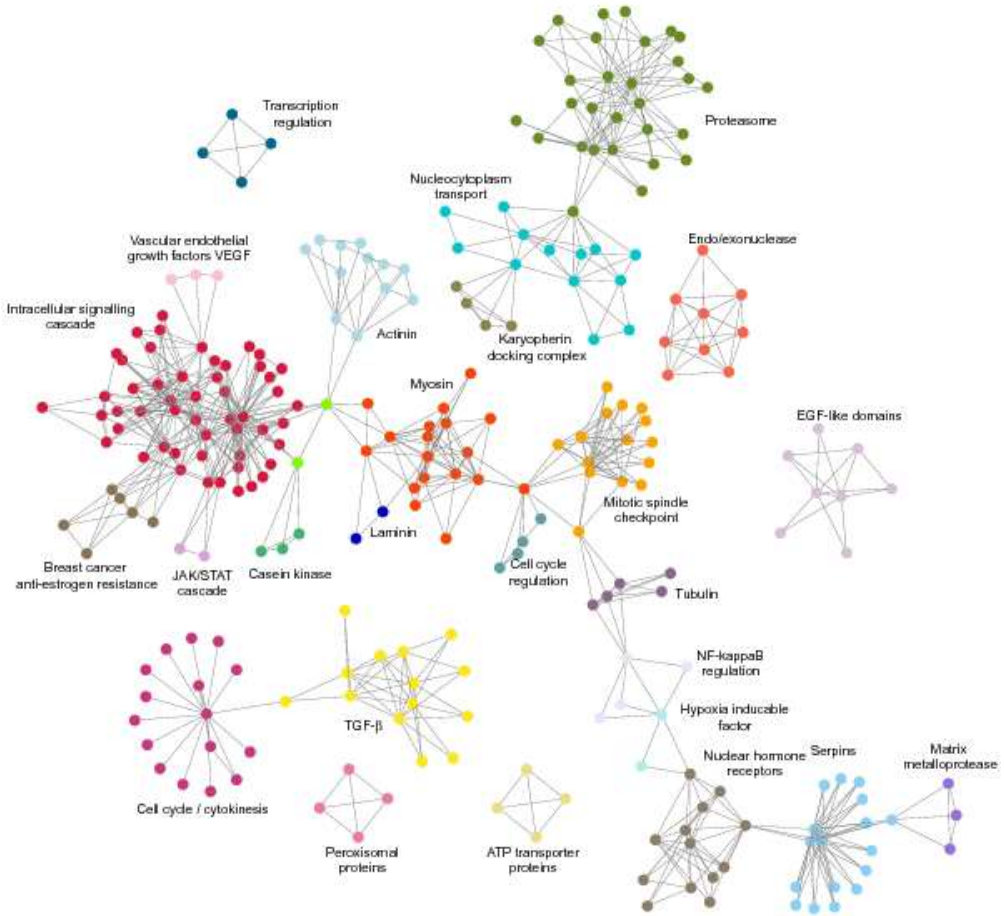
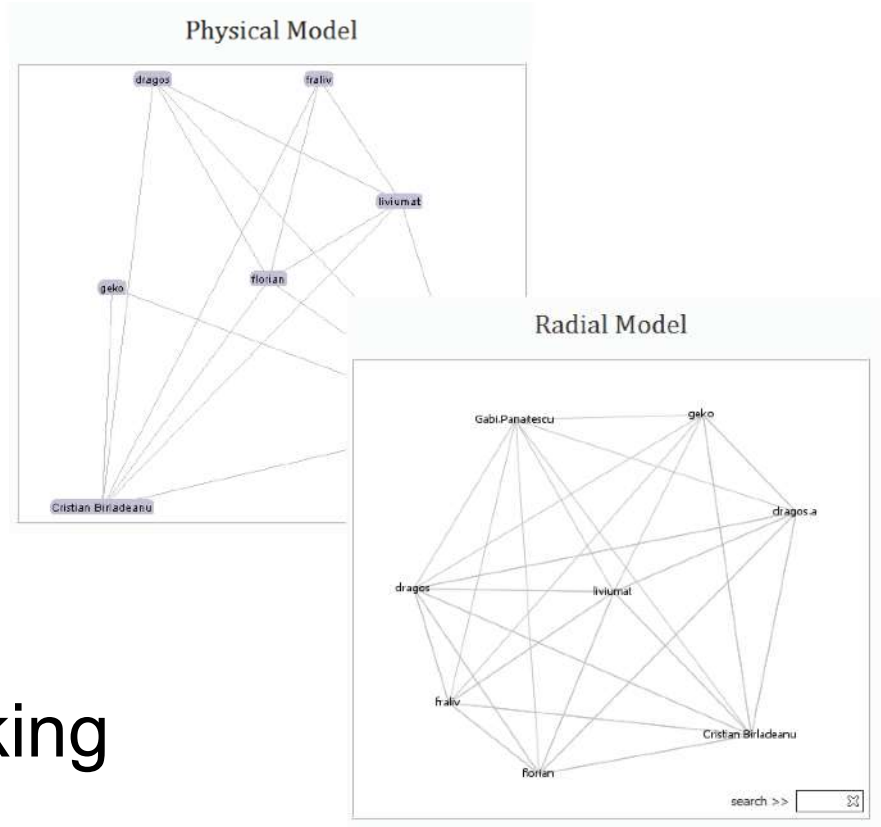
# Algoritmi pe grafuri - 1

Ștefan Trăușan-Matu


Transcription regulation
Proteasome
Nucleocytoplasm transport
Endo/exonuclease
Vascular endothelial growth factors VEGF
Intracellular signalling cascade
Actinin
Karyopherin docking complex
Mycosin
EGF-like domains
Mitotic spindle checkpoint
Laminin
Breast cancer anti-estrogen resistance
JAK/STAT cascade
Casein kinase
Cell cycle regulation
Tubulin
NF-kappaB regulation
Hypoxia inducable factor
TGF-β
Nuclear hormone receptors
Serpins
Matrix metalloprotease
Cell cycle / cytokinesis
Peroxisomal proteins
ATP transporter proteins

# Social Network Analysis

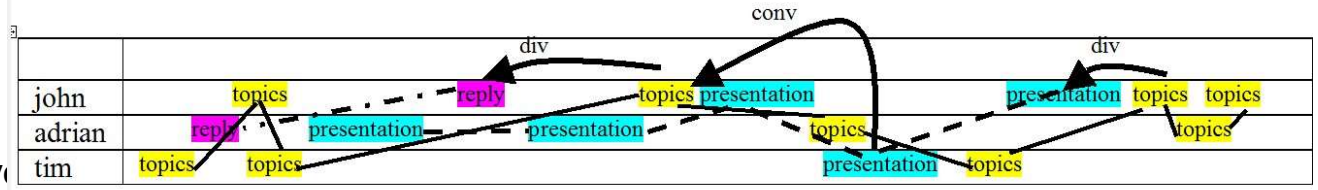
- Degree
- Centrality
  - Closeness
  - Graph
  - Eigen Value
- User Ranking
  - Google Page Ranking



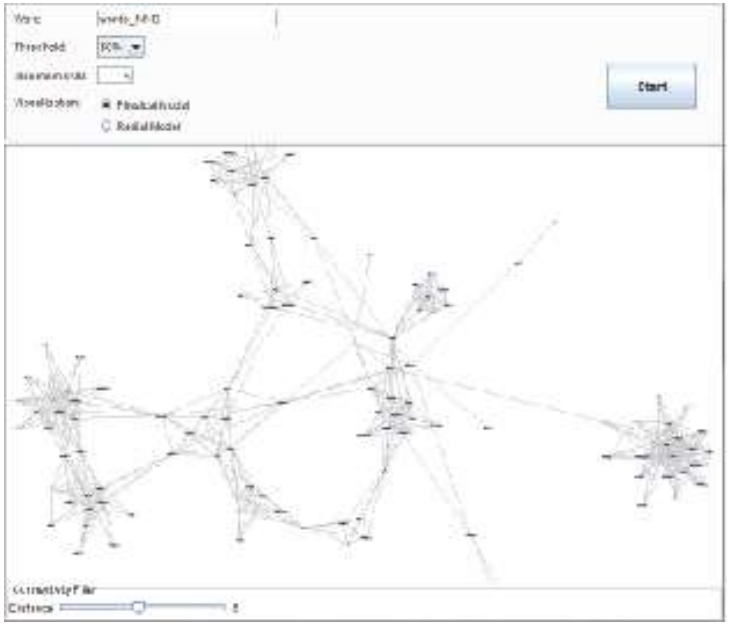
# Analiza limbajului natural – Polyphonic analysis

(Trausan-Matu & Stahl, 2007, http://gerrystahl.net/vmtwiki/stefan.pdf)

| Nr | Ref | Time | User | Text |
|---|---|---|---|---|
| 17 | | 10.26.25 | tim | You discussed about a topic separation |
| 18 | 15 | 10.26.37 | adrian | First of all, the reply method is cumbersome |
| 19 | 17 | 10.26.50 | john | yes.. because we did not like the way the topics were presented in concert chat |
| 20 | 18 | 10.26.56 | john | yes !! |
| 21 | 20 | 10.27.04 | john | i hate double-clicking ! |
| 22 | 20 | 10.27.18 | tim | and how can we find topics ? |
| 23 | 18 | 10.27.26 | adrian | What bothers me is the linear presentation of the discussin |
| 24 | 23 | 10.27.43 | john | Yep |
| 25 | 18 | 10.27.46 | adrian | and double-clicking too |
| 26 | | 10.27.54 | tim | You mean u want something like a chat forum ? :D |
| 27 | 24 | 10.27.58 | john | and the reply-to facility is supposed to help you |
| 28 | 18 | 10.28.15 | adrian | i'd like a tree presentation more |
| 29 | 18 | 10.28.38 | adrian | or maybe multiple chat columns, for each chat sub-thread |
| 30 | 27 | 10.28.58 | john | but it is really difficult to use in real time, because there are so many topics discussed which intertwine each other |
| 31 | 28 | 10.29.18 | john | i subscribe to a tree-like presentation form |
| 32 | P 30 | 10.29.20 | adrian | yes, that's why a clear separation of topics is needed |
| 33 | 31 | 10.29.47 | adrian | this is easy to implement, no problem here :) |
| 34 | 30 | 10.29.49 | tim | You need also a clever visual representation |
| 35 | 30 | 10.30.05 | tim | you'll need also a clever visual interface |
| 36 | | 10.30.22 | tim | Who decides the topics ? |
| 37 | 33 | 10.30.33 | john | i suppose you are refering to the visual representation, right ? |
| 38 | 37 | 10.30.45 | john | What i would like is a clever way to separate the topics :) |
| 39 | 38 | 10.30.59 | john | not just doing ot myself, manually |
| 40 | 37 | 10.31.00 | adrian | Yeah |
| 41 | 39 | 10.31.44 | adrian | When you start a new thread (a new message, non-related to other message), the app can assume a new topic |
| 42 | 39 | 10.31.46 | john | i would like the application to be able to detect w topic change all by itself |
| 43 | 42 | 10.32.01 | tim | That right |

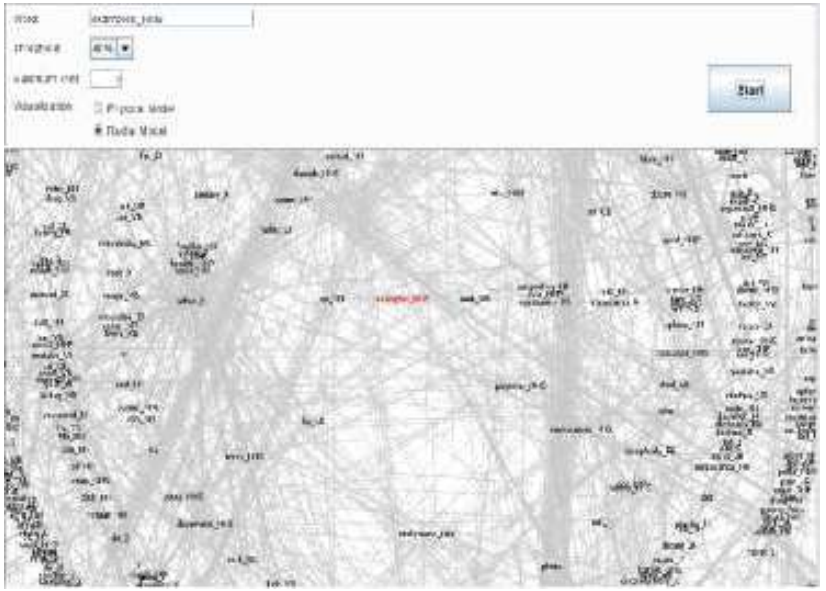


# Vector space visualization




Physical Model
- Relevance -

Radial Model
- Clustering -



# Tipuri de grafuri

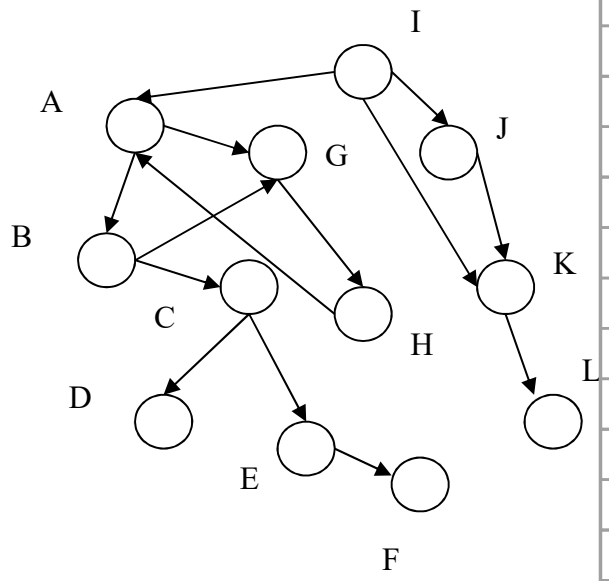
- Orientate
- Neorientate
- DAG (aciclice orientate)
- Cu ponderi (lungime, cost...) pe arce
- Ordonate
- Multigrafuri
- Arbori, păduri, liste

# Reprezentarea internă a grafurilor

- Matrice de adiacență
- Vector de adiacență
- Perechi de arce
- .....?

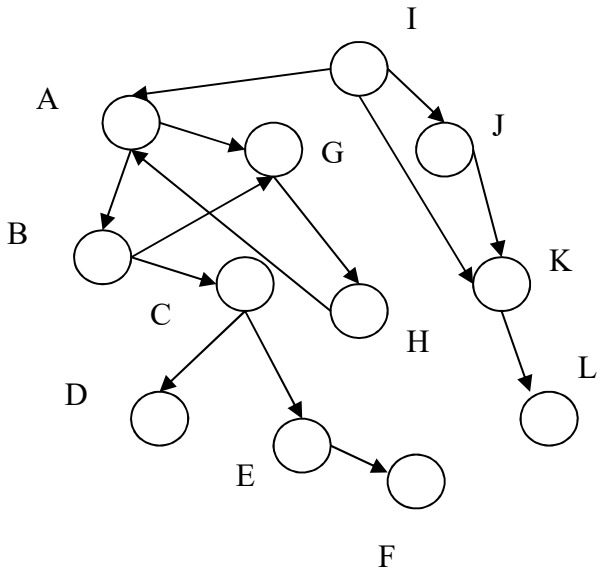
# Matrice de adiacență

Matricea este rară!



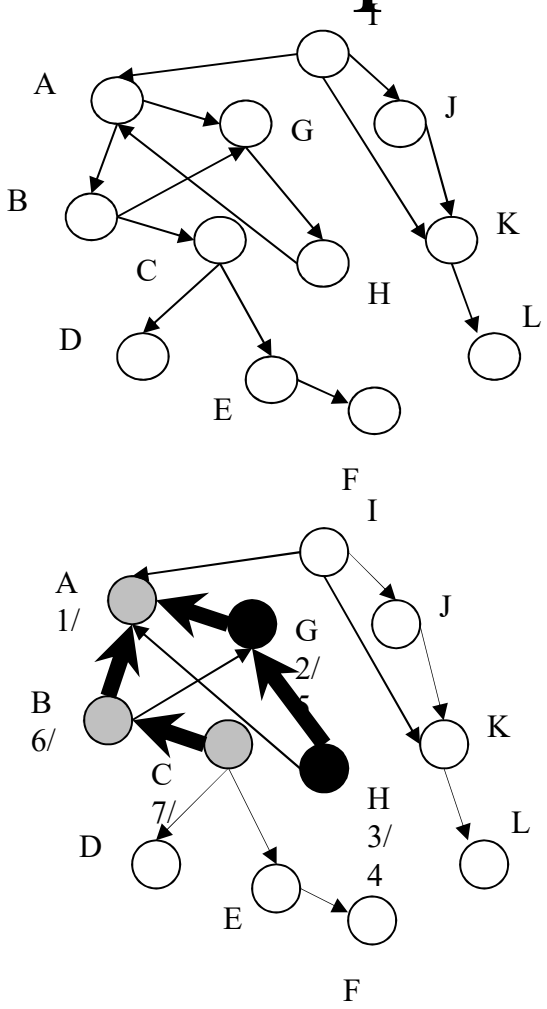

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **A** | | 1 | | | | | 1 | | | | | |
| **B** | | | 1 | | | | 1 | | | | | |
| **C** | | | | 1 | 1 | | | | | | | |
| **D** | | | | | | | | | | | | |
| **E** | | | | | | 1 | | | | | | |
| **F** | | | | | | | | | | | | |
| **G** | | | | | | | | 1 | | | | |
| **H** | 1 | | | | | | | | | | | |
| **I** | 1 | | | | | | | | | 1 | 1 | |
| **J** | | | | | | | | | | | 1 | |
| **K** | | | | | | | | | | | | 1 |
| **L** | | | | | | | | | | | | |

# Vector de adiacență




| | | | |
|---|---|---|---|
| **A** | **G** | **B** | |
| **B** | **C** | **G** | |
| **C** | **D** | **E** | |
| **D** | | | |
| **E** | **F** | | |
| **F** | | | |
| **G** | **H** | | |
| **H** | **A** | | |
| **I** | **A** | **J** | **K** |
| **J** | **K** | | |
| **K** | **L** | | |
| **L** | | | |

# O posibilă reprezentare

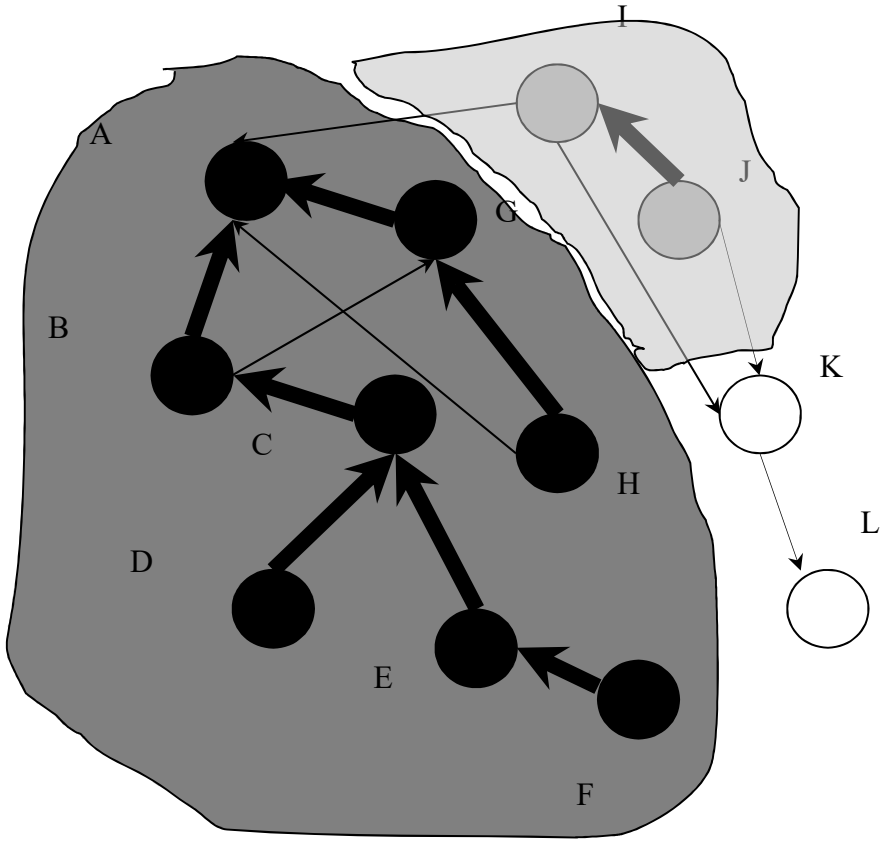



```
graf=[
['a','alb','nil',' ',' ',[6,1]],          0
['b','alb','nil',' ',' ',[2,6]],          1
['c','alb','nil',' ',' ',[3,4]],          2
['d','alb','nil',' ',' ',[]],             3
['e','alb','nil',' ',' ',[5]],            4
['f','alb','nil',' ',' ',[]],             5
['g','alb','nil',' ',' ',[7]],            6
['h','alb','nil',' ',' ',[0]],            7
['i','alb','nil',' ',' ',[0,9,10]],       8
['j','alb','nil',' ',' ',[10]],           9
['k','alb','nil',' ',' ',[11]],           10
['l','alb','nil',' ',' ',[]]]             11
```

**[nod, culoare, tată, debut, finis, lista_adiacențe]**

# Parcurgeri sistematice (”scanări”)

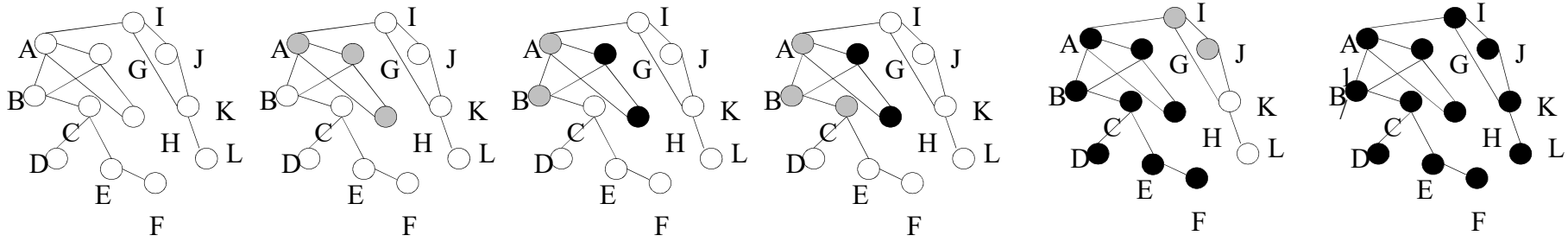
- În adâncime
- În lățime
- În adâncime iterativ
- După ”optim”
- .....?



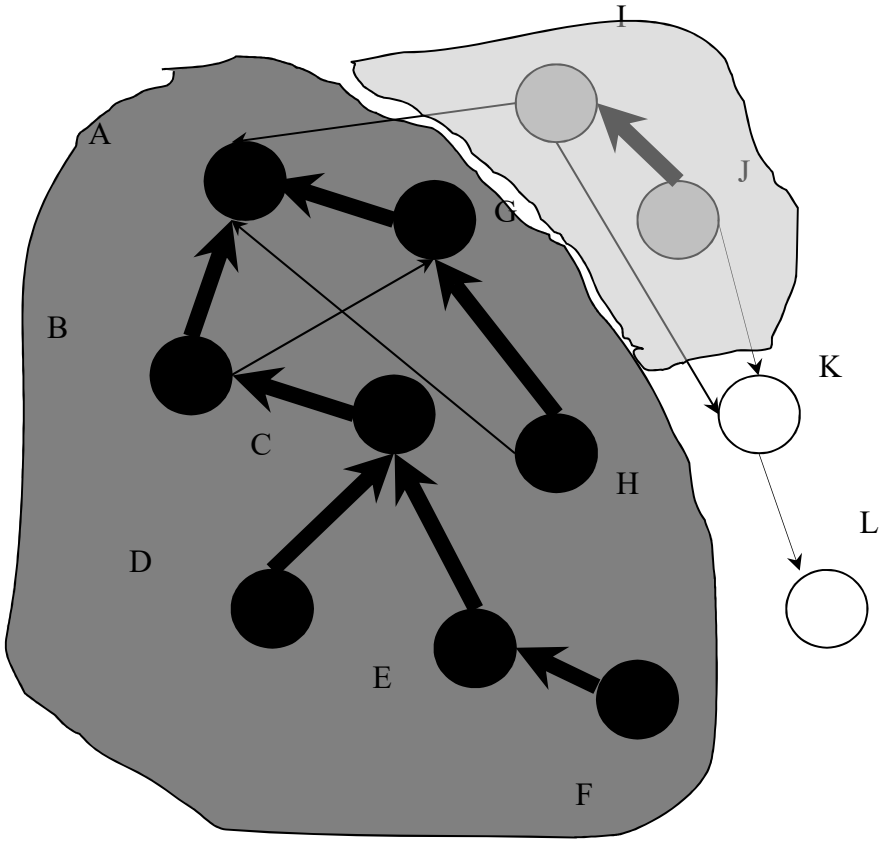
# Parcurgerea în adâncime

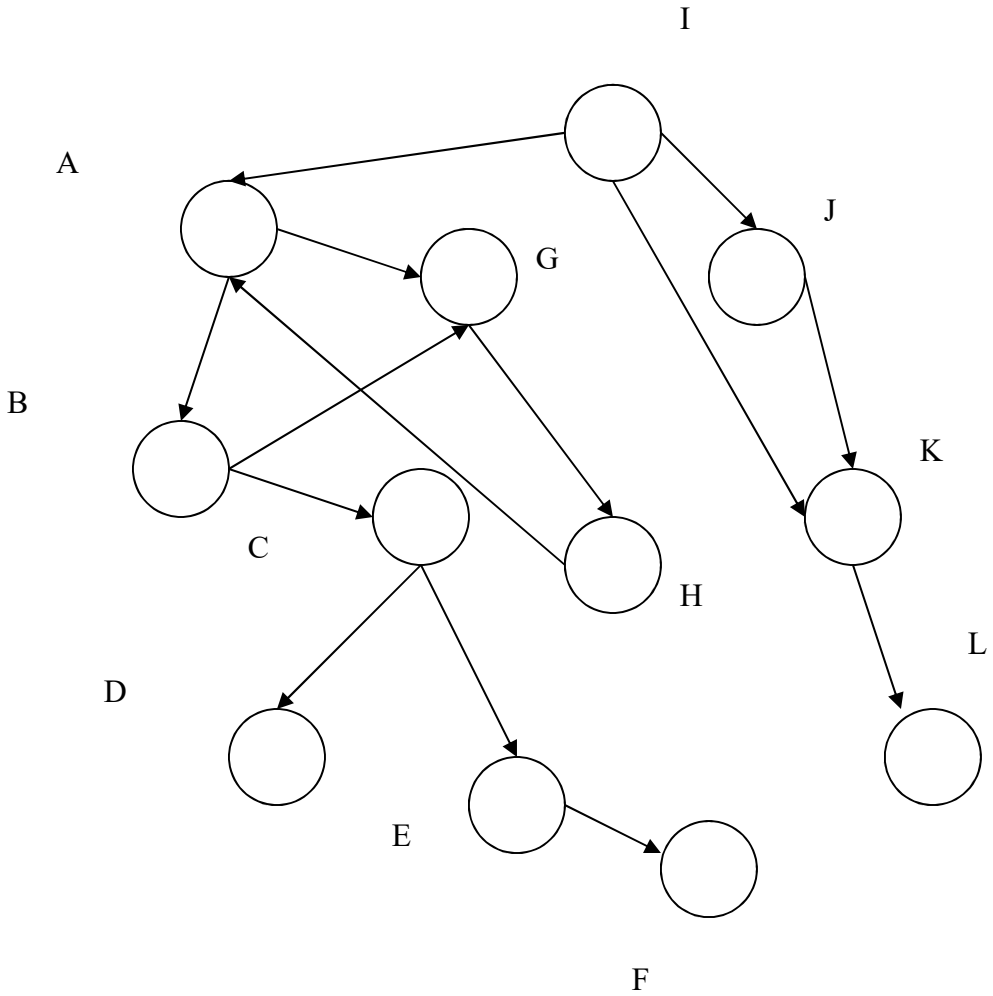
```
def parc_ad(g,n):
 for i in range(0,n-1):
   g[i][1]='alb'
   g[i][2]='nil'
 for i in range(0,n-1):
   if culoare(i)=='alb':
     parc_ad_rec(g,i)
```

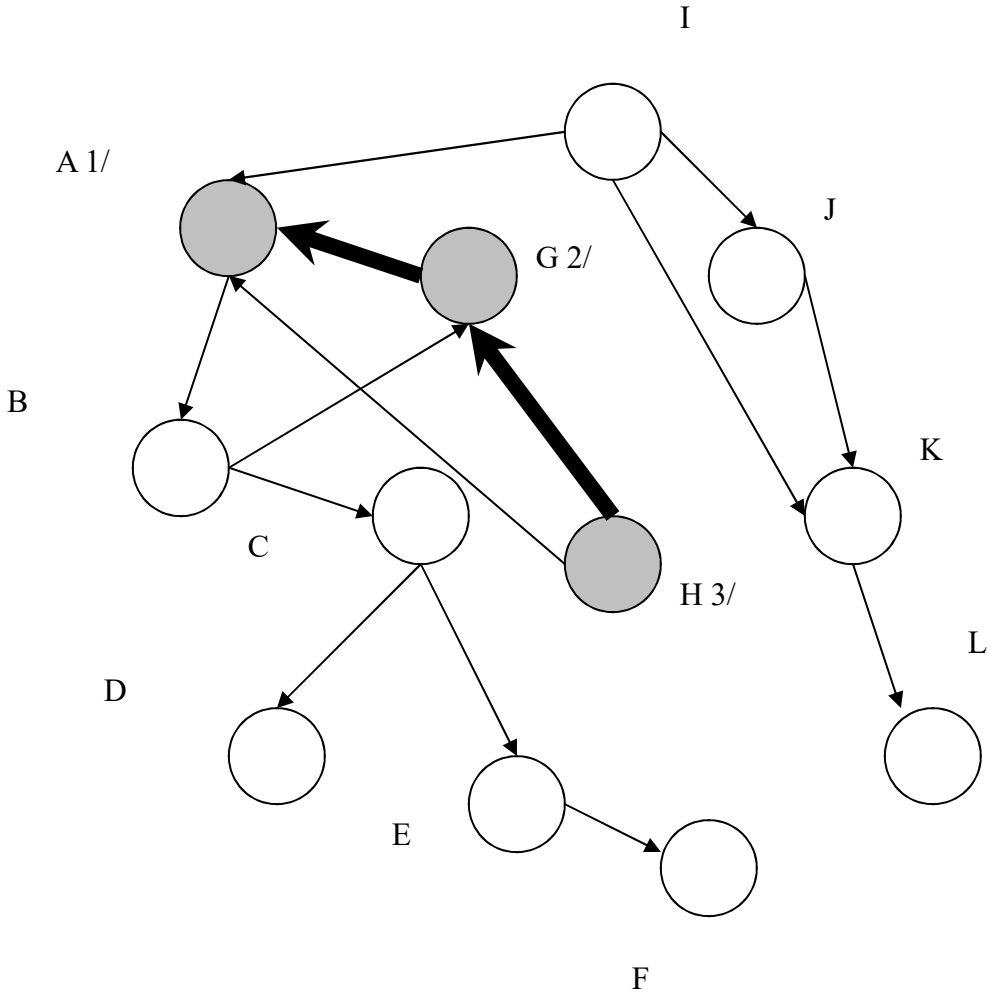
```
def parc_ad_rec(g,i):
 graf[i][3]=timp
 timp+=1
 graf[i][1]=‘gri’
 for j in adiacente (i):
  if culoare(j)=='alb':
   pune_tata(j,i)
   parc_ad_rec(g,j)
 graf[i][4]=timp
 timp+=1
 graf[i][1]=‘negru’
```

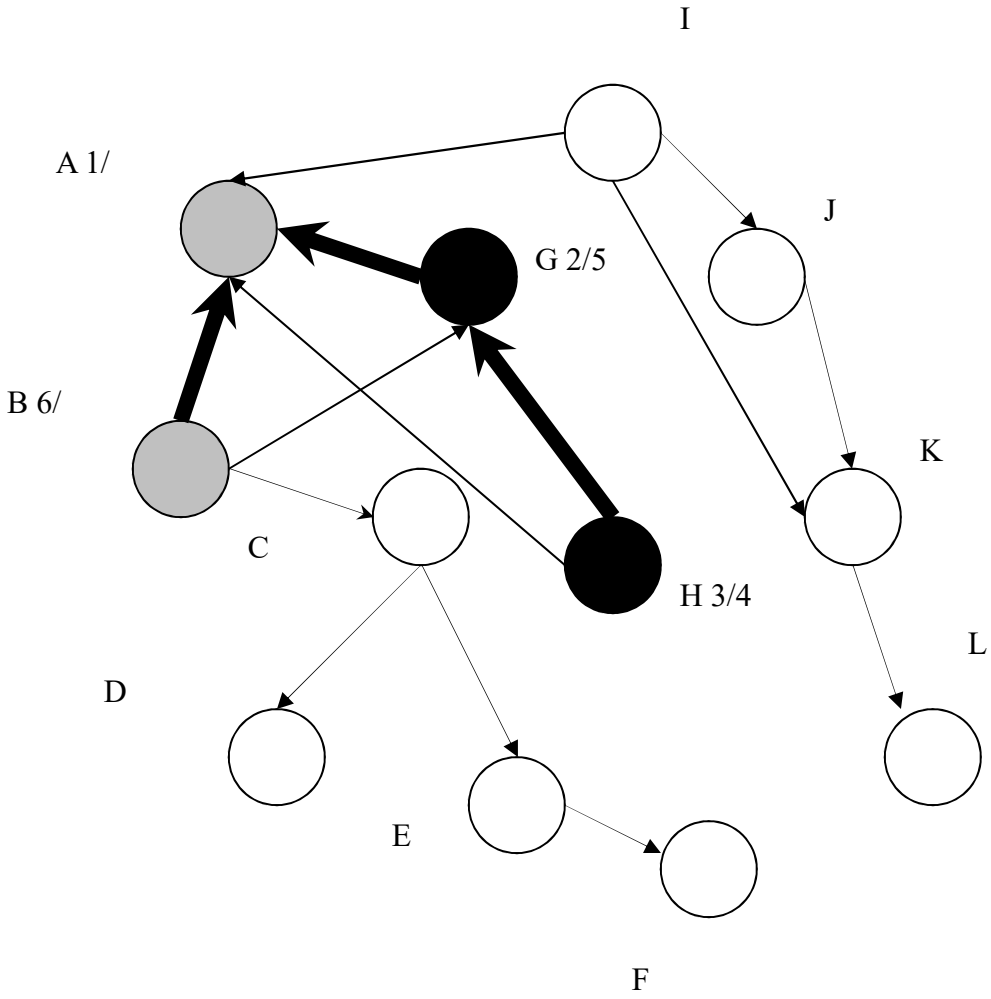

A
B
C
D
E
F
G
H
I
J
K
L

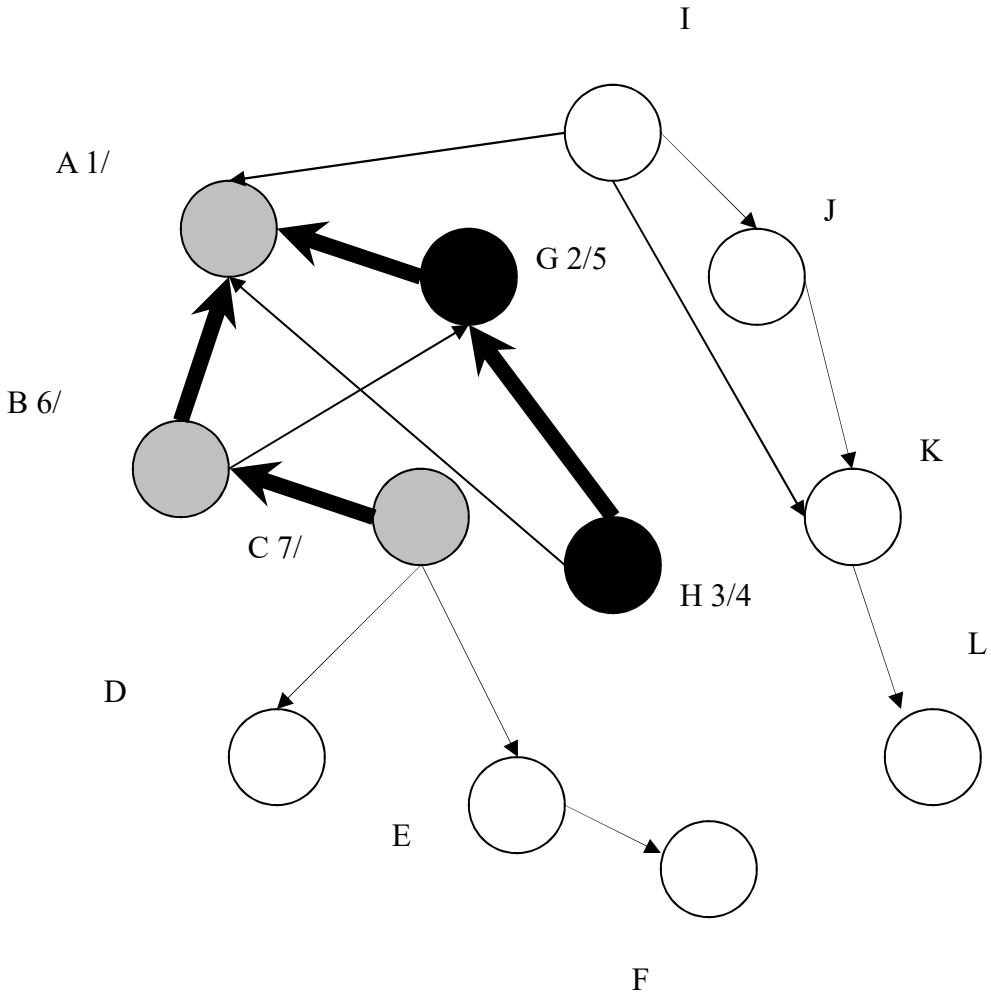
# Zonele parcurgerii grafului

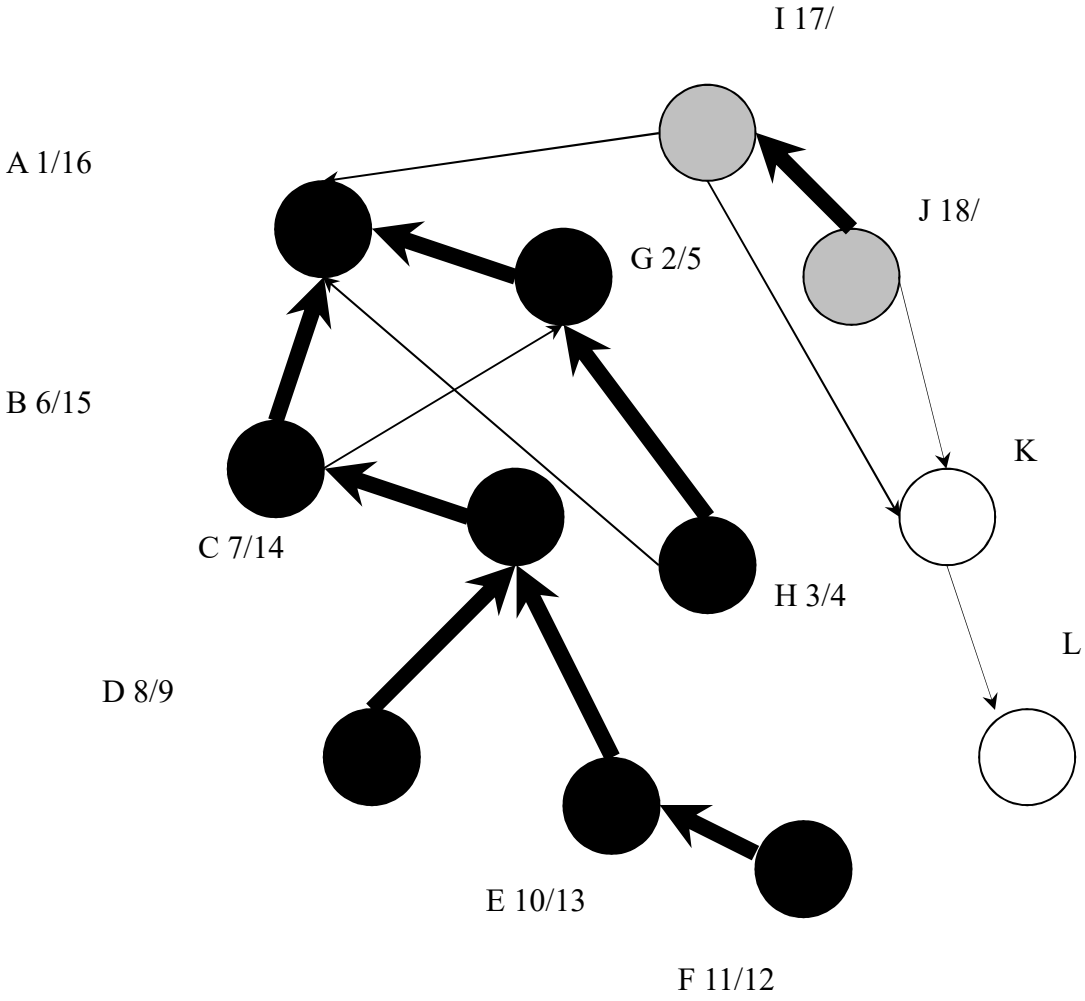


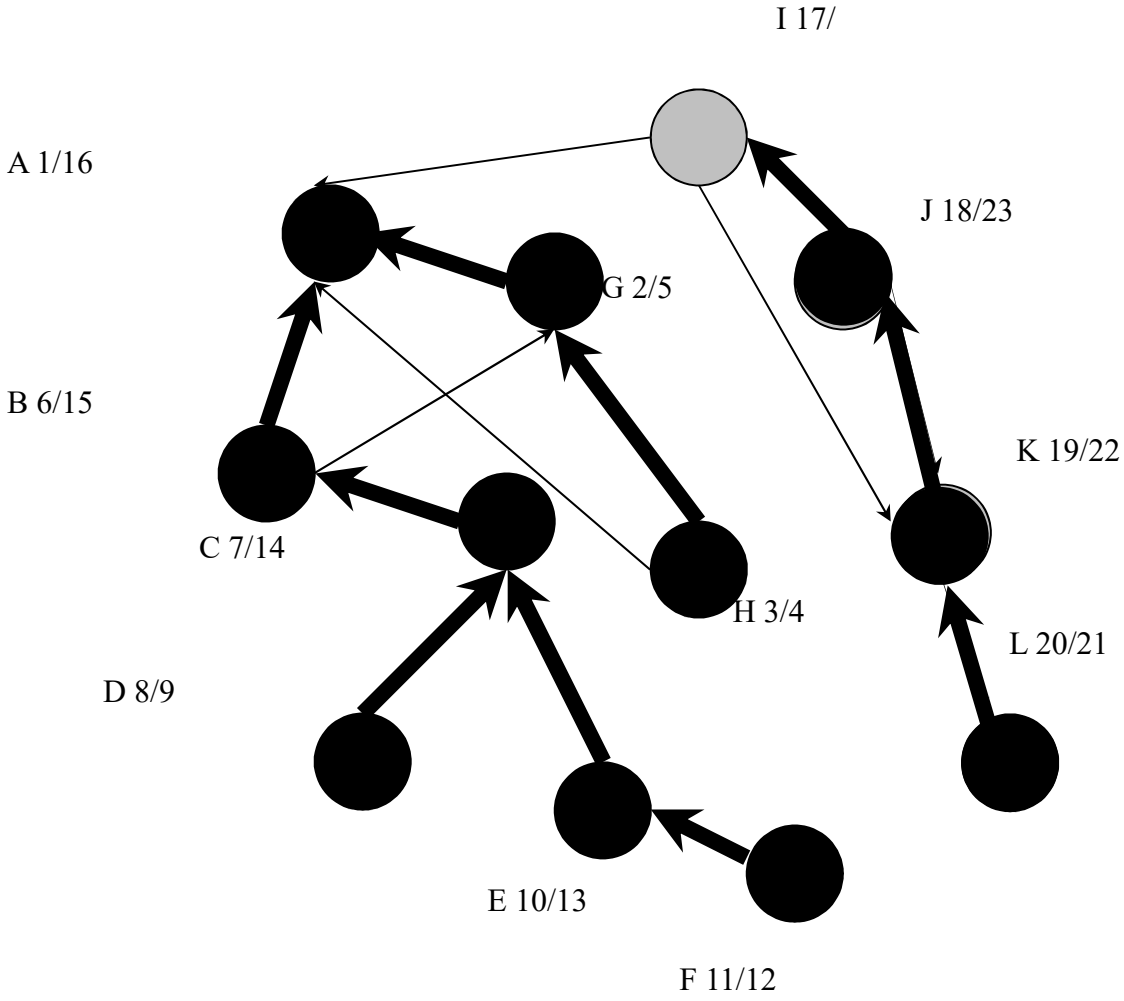

I
A
J
G
B
K
C
H
L
D
E
F

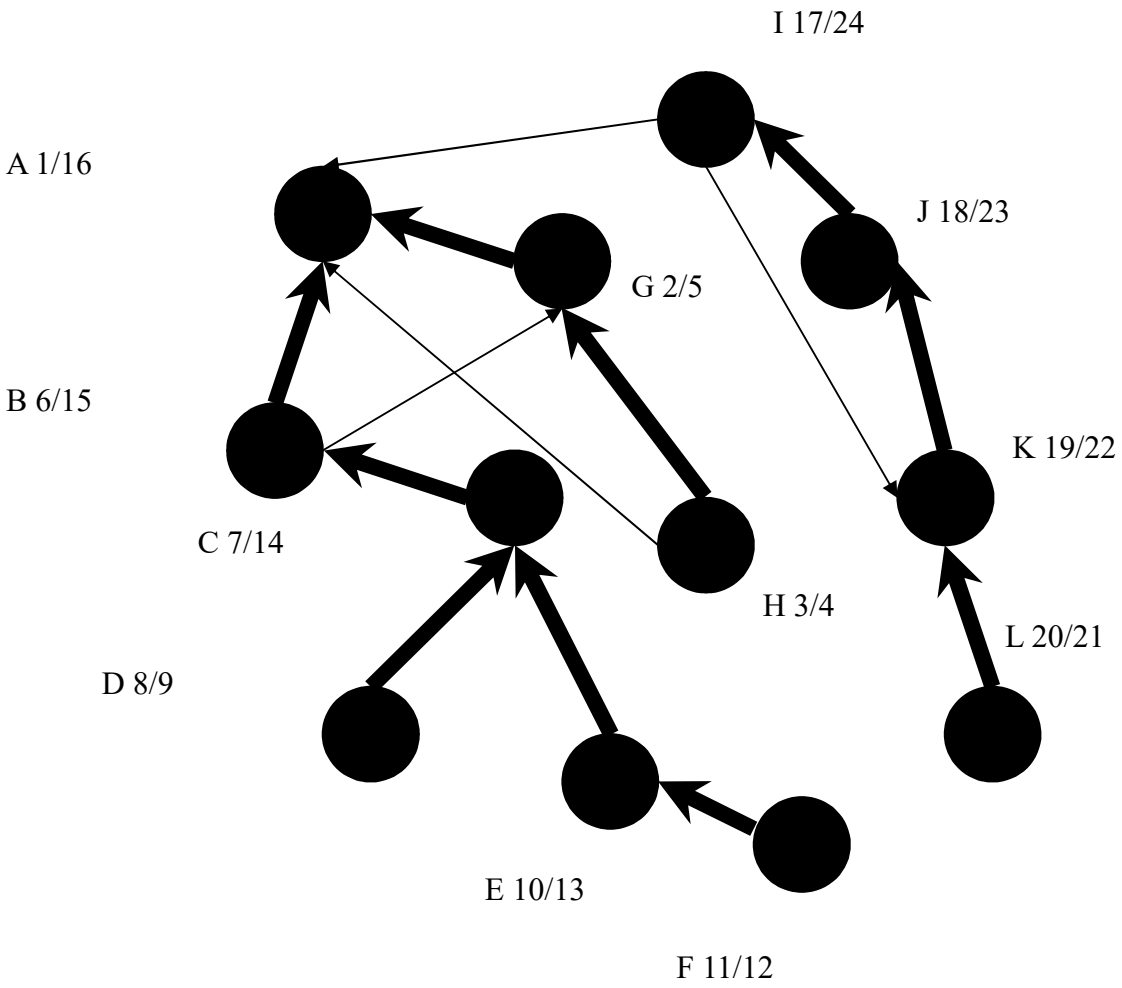

I
A 1/
J
G 2/
B
K
C
H 3/
L
D
E
F


I
A 1/
J
G 2/5
B 6/
K
C
H 3/4
L
D
E
F


I
A 1/
J
G 2/5
B 6/
K
C 7/
H 3/4
L
D
E
F


I 17/
A 1/16
J 18/
G 2/5
B 6/15
K
C 7/14
H 3/4
L
D 8/9
E 10/13
F 11/12

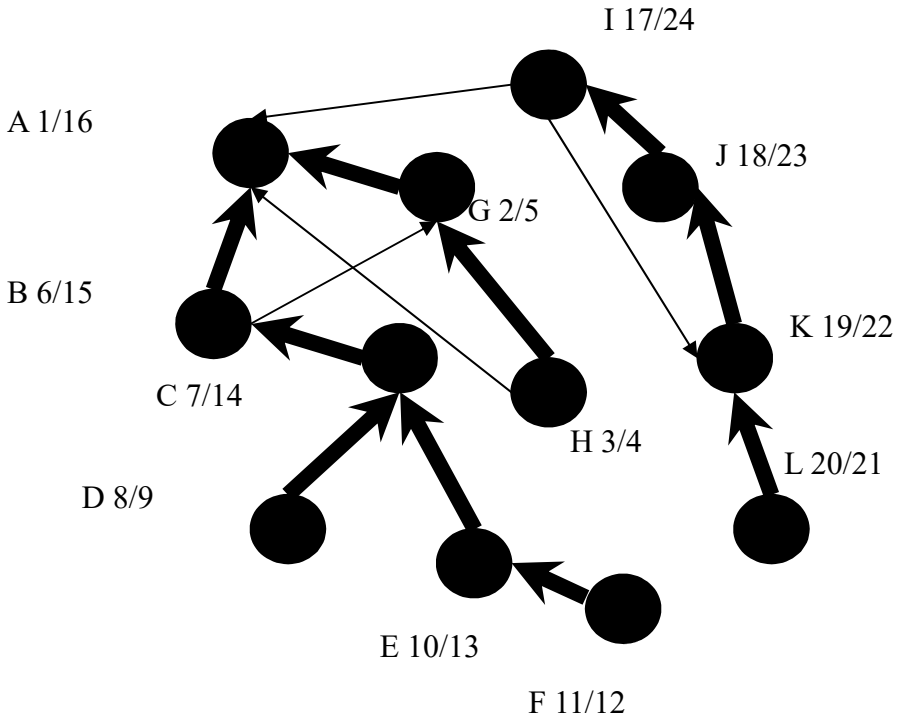

I 17/
A 1/16
J 18/23
G 2/5
B 6/15
K 19/22
C 7/14
H 3/4
L 20/21
D 8/9
E 10/13
F 11/12


I 17/24
A 1/16
J 18/23
G 2/5
B 6/15
K 19/22
C 7/14
H 3/4
L 20/21
D 8/9
E 10/13
F 11/12

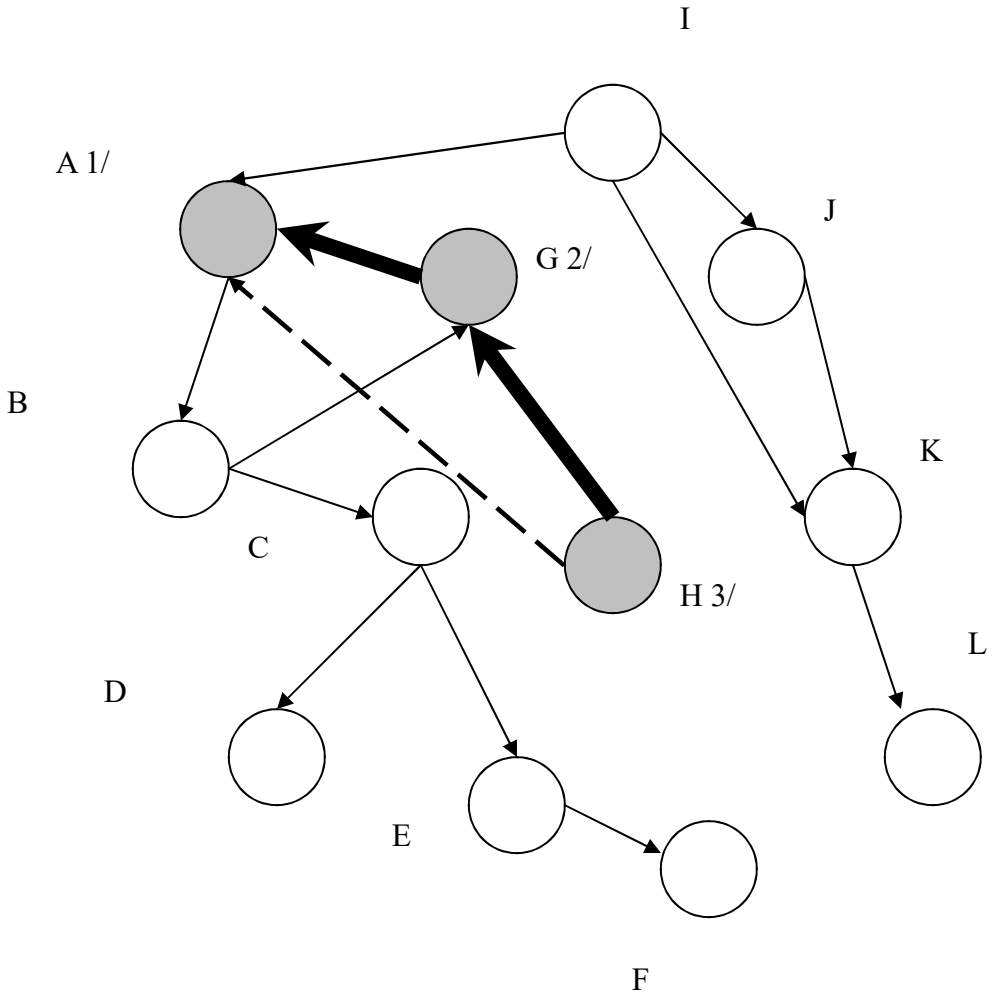
```
| | | ( a este gri debut/finis= 1 /
| | | | | | ( g este gri debut/finis= 2 /
| | | | | | | | | ( h este gri debut/finis= 3 /
| | | | | | | | | ) h este negru debut/finis= 3 / 4
| | | | | | ) g este negru debut/finis= 2 / 5
| | | | | | ( b este gri debut/finis= 6 /
| | | | | | | | | ( c este gri debut/finis= 7 /
| | | | | | | | | | | | ( d este gri debut/finis= 8 /
| | | | | | | | | | | | ) d este negru debut/finis= 8 / 9
| | | | | | | | | | | | ( e este gri debut/finis= 10 /
| | | | | | | | | | | | | | | ( f este gri debut/finis= 11 /
| | | | | | | | | | | | | | | ) f este negru debut/finis= 11 / 12
| | | | | | | | | | | | ) e este negru debut/finis= 10 / 13
| | | | | | | | | ) c este negru debut/finis= 7 / 14
| | | | | | ) b este negru debut/finis= 6 / 15
| | | ) a este negru debut/finis= 1 / 16
| | | ( i este gri debut/finis= 17 /
| | | | | | ( j este gri debut/finis= 18 /
| | | | | | | | | ( k este gri debut/finis= 19 /
| | | | | | | | | | | | ( l este gri debut/finis= 20 /
| | | | | | | | | | | | ) l este negru debut/finis= 20 / 21
| | | | | | | | | ) k este negru debut/finis= 19 / 22
| | | | | | ) j este negru debut/finis= 18 / 23
| | | ) i este negru debut/finis= 17 / 24
```

# Pădurea de arbori de parcurgere în adâncime

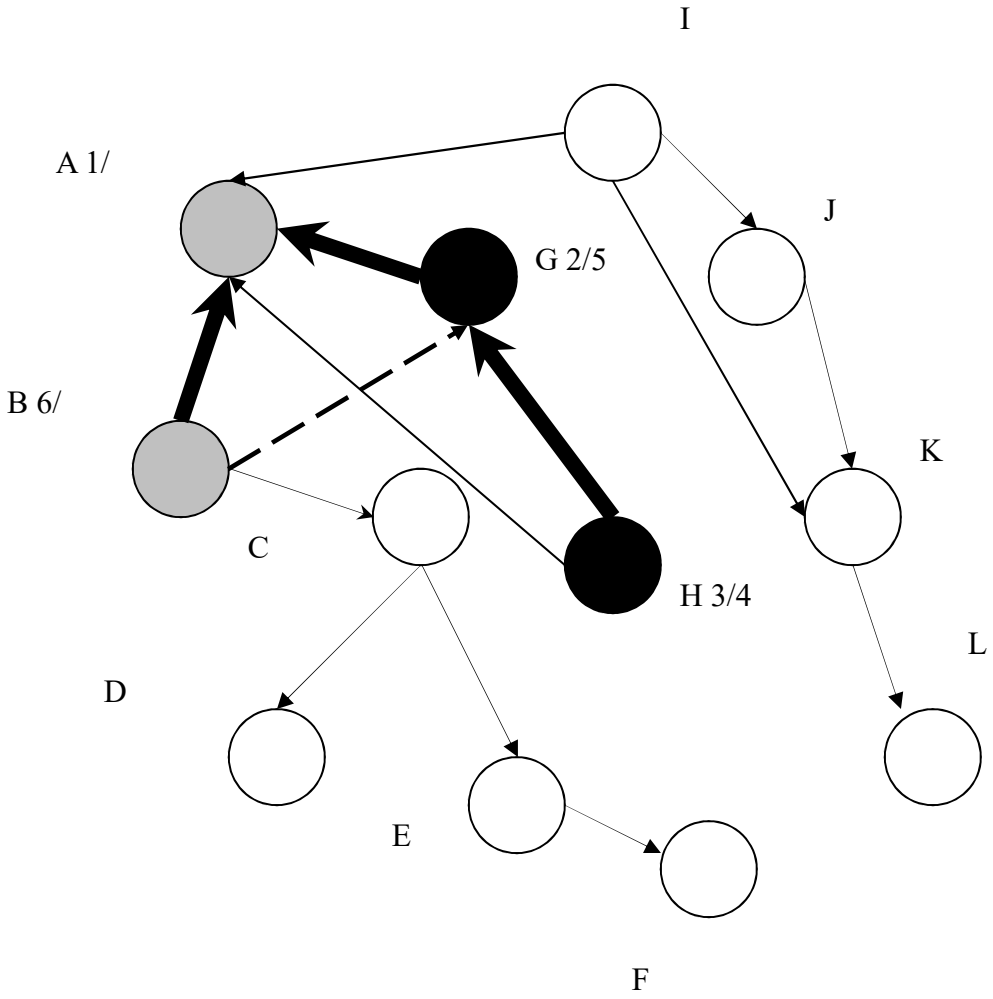
# Arce ”în arbore”



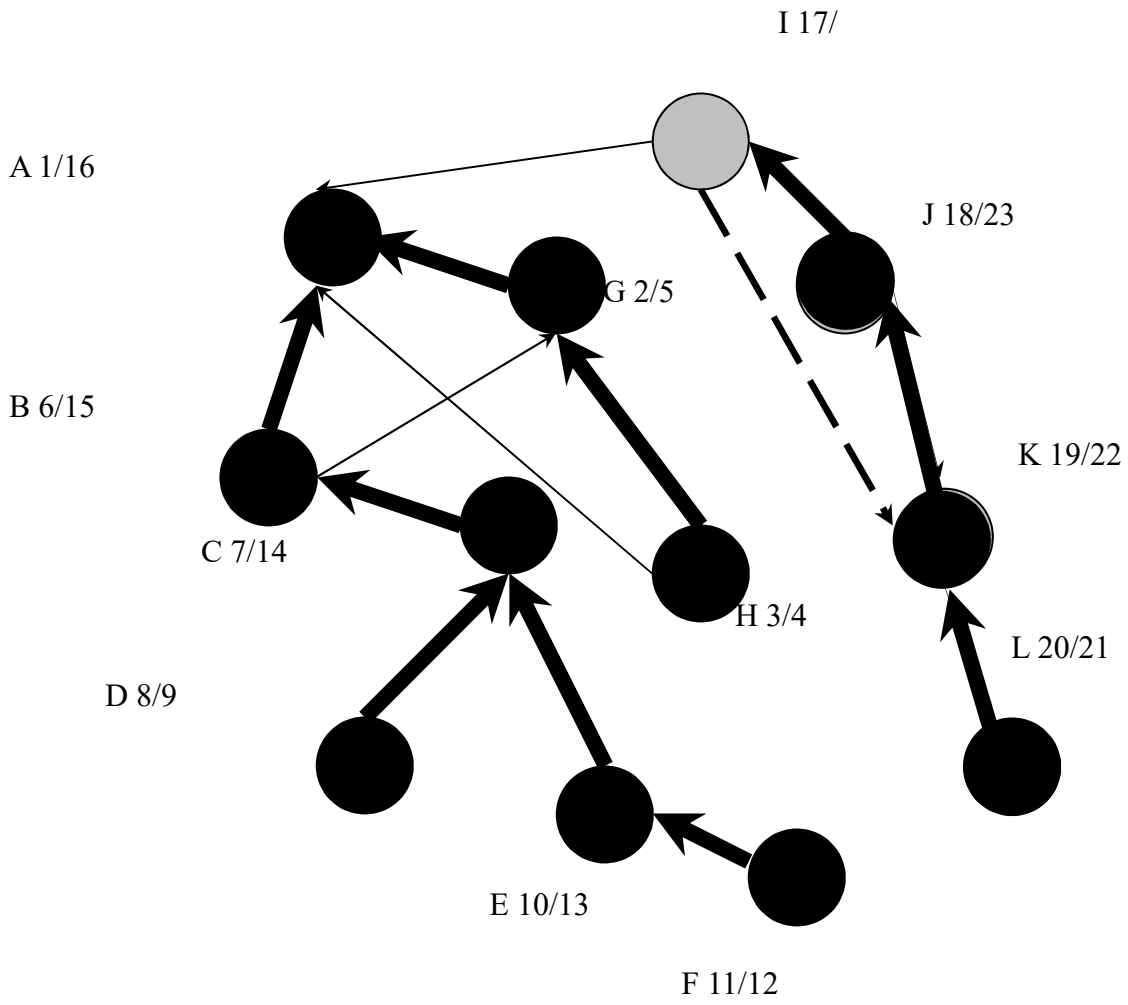
# Arc înapoi



# Arc de traversare



# Arc înainte



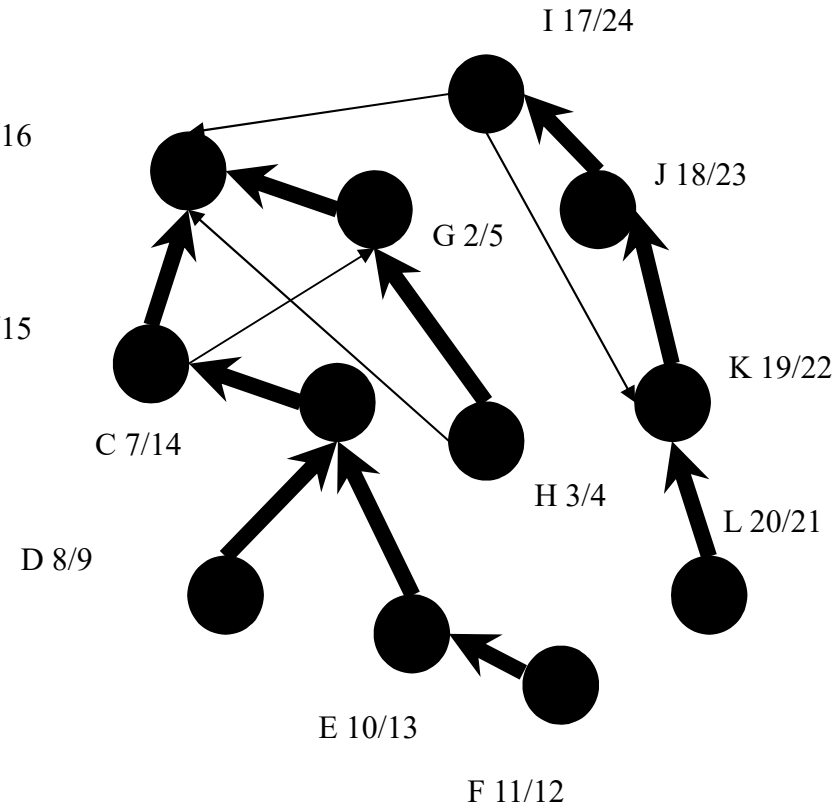
# Tipuri de arce

```
if culoare(j)=='alb':
   pune_tata(j,i)
   parc_ad_rec(g,j)
 elif culoare(j)=='gri':
  print 'arc inapoi',g[i][0],g[j][0],' - ciclu'
 elif culoare(j)=='negru' and g[j][3] < g[i][3]:
  print 'arc de traversare',g[i][0],g[j][0]
 elif culoare(j)=='negru' and g[j][3]  >  g[i][3]:
  print 'arc inainte',g[i][0],g[j][0]
```

# Tipuri de arce

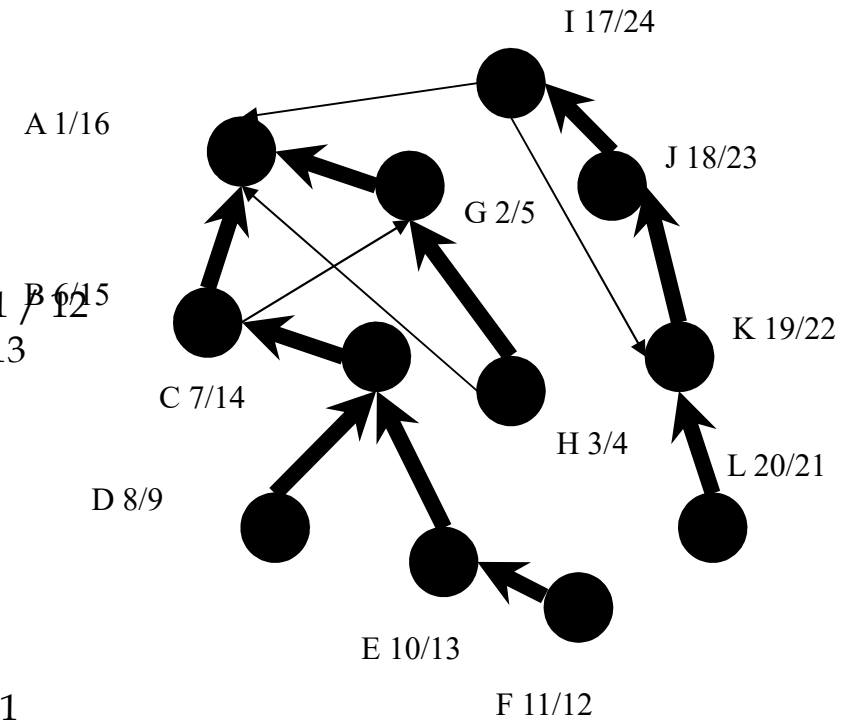
```
if culoare(j)=='alb':
   pune_tata(j,i)
   parc_ad_rec(g,j)
 elif culoare(j)=='gri':
  print 'arc inapoi',g[i][0],g[j][0],' - ciclu'
 elif culoare(j)=='negru' and g[j][3] < g[i][3]:
  print 'arc de traversare',g[i][0],g[j][0]
 elif culoare(j)=='negru' and g[j][3]  >  g[i][3]:
  print 'arc inainte',g[i][0],g[j][0]
```

| | | ( a este gri debut/finis= 1 /
| | | | | | ( g este gri debut/finis= 2 /
| | | | | | | | | ( h este gri debut/finis= 3 /
arc inapoi h a - ciclu
| | | | | | | | | ) h este negru debut/finis= 3 / 4
| | | | | | ) g este negru debut/finis= 2 / 5
| | | | | | ( b este gri debut/finis= 6 /
| | | | | | | | | ( c este gri debut/finis= 7 /
| | | | | | | | | | | | ( d este gri debut/finis= 8 /
| | | | | | | | | | | | ) d este negru debut/finis= 8 / 9
| | | | | | | | | | | | ( e este gri debut/finis= 10 /
| | | | | | | | | | | | | | ( f este gri debut/finis= 11 /
| | | | | | | | | | | | | | ) f este negru debut/finis= 11 / 12
| | | | | | | | | | | | ) e este negru debut/finis= 10 / 13
| | | | | | | | | ) c este negru debut/finis= 7 / 14
arc de traversare b g
| | | | | | ) b este negru debut/finis= 6 / 15
| | | ) a este negru debut/finis= 1 / 16
| | | ( i este gri debut/finis= 17 /
arc de traversare i a
| | | | | | ( j este gri debut/finis= 18 /
| | | | | | | | | ( k este gri debut/finis= 19 /
| | | | | | | | | | | | ( l este gri debut/finis= 20 /
| | | | | | | | | | | | ) l este negru debut/finis= 20 / 21
| | | | | | | | | ) k este negru debut/finis= 19 / 22
| | | | | | ) j este negru debut/finis= 18 / 23
arc inainte i k
| | | ) i este negru debut/finis= 17 / 24



# Structura de paranteze

înainte de  înainte de  înainte de  parc_ad(graf,12)
| | | (a este gri debut/finis= 1 /
| | | | | (g este gri debut/finis= 2 /
| | | | | | | | (h este gri debut/finis= 3 /
| | | | | | | | ) h este negru debut/finis= 3 / 4
| | | | | ) g este negru debut/finis= 2 / 5
| | | | | (b este gri debut/finis= 6 /
| | | | | | | | (c este gri debut/finis= 7 /
| | | | | | | | | | (d este gri debut/finis= 8 /
| | | | | | | | | | ) d este negru debut/finis= 8 / 9
| | | | | | | | | | (e este gri debut/finis= 10 /
| | | | | | | | | | | | (f este gri debut/finis= 11 /
| | | | | | | | | | | | ) f este negru debut/finis= 11 / 12
| | | | | | | | | | ) e este negru debut/finis= 10 / 13
| | | | | | | | ) c este negru debut/finis= 7 / 14
| | | | | ) b este negru debut/finis= 6 / 15
| | | ) a este negru debut/finis= 1 / 16
| | | (i este gri debut/finis= 17 /
| | | | | (j este gri debut/finis= 18 /
| | | | | | | | (k este gri debut/finis= 19 /
| | | | | | | | | | (l este gri debut/finis= 20 /
| | | | | | | | | | ) l este negru debut/finis= 20 / 21
| | | | | | | | ) k este negru debut/finis= 19 / 22
| | | | | ) j este negru debut/finis= 18 / 23
| | | ) i este negru debut/finis= 17 / 24



# Proprietăți ale parcurgerii în adâncime

- Structura de paranteze - teorema parantezelor (cazuri posibile)
  + corolar:

$$debut[u] < debut[v] < finiș[v] < finiș[u]$$

- Teorema căii albe
- Adaptare la grafuri neorientate

# Aplicații ale parcurgerii în adâncime

- Sortarea topologică
- Componente conexe
- Componente tare conexe
- Componente biconectate
- Puncte de articulații și punți
- .....?

# E necesar să:

- Concepem algoritmul plecând sistematic de la ceva cunoscut
- Demonstrăm că e corect
- Evaluăm complexitatea

# Sortarea topologică - DAG

L înainte de K,

J înainte de I,

I înainte de A,

K înainte de J, K înainte de I,

H înainte de G,

G înainte de A, G înainte de B,

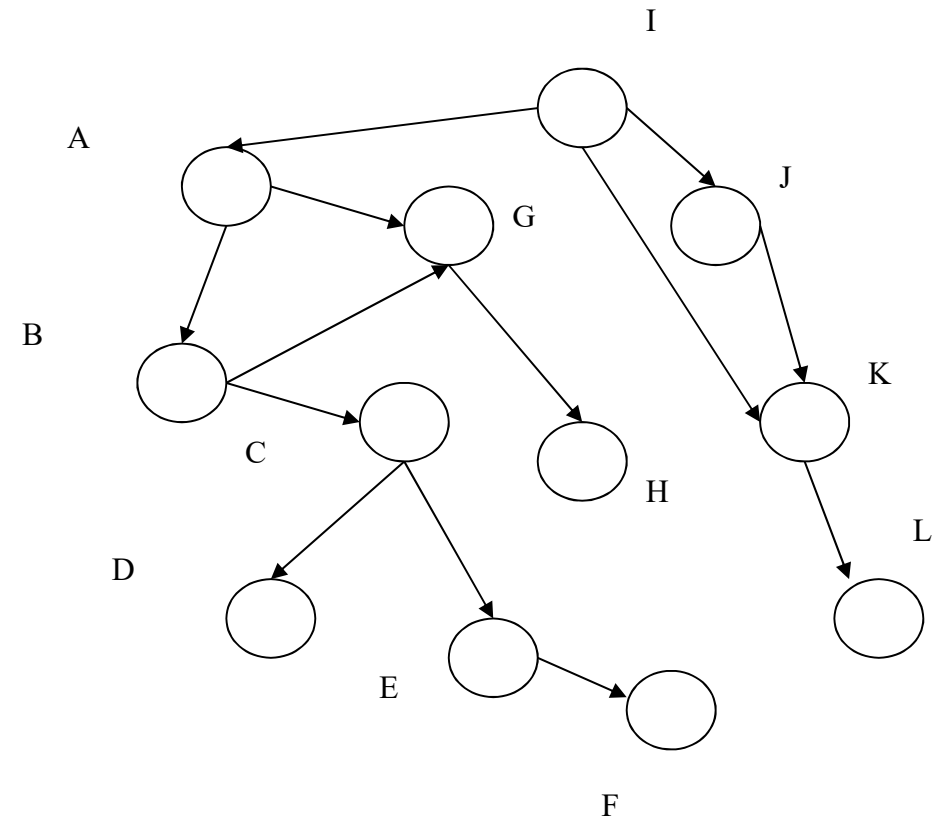
B înainte de A,

D înainte de C,

E înainte de C,

C înainte de B

F înainte de E

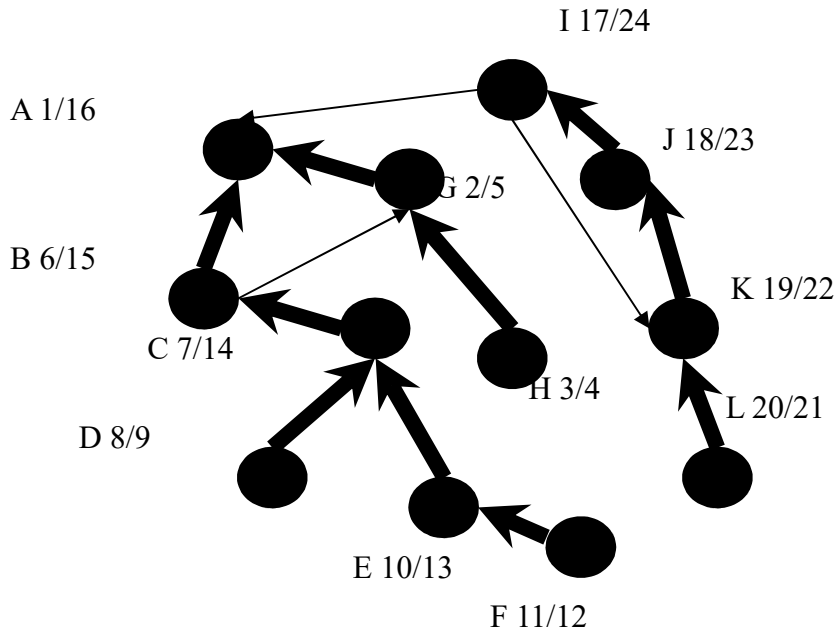


```
l=[]
def parc_ad_rec(g,i):
  global l
  graf[i][3]=timp
  timp+=1
  graf[i][1]='gri'
  for j in adiacente (i):
   if culoare(j)=='alb':
    pune_tata(j,i)
    parc_ad_rec(g,j)
  graf[i][4]=timp
  timp+=1
  graf[i][1]='negru'
  l=[i]+l
```

# Sortarea topologică

# Sortarea topologică

**i, j, k, l, a, b, c, e, f, d, g, h**



# Demonstrarea corectitudinii sortării topologice

- Lemă – arcele minim caracterizează DAG-urile
- Teoremă – algoritmul e corect