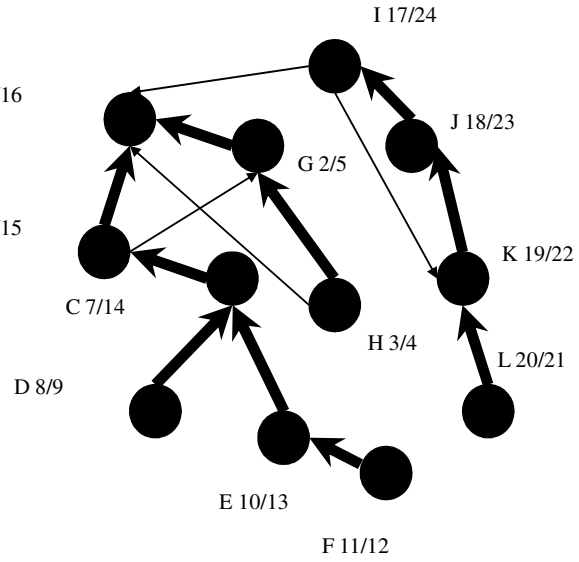
Algoritmi pe grafuri - 2

Ștefan Trăușan-Matu

Tipuri de arce

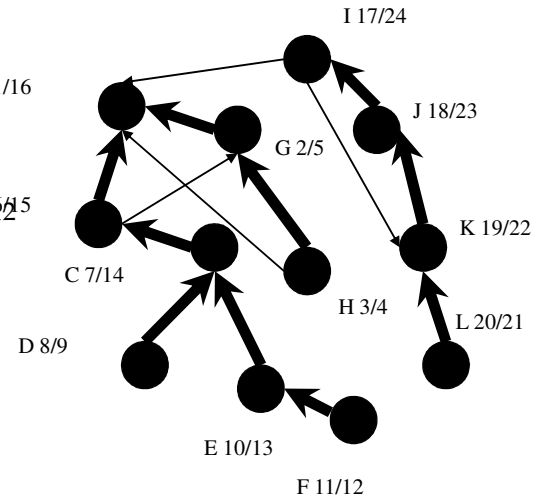
```
if culoare(j)=='alb':
   pune_tata(j,i)
   parc_ad_rec(g,j)
 elif culoare(j)=='gri':
  print 'arc inapoi',g[i][0],g[j][0],' - ciclu'
 elif culoare(j)=='negru' and g[j][3] < g[i][3]:
  print 'arc de traversare',g[i][0],g[j][0]
 elif culoare(j)=='negru' and g[j][3]  >  g[i][3]:
  print 'arc inainte',g[i][0],g[j][0]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

| | | (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,

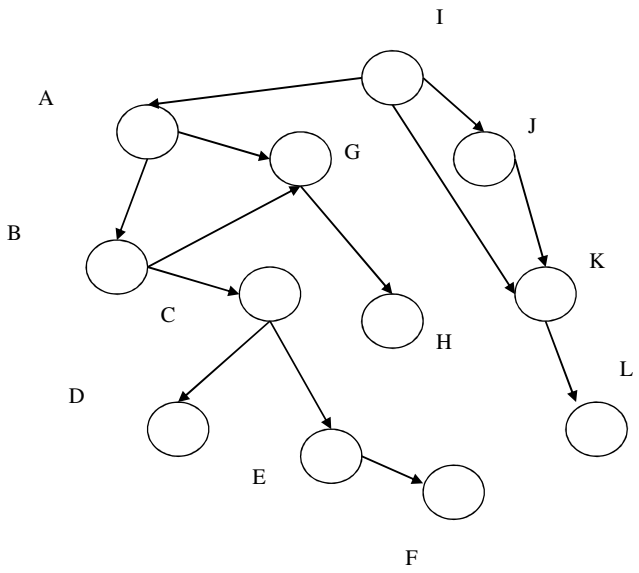
B înainte de A,

D înainte de C,

E înainte de C,

C înainte de B

F înainte de E

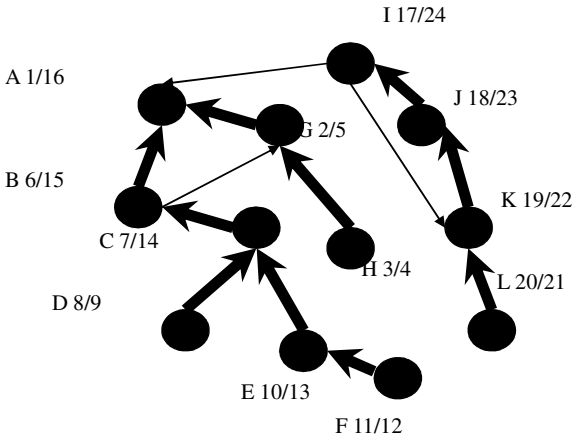


Sortarea topologică

```
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

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