 noduri din  $G'$  dacă exista o componenta biconexa din care să facă parte cele 2 noduri critice corespunzătoare celor două noduri din  $G'$ . Aceasta idee corespunde unei partitionari de genul : noduri critice = noduri, componente biconexe = muchii. Totuși, privind figura 6, observăm că nici de data aceasta nu obținem rezultatul dorit.



**Figura 6. Graful nodurilor critice**

Ne-am fi dorit ca o componenta biconexa să se transforme într-o muchie. Totuși, deoarece se pot afla mai multe noduri critice în aceeași componenta biconexa, o componenta biconexa se transformă, de fapt, într-o clică.

#### 4.3. Graful componentelor biconexe și al nodurilor critice

Considerăm în continuare următorul graf bipartit asociat unui graf  $G$  : toate nodurile din partea stângă corespund unei componente biconexe a lui  $G$  și toate nodurile din partea dreaptă corespund unui nod critic al lui  $G$ . Avem muchie între un nod  $u$  din partea stângă și un nod  $v$  din partea dreaptă, dacă nodul critic corespunzător lui  $v$  face parte din componenta biconexa corespunzătoare lui  $u$ . Se poate observa (și demonstra) ușor că acest graf bipartit al componentelor biconexe și nodurilor critice este un arbore.



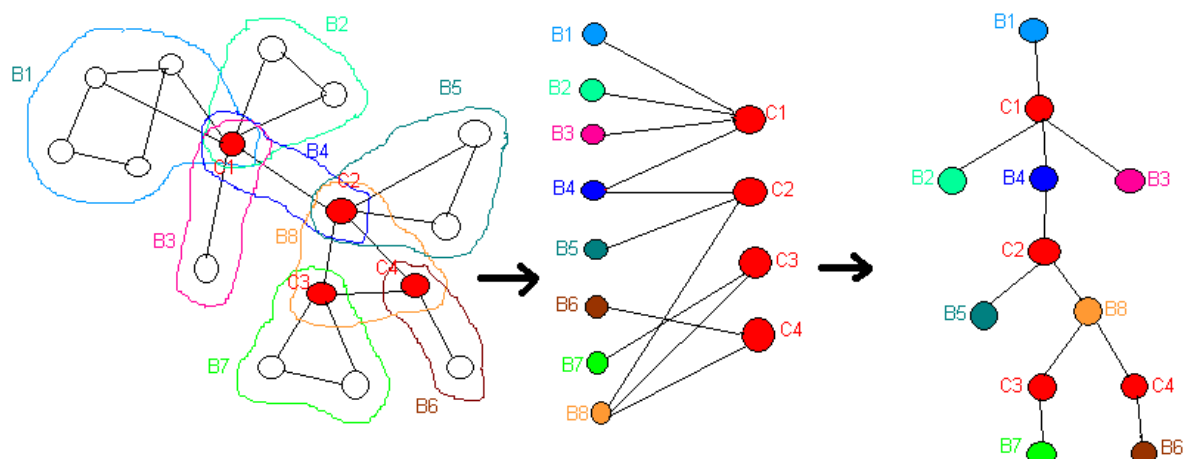


Figura 7. Componentele biconexe si nodurile critice ale unui graf. Graful si arborele componentelor biconexe si al nodurilor critice

## 5. Descompunerea arborescenta a unui graf

O **descompunere arborescenta** a unui graf  $G$  consta in construirea unui arbore  $T$ , in care nodurile arborelui sunt reprezentate de submultimi de noduri ale lui  $G$ . Sa consideram ca nodurile arborelui sunt  $X_1, X_2, \dots, X_Q$ . Descompunerea arborescenta a lui  $G$  are urmatoarele proprietati:

- 1) Pentru orice nod  $u$  al lui  $G$ , exista cel putin un nod  $X_i$  al lui  $T$  astfel incat  $u$  apartine lui  $X_i$
- 2) Pentru orice muchie  $(u, v)$  a lui  $G$  exista cel putin un nod  $X_i$  al lui  $T$  astfel incat atat  $u$ , cat si  $v$  apartin lui  $X_i$
- 3) Oricare ar fi doua noduri ale lui  $T$ ,  $X_i$  si  $X_j$  si un al treilea nod  $X_k$  ce se afla pe drumul de la  $X_i$  la  $X_j$ , are loc proprietatea :  $X_i \cap X_j$  este inclus in  $X_k$ .

Proprietatea 3 poate fi reformulata sub forma 3' :

- 3') Oricare ar fi doua noduri ale lui  $T$ ,  $X_i$  si  $X_j$ , daca un nod  $u$  al lui  $G$  apartine atat lui  $X_i$ , cat si lui  $X_j$ , atunci el apartine tuturor nodurilor lui  $T$  ce se afla pe drumul dintre  $X_i$  si  $X_j$ .

Orice graf  $G$  are cel putin o descompunere arborescenta. Aceasta afirmatie este banala, deoarece nu s-a impus nici o limita pentru numarul de noduri ale arborelui  $T$ . Astfel, se poate construi un arbore cu un singur nod, ce corespunde multimii tuturor nodurilor lui  $G$ .

Pentru o descompunere arborescenta, se defineste **dimensiunea** acesteia ca fiind  $\max\{|X_i| - 1\}$ . Arborele ce contine un singur nod (deci tot graful intr-un singur nod) are dimensiunea maxima. In general, ne intereseaza descompuneri arborescente de dimensiune cat mai mica. Determinarea descompunerii arborescente de dimensiune minima nu se poate realiza, insa, in timp polinomial (decat pentru clase particulare de grafuri).

Folosind o descompunere arborescenta a unui graf, putem rezolva mai usor diverse probleme care se pot rezolva in timp polinomial pentru arbori, dar nu se pot rezolva in timp polinomial pentru grafuri generale. De exemplu, determinarea celui mai lung drum, determinarea celei mai mari clici, determinarea celei mai mari multimii intern stabile, determinarea unei acoperiri a tuturor muchiilor cu numar minim de noduri, realizarea unei colorari cu numar minim de culori sunt doar cateva probleme ce pot fi rezolvate in timp polinomial pentru arbori, dar care sunt **NP** pentru grafuri generale. Folosind o descompunere arborescenta, putem rezolva aceste probleme mai eficient, folosind algoritmi specifici arborilor, ajungand la o complexitate exponentiala (de cele mai multe ori) in dimensiunea descompunerii si nu in numarul de noduri ale grafului. Din acest motiv ne intereseaza descompuneri arborescente avand dimensiuni cat mai mici.

Sa analizam in continuare daca graful componentelor biconexe si al nodurilor critice (pe care il vom numi de acum inainte arborele componentelor biconexe si al nodurilor critice) reprezinta o descompunere arborescenta.

Nodurile arborelui corespunzatoare componentelor biconexe contin ca submultime de noduri nodurile din componenta respectiva, iar nodurile corespunzatoare nodurilor critice contin ca submultime de noduri o submultime ce are un singur element : nodul critic respectiv.

Primele 2 proprietati reies imediat din definitia componentelor biconexe. Sa verificam in continuare a treia proprietate :

- orice nod  $u$  care nu este nod critic face parte din exact un nod al arborelui, deci verifica proprietatea 3
- orice nod critic  $u$  face parte dintr-un nod al arborelui corespunzator nodului critic respectiv si din mai multe componente biconexe ; oricare 2 componente biconexe,  $B_i$  si  $B_j$  din care face parte nodul  $u$ , se afla la distanta de 2 muchii in arbore si singurul nod intermediar de pe drumul dintre ele este chiar nodul corespunzator nodului critic  $u \Rightarrow$  proprietatea este verificata

Arborele componentelor biconexe si al nodurilor critice reprezinta o buna modalitate de a reprezenta un graf sub forma unui arbore. In general, dimensiunea descompunerii arborescente obtinute este mai mica decat numarul de noduri ale grafului minus 1, inasa, pentru grafuri indeajuns de dense, dimensiunea este, in continuare, prea mare. Pentru a folosi acest arbore in rezolvarea de probleme ce au complexitate exponentiala in dimensiunea descompunerii arborescente, trebuie sa avem informatii suplimentare despre numarul maxim de noduri dintr-o componenta biconexa.

Exista algoritmi care incearca sa realizeze o descompunere arborescenta de dimensiune cat mai mica a unui graf si exista si clase de grafuri pentru care exista descompuneri arborescente cu proprietati speciale (specifice clasei de grafuri respective).

## 6. Cateva probleme rezolvabile folosind arborele componentelor biconexe si al nodurilor critice

**6.1.** Se da un graf neorientat si conex, in care fiecare muchie are o anumita lungime. Se mai stie ca fiecare componenta biconexa are cel mult 9 noduri. Se doreste determinarea celui mai lung drum simplu din acest graf (nu se specifica capatele acestui drum ; ele pot fi orice pereche de noduri din graf)

Aceasta problema (intr-o forma “imbracata”) a fost data la tabara de pregatire comuna a loturilor de informatica din Cehia, Polonia si Slovacia (CPSPC), in anul 2000. Pentru a o rezolva, construim arborele componentelor biconexe si al nodurilor critice corespunzator grafului dat. Apoi, pentru fiecare componenta biconexa, pentru fiecare pereche de noduri  $(u,v)$  ce fac parte din componenta, determinam (prin backtracking, generand permutari ale nodurilor, cu o complexitate  $9!$ , sau cu programare dinamica, cu o complexitate de cel mult  $9^2 \cdot 2^9$ ) cel mai lung drum (simplu) care uneste nodurile  $u$  si  $v$  si trece numai prin alte noduri din componenta respectiva (sa notam aceasta distanta calculata cu  $D[x, u, v]$ , unde  $x$  este nodul din arbore corespunzator componentei biconexe). Urmeaza acum etapa de programare dinamica pe arborele construit.

In mod similar cazului cand graful ar fi arbore, vom calcula, pentru fiecare nod  $x$  al arborelui construit, doua valori :

- $A[x, u]$  = lungimea celui mai lung drum care incepe undeva mai jos in arbore si se termina la nodul  $x$ , in nodul  $u$  al grafului din nodul  $x$  al arborelui
- $B[x]$  = lungimea celui mai lung drum care incepe undeva mai jos in arbore, trece prin nodul  $x$  si se termina undeva mai jos in arbore

Pentru un nod  $x$  de tipul “**nod critic**” :

- $A[x, u]$  (cu  $u$  nodul critic respectiv) se calculeaza ca fiind maximul dintre  $A[y, u]$  cu  $y$  fiu al lui  $x$
- $B[x] = \max\{A[y, u]\} + \max\{A[y, u]\}$  , cu  $y$  fiu al lui  $x$  (adica suma dintre primul maxim si al doilea maxim)

Pentru un nod  $x$  de tipul “**componenta biconexa**” :

- $A[x, u] = \max\{D[x, t, u] + \{0 - \text{daca } t \text{ nu e nod critic ; } A[y, t] - \text{daca } t \text{ este nod critic, } y \text{ fiind nodul din arbore corespunzator nodului critic } t\}\}$  , cu  $u$  si  $t$  noduri din componenta corespunzatoare lui  $x$
- $B[x] = \max\{D[x, u, v] + \{0 - \text{daca } u \text{ nu e nod critic ; } A[y, u] - \text{daca } u \text{ este nod critic, } y \text{ fiind nodul din arbore corespunzator nodului critic } u, \text{ si } y \text{ fiu al lui } x\} + \{0 - \text{daca } v \text{ nu e nod critic ; } A[y, v] - \text{daca } v \text{ este}$

nod critic,  $y$  fiind nodul din arbore corespunzator nodului critic  $v$ , si  $y$  fiu al lui  $x$  } }, cu  $u$  si  $v$  noduri din componenta corespunzatoare lui  $x$

Drumul maxim din graf este egal cu maximul dintre toate valorile  $A[x, u]$  si  $B[x]$ . Daca drumul trebuie si reconstituit, problema devine un pic mai complicata. Trebuie gasit nodul  $x$  din care s-a determinat maximul, apoi trebuie determinat cum s-a calculat valoarea  $A$  sau  $B$  din care a rezultat maximul (si drumul se reconstituie in mod recursiv, mergand in fiii lui  $x$ ).

**6.2.** Se da un graf neorientat & conex. Sa se coloreze nodurile grafului folosind un numar minim de culori, cu conditia ca oricare doua noduri intre care exista muchie sa fie colorate in culori diferite. Se stie ca fiecare componenta biconexa contine cel mult **8** noduri.

Construim arborele componentelor biconexe si al nodurilor critice. Vom calcula apoi  $A[x] =$  numarul minim de culori cu care se poate colora subarborele ce il are ca radacina pe nodul  $x$ .

Pentru un nod de tipul **“nod critic”**:

- $A[x] = \max \{ A[y] \}$ , cu  $y$  fiu al lui  $x$

Putem sa consideram ca nodul critic are culoarea 1. In fiecare componenta biconexa corespunzatoare fiecaruia din fii, nodul respectiv este colorat in culoarea 1, culorile celorlalte noduri permutandu-se, in cazul in care colorarea s-a calculat considerand ca nodul respectiv avea alta culoare.

Pentru un nod de tip **“componenta biconexa”**:

- se determina, prin backtracking (generand permutari si colorand in ordinea determinata de permutare sau generand colorari cu 1,2,3,...,8 culori si verificand apoi ca sunt valide) numarul minim de culori necesar pentru a colora componenta biconexa respectiva ; fie acest numar  $C$
- $A[x] = \max \{ C, \max \{ A[y] \} \}$ , cu  $y$  fiu al lui  $x$

Numarul minim de culori, sa-l notam  $K$ , este egal cu  $A[\text{radacina arborelui}]$ .

Daca trebuie reconstituita solutia, parcurgem arborele de la radacina in jos. Pentru fiecare nod  $x$  al arborelui, o parte din nodurile grafului corespunzatoare nodului arborelui vor fi deja colorate cand ajungem la nodul  $x$ . Mai trebuie doar sa generam o colorare valida cu maxim  $K$  culori pentru nodurile ramase necolorate si sa parcurgem, mai departe, fiii lui  $x$ .

**6.3.** Un cactus este un graf neorientat & conex cu proprietatea ca orice muchie a grafului face parte din cel mult un ciclu simplu. Dandu-se un cactus, determinati cate subgrafuri ale acestuia, care contin toate nodurile grafului, sunt, la randul lor, cactusi.

Problema a fost data la etapa regionala a concursului ACM din nord-estul Europei (NEERC), in anul 2005. Construim intai arborele componentelor biconexe si al nodurilor critice. Observam usor ca, intr-un cactus, orice componenta biconexa este ori o muchie critica, ori un ciclu simplu.

Vom calcula  $A[x]$ , reprezentand numarul de subgrafuri cactusi ce contin toate nodurile grafului ce se gasesc in noduri din subarborele nodului  $x$  al arborelui construit.

Pentru un nod de tip **“nod critic”**:

- $A[x] = \prod (A[y])$ , unde  $y$  este fiu al nodului  $x$  in arbore

$A[x]$  este produsul valorilor  $A[y]$ , cu  $y$  fiu al lui  $x$

Pentru un nod de tip **“componenta biconexa” – muchie critica**:

Sa presupunem ca muchia critica este  $(u,v)$ . Aceasta trebuie pastrata neaparat in cadrul oricarui subgraf cactus. Nodul  $x$  corespunzator muchiei critice poate fi radacina arborelui, caz in care  $A[x] = A[y]*A[z]$ , unde  $y$  si  $z$  sunt nodurile arborelui corespunzatoare nodurilor critice  $u$  si  $v$ . Daca  $u$  sau  $v$  nu sunt noduri critice (daca graful

contine doar muchia u-v, atunci nici u si nici v nu sunt critice ; daca unul din noduri are gradul 1, atunci nu este critic), atunci inlocuim  $A[y]$ , respectiv  $A[z]$ , cu valoarea 1.

Daca nodul x este frunza,  $A[x]=1$ . Altfel, nodul x are un singur fiu, y, de tip “nod critic” si  $A[x]=A[y]$ .

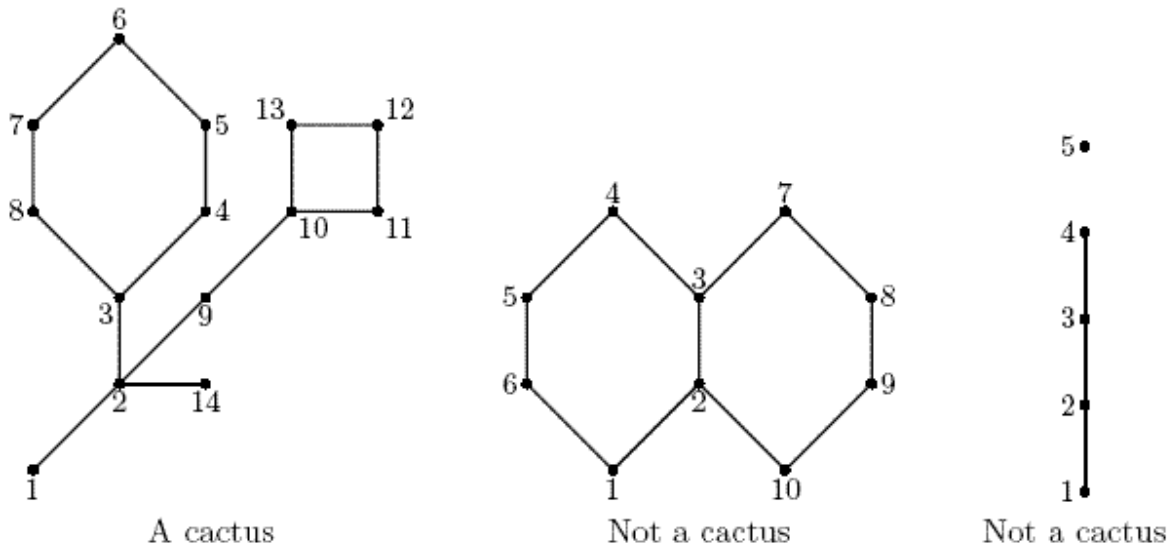


Figura 8. Exemple de grafuri care sunt cactusi si grafuri care nu sunt cactusi

Pentru un nod de tip “componenta biconexa” – ciclu:

Sa presupunem ca exista K noduri in componenta conexa respectiva. Deci, exista si K muchii. Daca s-ar elimina doua din aceste K muchii, s-ar obtine un subgraf care nu va mai fi conex. Asadar, se poate elimina cel mult 1 muchie. Exista, in total,  $K+1$  posibilitati (se poate elimina oricare din cele K muchii sau nu se elimina nici una din ele). Avem  $A[x] = (K+1) * \prod (A[y])$ , unde x este nodul corespunzator componentei biconexe, iar y este fiu al lui x.

Rezultatul il avem in  $A[\text{radacina arborelui}]$ .

In cazul acestei probleme, constatam ca folosirea arborelui componentelor biconexe si al nodurilor critice este o “masura exagerata”. Daca analizam mai atent algoritmul anterior, vom observa ca rezultatul este: *produs din { numarul de muchii ale fiecarui ciclu + 1}*. Asadar, tot ceea ce ne intereseaza este sa determinam ciclurile. Determinarea ciclurilor se poate face folosind algoritmul pentru gasirea componentelor biconexe sau folosind o parcurgere DF mai simpla.

## 7. Bibliografie & referinte

Arborele componentelor biconexe si al nodurilor critice a fost descoperit in mod independent de autorul acestui document, in luna februarie, 2006. In urma unor investigatii recente, autorul a descoperit, din nefericire, ca acest tip de arbore este deja cunoscut in literatura de specialitate si, pentru a cita dintr-unul din documentele citite, acest tip de arbore este “well-known”. Cateva denumiri ale acestui arbore sunt “block tree”, “block-cut vertex tree” sau “block-cut point tree”. Pentru informatii suplimentare despre acest tip de arbore, cautati cele 3 denumiri mentionate anterior pe Google, adaugand, eventual si cuvintele cheie “graph” sau “algorithm”.

- “Introduction to Algorithms”, Cormen, Leiserson & Rivest (<http://zhuzeyuan.hp.infoseek.co.jp/ita/toc.htm> , varianta online)
- “Gazeta de Informatica”, numerele din 1995-1998, unde autorul a intalnit pentru prima data algoritmul de determinare a componentelor biconexe
- “Culegere de probleme si programe PASCAL”, Cadar Cristian, Stroe Mihai
- <http://neerc.ifmo.ru/information/home.html>
- <http://cs.pub.ro/~sd/index.php?section=Laboratoare&file=Laborator%2012>