Parcours de graphes

Algorithmique – L3

François Laroussinie

novembre 2011

Parcours de graphes

Procédure `Parcours` G
$//G = S, A$
begin
  pour chaque $x \in S$ **faire** $Couleur[x] := blanc$
  Choisir $s \in S$
  $Couleur[s] := gris$
  répéter
    Choisir x tq $Couleur[x] = gris$
    pour chaque $x, y \in A$ **faire**
      si $Couleur[y] = blanc$ **alors** $Couleur[y] := gris$
    $Couleur[x] := noir$
  jusqu'à $\forall x. Couleur[x] \neq gris$;
end

Parcours

Il existe un sommet s dont tous les successeurs n'ont pas encore été découverts.

- On choisit un sommet s dont tous les successeurs n'ont pas encore été découverts.
- On explore les transitions issues de s et on marque les successeurs de s pas encore découverts.

Parcours

Idée :

- On choisit un sommet s et on ajoute tous les successeurs qui ont pas encore été découverts.
- On explore les transitions issues de s et on cherche parmi les successeurs de s pas encore découverts.

Trois états pour les sommets :

- non découvert,
- découvert mais certains successeurs restent non découverts, ou
- découvert ainsi que ses successeurs.

→ trois couleurs blanc/gris/noir .

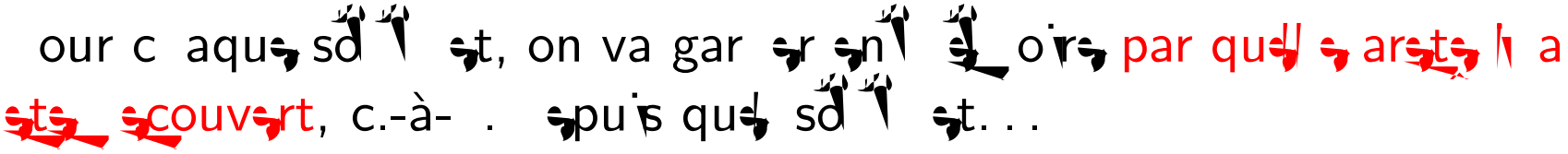
Plan

Plan

Parcours en largeur

on considère un graphe non orienté.

Arborescence de parcours



Arborescence de parcours

our c aqu sd t, on va gar r n o r par qu ar t a t couv rt, c.-à- . pu s qu sd t. . .

$S \to S \cup \{n\}$

$\Pi[n] = u$ ssi u a t couv rt pu s v.

Arborescence de parcours

our c aqu sd t, on va gar r n o r par qu ar t a t couv rt, c.-à- . pu s qu sd t. . .

$\Pi : S \to S \cup \{\text{nil}\}$

$u \underset{\Pi}{=} v$ ssi u a t couv rt pu s v.

n construit onc un arbr $G_\Pi = (S, A_\Pi)$ rac n s.

$$A_\Pi \overset{\text{def}}{=} \{ (\underset{\Pi}{v}, v) \mid \underset{\Pi}{v} \neq \text{nil} \}$$

Propriété du parcours en largeur

L'algorithme va parcourir les sommets accessibles depuis s en commençant par ceux situés à la distance 1, puis ceux situés à la distance 2, *etc.*

Propriété du parcours en largeur

L'algorithme va parcourir les sommets accessibles depuis s en commençant par ceux situés à la distance 1, puis ceux situés à la distance 2, *etc.*

De plus, l'algorithme va

1. calculer la distance minimale de chacun des sommets à l'origine s, et
2. les chemins dans G_Π seront les plus courts chemins dans G.

Parcours en largeur de $G = (S, A)$ non-orienté

Procédure PL G,s
pour chaque $x \in S \backslash \{s\}$ **faire**
  Couleur[x] $\leftarrow$ blanc ; $\Pi[x] \leftarrow$ nil ; Dist[x] $\leftarrow \infty$;
Couleur[s] $\leftarrow$ gris ; $\Pi[s] \leftarrow$ nil ; Dist[s] $\leftarrow 0$;
$F \leftarrow$ File vide
Ajouter F, s
tant que $F \neq \emptyset$ **faire**
  $x \leftarrow$ Extraire tête F ;
  pour chaque $x, y \in A$ **faire**
    si *Couleur*[y] $=$ *blanc* **alors**
      Couleur[y] $\leftarrow$ gris ;
      Dist[y] $\leftarrow$ Dist[x] $+ 1$;
      $\Pi[y] \leftarrow x$;
      Ajouter F, y ;
  Couleur[x] $\leftarrow$ noir

Parcours en largeur

Le sens du codage blanc/gris/noir est :

- blanc : les sommets pas encore découverts et à l'initialisation, sauf s .
- gris : les sommets déjà découverts et dont les successeurs n'ont pas encore été tous découverts ;
- noir : les sommets découverts dont tous les successeurs ont aussi été découverts.

Terminaison et complexité

out ajout u ans F...

- st con itionn par « Cou ur[u] = b anc », t
- st su iv i par un co orag n gris u.
- n so t n st co ori n b anc qui à l i nit ia isation.

Terminaison et complexité

out ajout u ans F...

- st con itionn par « Cou ur[u] = b anc », t
- st su iv i par un co ori ag n gri u.
- n so t n st co ori n b anc qui à l i niti a isation.

$\Rightarrow$ u n put tr ajout t xtrait qui au p us un ro is.
$\Rightarrow$ l a gorit t t r in

Terminaison et complexité

out ajout s u ans F...

- st con itionn s par « Cou sur[u] = b anc », t
- st su iv i par un co ori ag s n gri s s u.
- n so t n st co ori s n b anc qui à l n iti a isat on.

$\Rightarrow$ u n s p ut tr s ajout s t xtra it qui au p us un ois.
$\Rightarrow$l a gor it s t n
Et sa l ist l a jac nc n st parcouru qui au p us un ois.

Terminaison et complexité

out ajout s u ans F...

- st con itionn s par « Cou ur[u] = b anc », t
- st su iv i par un co orisag s n gris s u.
- n sd t n st co oris s n b anc qui à l i niti a isation.

$\Rightarrow$ u n s put str s ajout s t xtrait qui au p us un s ois.
$\Rightarrow$ l a gorit s t n
Et sa ist s a jac nc s n st parcouru s qui au p us un s ois.
$\Rightarrow$ Cd p xit s tota s :

- s l a boud s princ pa s $O\ |A|$,
- s l i niti a isation $O\ |S|$.

$\Rightarrow$ Cd p xit s n $O\ |S| + |A|$ ou $O\ |G|$

Correction du parcours en largeur

Théorème Soient $G = (S, A)$ un graphe non-orienté et $s \in S$ un sommet. L'algorithme PL (G, s) :

1. recouvre tous les sommets atteignables depuis s et uniquement eux ;
2. termine avec $Dist[v] = \delta(s, v)$ pour tout $v \in S$;
3. construit le tableau Π de sorte que pour tout sommet $u \neq s$ atteignable depuis s, l'arête $(\Pi[u], u)$ appartient à un plus court chemin dans G de s à u dont c'est la dernière transition.

Preuve : $\delta(s, v)$ est la longueur d'un plus court chemin entre s et v

Propriétés – distances

Définition $G = (S, A)$, $u, v \in S$

$$\delta(u, v) = \begin{cases} \min\{p \mid u \to^p v\} & \text{si } u \to^* v \\ \infty & \text{sinon} \end{cases}$$

Propriété 13 Soient $u, v \in S$ tels que $(u, v) \in A$, alors on a :

$$\delta(s, v) \leq \delta(s, u) + 1$$

Propriétés – parcours en largeur

Propriété 14 A tout moment de l'algorithme, on a pour tout sommet u : $Dist[u] \geq \delta(s, u)$.

Propriété 15 Lors de l'exécution de PL sur $G = (S, A)$ depuis s, à **chaque étape de l'algorithme**, si le contenu F de la file est $[v_1, v_2, \ldots, v_k]$ où v_1 désigne le sommet le plus ancien et v_k le plus récent, alors on a :

- $Dist[v_i] \leq Dist[v_{i+1}]$ pour $i = 1, \ldots, k-1$
- $Dist[v_k] \leq Dist[v_1] + 1$

Plan

Idée générale

On considère des graphes orientés.

Ici on choisit de dessiner tout gris recouvert de plus récent.

Au lieu d'une de, on utilise une plus.

Idée générale

On considère des graphes orientés.

Ici on choisit le sommet gris recouvert le plus récemment.

Au lieu d'une file, on utilise une pile.

On enroule donc un chemin gris le plus loin possible.

Idée générale

On considère les graphes orientés.

Ici on choisit le sommet est gris recouvert le plus récent.

Au lieu d'une file, on utilise une pile.

On enroule donc un chemin gris le plus loin possible.

Les trois couleurs signifient :

- blanc : sommet est non encore découvert ;
- gris : sommet est découvert mais ont certains **descendants** n'ont pas encore été découverts ;
- noir : sommet est découvert ainsi que tous ses descendants.

Algorithme - 1

Cd l pour l parcours n l arg ur, on construit G_Π... l ais ici c l sera un **forêt**.

Algorithme - 1

Cet algorithme, pour le parcours en largeur, on construit G_Π... ici ce sera une **forêt**.

On va associer à tout sommet $u \in S$:

- une date de coloriage en gris $[u]$ et
- une date de coloriage en noir $f[u]$

une date = un entier entre 1 et $2 \cdot |S|$

Algorithme - 1

Cet algorithme, pour le parcours en largeur, on construit G_Π... Il a ici ce sera une **forêt**.

On va associer à tout sommet $u \in S$:

- une date de coloriage en gris $[u]$, et
- une date de coloriage en noir $f[u]$

une date = un entier entre 1 et $2 \cdot |S|$

$[u] < f[u]$

Et...

- avant la date $[u]$, u est en blanc ;
- entre $[u]$ et $f[u]$, u est en gris ;
- après la date $f[u]$, u est en noir.

Parcours en profondeur (init)

```
Procédure PP G
//G = S, A
begin
    pour chaque x ∈ S faire
        Couleur[x] := blanc ;
        π[x] := nil ;
    temps := 0 ;
    pour chaque x ∈ S faire
        si Couleur[x] = blanc alors
            PP-Visiter G, x ;
end
```

Parcours en profondeur

Procédure `PP-Visiter` G, s

$Couleur[s] \leftarrow gris$;

$temps \leftarrow temps + 1$;

$d[s] \leftarrow temps$;

pour chaque $s, u \in A$ **faire**

  si $Couleur[u] = blanc$ **alors**

    $\pi[u] \leftarrow s$;

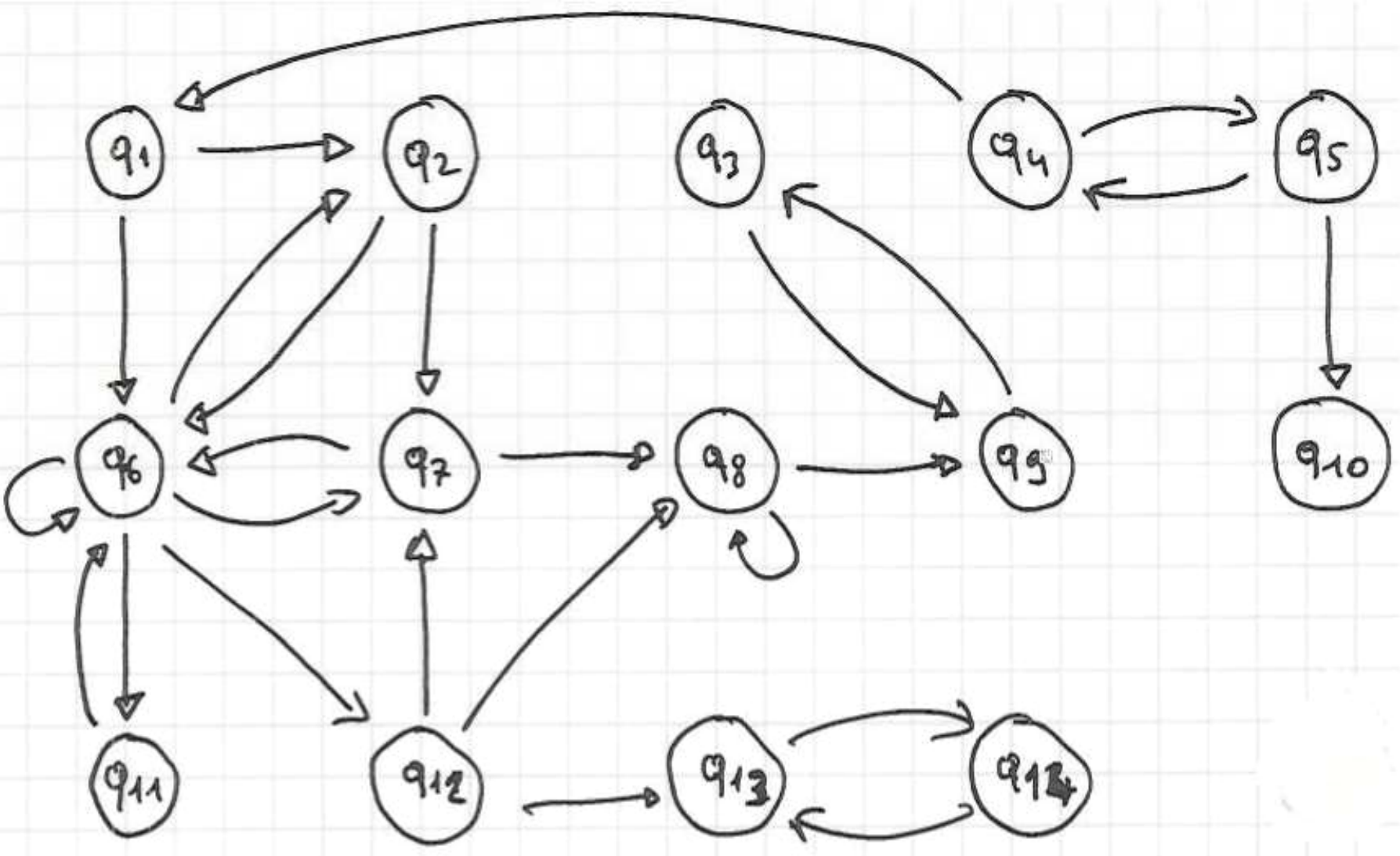
    `PP-Visiter` G, u;

$Couleur[s] \leftarrow noir$;

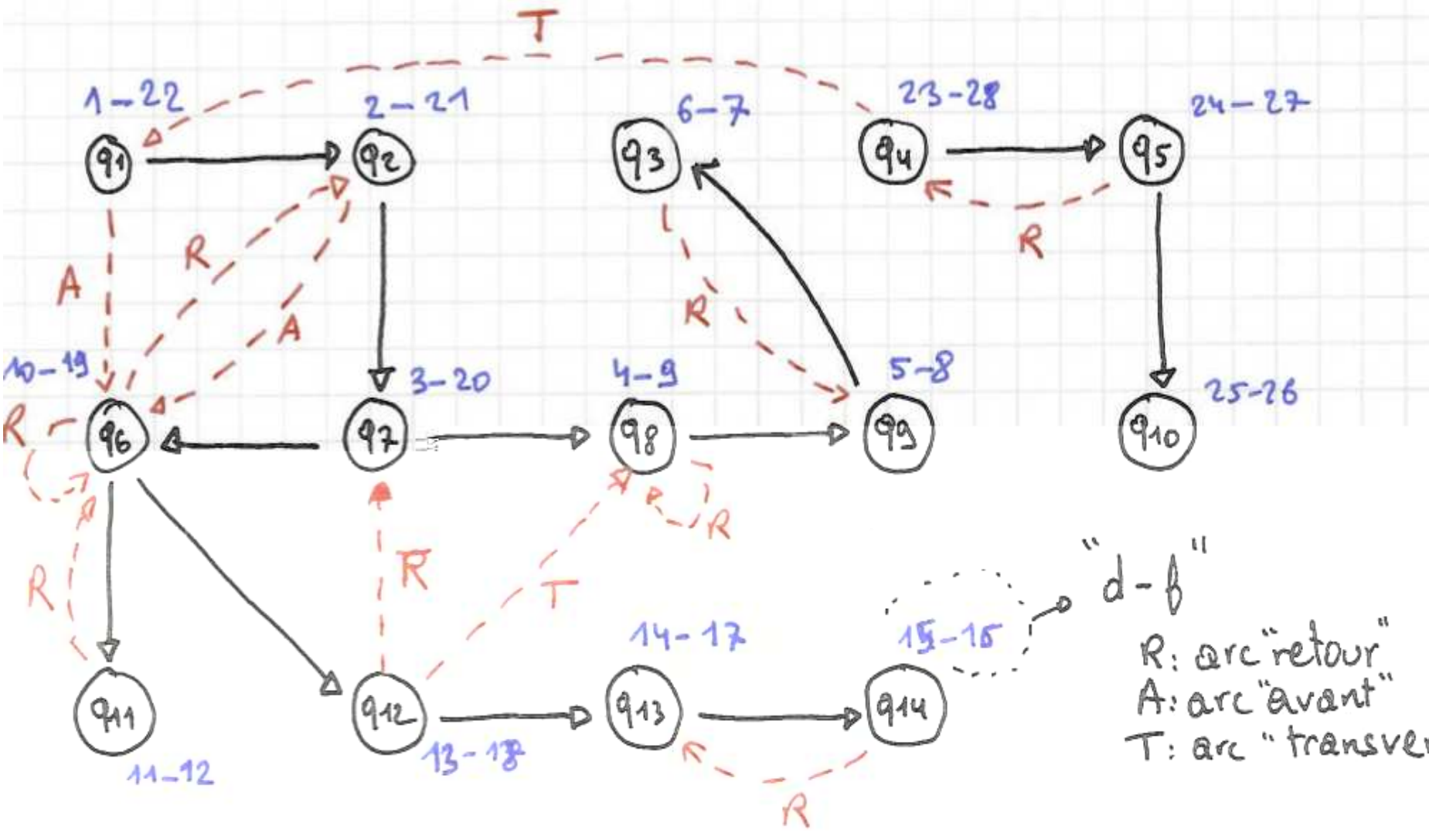
$temps \leftarrow temps + 1$;

$f[s] \leftarrow temps$;

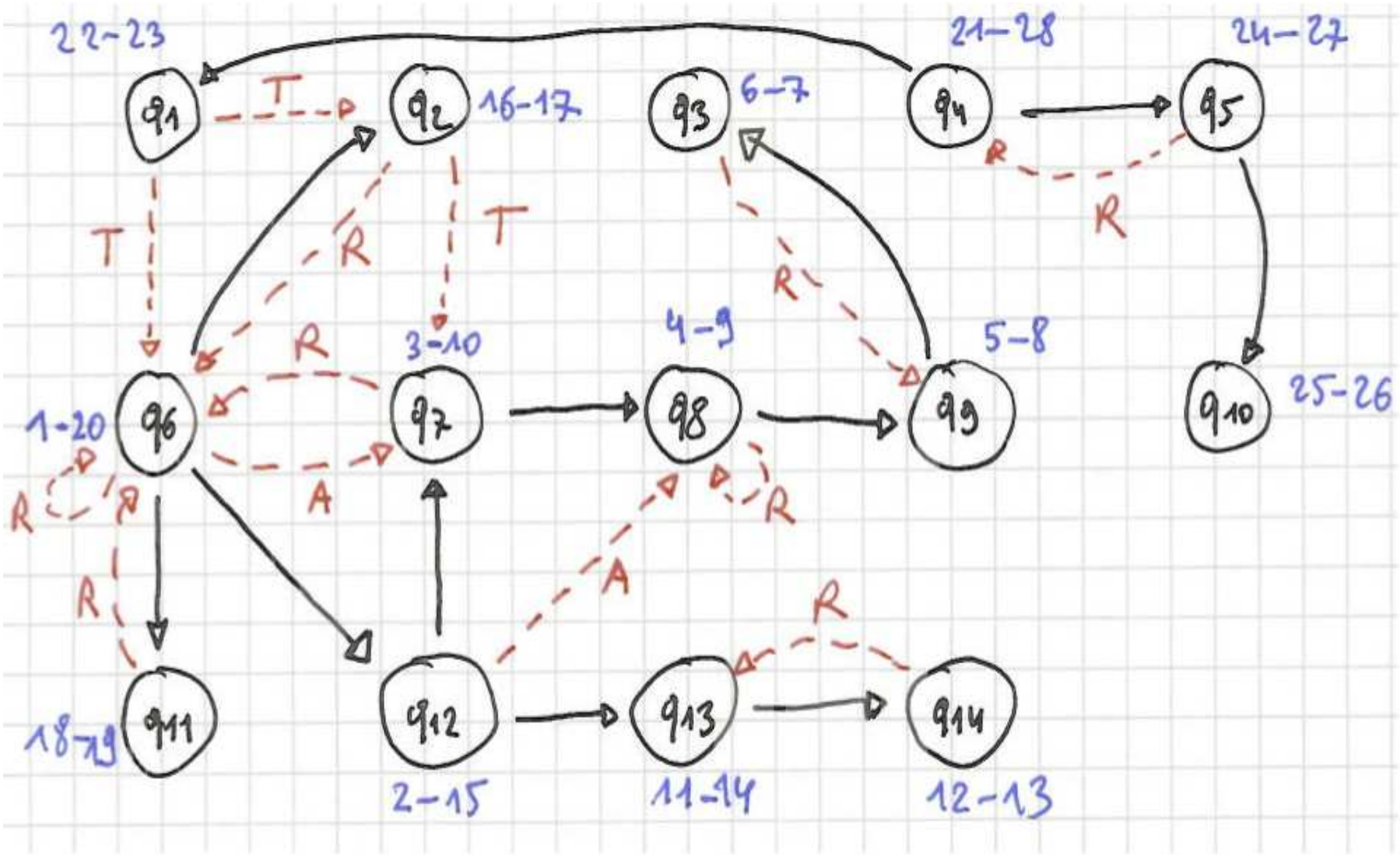
Exemple



Exemple


T
1-22
2-21
6-7
23-28
24-27
q1
q2
q3
q4
q5
A
R
A
R
R
R
10-19
3-20
4-9
5-8
25-26
R
q6
q7
q8
q9
q10
R
R
R
T
14-17
15-15
"d-f"
q11
q12
q13
q14
11-12
13-18
R
R: arc "retour"
A: arc "avant"
T: arc "transver

Exemple


22-23
21-28
24-27
q1
T
q2
16-17
q3
6-7
q4
q5
T
R
T
R
R
R
4-9
3-10
5-8
1-20
q6
R
q7
q8
q9
q10
25-26
R
R
A
R
R
A
R
18-19
q11
q12
q13
q14
2-15
11-14
12-13

Complexité

L'initialisation est en temps $O(|S|)$.

La procédure PP-Visiter est appelée exactement une fois sur chaque sommet.

La condition d'existence des boucles « **Pour chaque** $s, u \ldots$ » dans tous les appels PP-Visiter est donc en $O(|A|)$.

On a donc un algorithme linéaire, *i.e.* en $O(|S| + |A|)$.

Classification des arcs. . .

out arc v, w A st soit un arc G_Π, soit. . .

- un arc G_Π *i.e.* $\Pi[w] = v$;
- un arc « retour » : v st un scn ant w ans G_Π * ;
- un arcs « avant » : w st un scn ant v ans G_Π ais $\Pi[w] \neq v$ * ;
- un arc « transvrs » tous les autres cas.

* lors lexn v, w ans `PP-Visiter`.

Propriétés du Parc. en Profondeur - 1

Propriété 16 our tout sd Γ t u t v, on a :

- soit l s intervalles $[\,[u]_{\Gamma}, [u]]$ t $[\,[v]_{\Gamma}, [v]]$ sont sjoints ;
- soit $[\,[u]_{\Gamma}, [u]]$ st contnu ans $[\,[v]_{\Gamma}, [v]]$;
- soit $[\,[v]_{\Gamma}, [v]]$ st contnu ans $[\,[u]_{\Gamma}, [u]]$.

Propriétés du Parc. en Profondeur - 1

Propriété 16 our tout sd $\mathcal{T}$ et u et v, on a :

- soit les intervalles $[\,[u]_{\mathcal{T}}, [u]]$ et $[\,[v]_{\mathcal{T}}, [v]]$ sont disjoints ;
-

Propriétés du Parc. en Profondeur - 1

Propriété 16 our tout sd $\mathcal{T}$ t u t v, on a :

- soit l s intervalles $[t_d[u], t_f[u]]$ t $[t_d[v], t_f[v]]$ sont disjoints ;
- soit $[t_d[u], t_f[u]]$ st contenu ans $[t_d[v], t_f[v]]$;
- soit $[t_d[v], t_f[v]]$ st contenu ans $[t_d[u], t_f[u]]$.

S upposons $[t_d[u], t_f[u]] < [t_d[v], t_f[v]]$.

- $[t_d[v], t_f[v]] < [t_d[u], t_f[u]]$:
  - En $[t_d[v]$, u st gris PP-Visiter G, u n st pas n

Propriétés du Parc. en Profondeur - 1

Propriété 16 our tout sd $\mathcal{T}$ st u t v, on a :

- soit l s intrvals $[\,[u]\,;[u]\,]$ t $[\,[v]\,;[v]\,]$ sont sjoints ;
- soit $[\,[u]\,;[u]\,]$ st contnu ans $[\,[v]\,;[v]\,]$;
- soit $[\,[v]\,;[v]\,]$ st contnu ans $[\,[u]\,;[u]\,]$.

S upposons $[u] < [v]$.

- $[v] < [u]$:
  - En $[v]$, u st gris PP-Visiter G, u n st pas ni.
  - on va appler PP-Visiter G, v' pour tout $v, v' \in A$ t.q. Coulur$[v'] =$ blanc.

Propriétés du Parc. en Profondeur - 1

Propriété 16 our tout sd T et u et v, on a :

- soit les intervalles $[\,[u]_T, [u]_T]$ et $[\,[v]_T, [v]_T]$ sont disjoints ;
- soit $[\,[u]_T, [u]_T]$ est contenu dans $[\,[v]_T, [v]_T]$;
- soit $[\,[v]_T, [v]_T]$ est contenu dans $[\,[u]_T, [u]_T]$.

Supposons $[\,[u]_T, [u]_T] < [\,[v]_T, [v]_T]$.

- $[\,[v]_T, [v]_T] < [\,[u]_T, [u]_T]$:
  - En $[\,[v]_T$, u est gris : PP-Visiter G, u n'est pas ni.
  - on va appeler PP-Visiter G, v' pour tout $v, v' \in A$ t.q. Couleur$[v'] =$ blanc.
  - puis colorier v en noir ... et affecter $[\,[v]_T$,

Propriétés du Parc. en Profondeur - 1

Propriété 16 our tout sd $\mathcal{T}$ st u t v, on a :

- soit l s intrvalls $[\,[u]_1, [u]\,]$ t $[\,[v]_1, [v]\,]$ sont sjoints ;
- soit $[\,[u]_1, [u]\,]$ st contnu ans $[\,[v]_1, [v]\,]$;
- soit $[\,[v]_1, [v]\,]$ st contnu ans $[\,[u]_1, [u]\,]$.

S upposons $[u] < [v]$.

- $[v] < [u]_1$:
  - En $[v]$, u st gris PP-Visiter G, u n st pas ni.
  - on va appler PP-Visiter G, v' pour tout $v, v' \in A$ t.q. Coulur$[v'] =$ blanc.
  - puis cloturer v n noir . . . t affecter $[v]_1$,
  - nir ls PP-Visiter G, u' s autrs scsn ants , u,

Propriétés du Parc. en Profondeur - 1

Propriété 16 our tout sd $\P$ st u st v, on a :

- soit l s intrvall s $[\ [u]\ ,[u]]$ t $[\ [v]\ ,[v]]$ sont sjoints ;
- soit $[\ [u]\ ,[u]]$ st contnu ans $[\ [v]\ ,[v]]$;
- soit $[\ [v]\ ,[v]]$ st contnu ans $[\ [u]\ ,[u]]$.

S upposons $[u] < [v]$.

- $[v] < [u]$:
  - En $[v]$, u st gris PP-Visiter G, u n st pas ni.
  - on va appler PP-Visiter G, v' pour tout $v, v' \in A$ t.q. Coulur$[v'] =$ blanc.
  - puis clorer v n noir . . . t arcter $[v]$,
  - nirls PP-Visiter G, u' s autrs scn ants u,
  - t n n arcter $[u]$.

Propriétés du Parc. en Profondeur - 1

Propriété 16 our tout sd $\P$ st u st v, on a :

- soit l s intrvall s $[\ [u]\ ,[u]]$ t $[\ [v]\ ,[v]]$ sont sjoints ;
- soit $[\ [u]\ ,[u]]$ st contnu ans $[\ [v]\ ,[v]]$;
- soit $[\ [v]\ ,[v]]$ st contnu ans $[\ [u]\ ,[u]]$.

S upposons $[u] < [v]$.

- $[v] < [u]$:
  - En $[v]$, u st gris PP-Visiter G, u n st pas n i.
  - on va appler PP-Visiter G, v' pour tout $v, v' \in A$ t.q. Coul ur$[v'] =$ blanc.
  - puis cder v n noir . . . t affecter $[v]$,
  - nir l s PP-Visiter G, u' s autres scn ants u,
  - t n n affecter $[u]$.
- $[v] > [u]$ alors $[u] < [u] < [v] < [v]$ . . .

Propriétés du Parc. en Profondeur - 2

Propriété 17 our tout sommet u, v, on a $u \rightarrow^*_{G_\Pi} v \Leftrightarrow [u] < [v] < f[v] < f[u]$.

Propriétés du Parc. en Profondeur - 2

Propriété 17 our tout sd ' ' ; t u, v, on a $u \rightarrow^*_{G_\Pi} v \Leftrightarrow$ $[u] < [v] < [v] < [u]$.

- 1 $\Rightarrow$ $u_0 = u \rightarrow_{G_\Pi} u_1 \rightarrow_{G_\Pi} u_2 \ldots \rightarrow_{G_\Pi} u_k = v$

Propriétés du Parc. en Profondeur - 2

Propriété 17 our tout sd ' ' st u, v, on a $u \to^*_{G_\Pi} v \Leftrightarrow$ $[u] < [v] < [v] < [u]$.

- 1 $\Rightarrow$ $u_0 = u \to_{G_\Pi} u_1 \to_{G_\Pi} u_2 \ldots \to_{G_\Pi} u_k = v$
  - u_i st couvrt puis u_{i-1} $[u_{i-1}] < [u_i] < [u_{i-1}]$.

Propriétés du Parc. en Profondeur - 2

Propriété 17 Pour tout sommet u, v, on a $u \rightarrow^*_{G_\Pi} v \Leftrightarrow [u] < [v] < f[v] < f[u]$.

- $1 \Rightarrow$ $u_0 = u \rightarrow_{G_\Pi} u_1 \rightarrow_{G_\Pi} u_2 \ldots \rightarrow_{G_\Pi} u_k = v$
  - u_i est découvert puis u_{i-1} : $[u_{i-1}] < [u_i] < f[u_i] < f[u_{i-1}]$.
  - par la prop. 16 : $[u_{i-1}] < [u_i] < f[u_i] < f[u_{i-1}]$, ...

Propriétés du Parc. en Profondeur - 2

$T \quad . \quad T \quad . \quad Td \quad Td \quad .T \quad . \quad T \quad . \quad Td$

Propriété 17 our tout sd ' ' ; t u, v, on a $u \rightarrow^*_{G_\Pi} v \Leftrightarrow$ $[u] < [v] < [v] < [u]$.

- 1 $\Rightarrow$ $u_0 = u \rightarrow$

Propriétés du Parc. en Profondeur - 2

$T \quad . \quad T \quad . \quad Td \quad Td \quad .T \quad . \quad T \quad . \quad Td$

Propriété 17 our tout sd ' ' ; t u, v, on a $u \rightarrow^*_{G_\Pi} v \Leftrightarrow$ $[u] < [v] < [v] < [u]$.

- 1 $\Rightarrow$ $u_0 = u \rightarrow$

Propriétés du Parc. en Profondeur - 2

Propriété 17 our tout sd ' ' st u, v, on a $u \to^*_{G_\Pi} v \Leftrightarrow$ $[u] < [v] < [v] < [u]$.

- $1 \Rightarrow$ $u_0 = u \to_{G_\Pi} u_1 \to_{G_\Pi} u_2 \ldots \to_{G_\Pi} u_k = v$
  - u_i st couvert puis u_{i-1} $[u_{i-1}] < [u_i] < [u_i] < [u_{i-1}]$.
  - par la rop. 16 $[u_{i-1}] < [u_i] < [u_i] < [u_{i-1}]$, ...
- $\Rightarrow 1$ soient u st v tq $[u] < [v] < [v] < [u]$ st $u \not\to^*_{G_\Pi} v$. n c oisit v ' an ère à ' in ' ser $[v]$. Considérons $w = \Pi v$ l xiste car $[u] < [v] < [u]$.
  - $[u] < [w]$

Propriétés du Parc. en Profondeur - 2

Propriété 17 our tout sd ' t u, v, on a $u \to^*_{G_\Pi} v \Leftrightarrow$ $[u] < [v] < [v] < [u]$.

- $1 \Rightarrow$ $u_0 = u \to_{G_\Pi} u_1 \to_{G_\Pi} u_2 \ldots \to_{G_\Pi} u_k = v$
  - u_i st couvert puis u_{i-1} $[u_{i-1}] < [u] < [u_{i-1}]$.
  - par la rop. 16 $[u_{i-1}] < [u] < [u] < [u_{i-1}]$, ...
- $\Rightarrow 1$ soient u et v tq $[u] < [v] < [v] < [u]$ et $u \not\to^*_{G_\Pi} v$. n c oisit v ' an ère à ' in ' ser $[v]$.
  Consi rons $w = \prod_{u \to^* w \not\to v} v$ l xiste car $[u] < [v] < [u]$.
  - $[u] < [w]$ Alors $[u] < [w] < [u]$ car $[w] < [v]$ rop. 16 $\Rightarrow [u] < [w] < [w] < [u]$ et onc $u \to^*_{G_\Pi} w$ car v a ts « bin » c oisi. . . Donc $u \to^*_{G_\Pi} v$, contra ict on.
  - $[w] < [u]$

Propriétés du Parc. en Profondeur - 2

Propriété 17 our tout sd ' ' st u, v, on a $u \to^*_{G_\Pi} v \Leftrightarrow$ $[u] < [v] < [v] < [u]$.

- $1 \Rightarrow$ $u_0 = u \to_{G_\Pi} u_1 \to_{G_\Pi} u_2 \ldots \to_{G_\Pi} u_k = v$
  - u_i st couvert puis u_{i-1} $[u_{i-1}] < [u] < [u_{i-1}]$.
  - par la rop. 16 $[u_{i-1}] < [u] < [u] < [u_{i-1}]$, ...
- $\Rightarrow 1$ soient u et v tq $[u] < [v] < [v] < [u]$ et $u \not\to^*_{G_\Pi} v$. n c oisit v ' ani re à ' in isr $[v]$.
  Consi rons $w =$ v I xiste car $[u] < [v] < [u]$.
  - $[u] < [w]$ Alors $[u] < [w] < [u]$ car $[w] < [v]$ rop. 16 $\Rightarrow [u] < [w] < [w] < [u]$ t onc $u \to^*_{G_\Pi} w$ car v a ts « bi n » c oisi. . Donc $u \to^*_{G_\Pi} v$, contra ict on.
  - $[w] < [u]$ u st couvert par un `PP-Visiter` G, w' ... avant l app `PP-Visiter` G, v qui couvrira v.
    Donc n $[v]$, `PP-Visiter` G, u st t ' ins t $[u] < [u] < [v] < [v]$, contra ict on.

Propriétés des arcs

n arc v, w qui n'est pas dans G_Π est un...

- arc "retour" ssi v est un descendant de w dans G_Π,
- arc "avant" ssi w est un descendant de v dans G_Π,
- un arc "transverse" dans les autres cas...

Propriétés des arcs

n arc v, w qui n'est pas dans G_Π est un...

- arc "retour" ssi v est un descendant de w dans G_Π,
- arc "avant" ssi w est un descendant de v dans G_Π,
- un arc "transverse" dans les autres cas...

Propriété 18 Un arc v, w est un...

1. arc "retour" ssi $[w] < [v] < f[v] < f[w]$,
2. arc "avant" ssi $[v] < [w] < f[w] < f[v] \wedge \pi[w] \neq v$,
3. arc "transverse" ssi $[w] < f[w] < [v] < f[v]$.

Propriétés des arcs

n arc v, w qui n'est pas dans G_Π est un...

- arc "retour" ssi v est un descendant de w dans G_Π,
- arc "avant" ssi w est un descendant de v dans G_Π,
- un arc "transverse" dans les autres cas...

Propriété 18 n arc v, w est un...

1. arc "retour" ssi $[w] < [v] < f[v] < f[w]$,
2. arc "avant" ssi $[v] < [w] < f[w] < f[v] \wedge \Pi[w] \neq v$,
3. arc "transverse" ssi $[w] < f[w] < [v] < f[v]$.

- 1 est la propriété 17...

Propriétés des arcs

n arc v, w qui n'est pas dans G_Π est un...

- arc "retour" ssi v est un descendant de w dans G_Π,
- arc "avant" ssi w est un descendant de v dans G_Π,
- un arc "transverse" dans les autres cas...

Propriété 18 n arc v, w est un...

1. arc "retour" ssi $[w] < [v] < f[v] < f[w]$,
2. arc "avant" ssi $[v] < [w] < f[w] < f[v] \wedge \Pi[w] \neq v$,
3. arc "transverse" ssi $[w] < f[w] < [v] < f[v]$.

- 1 est la propriété 17...
- 3 on a soit $[v] < f[v] < [w] < f[w]$ ou $[w] < f[w] < [v] < f[v]$.
  dans ce cas là, v, w n'est pas dans G_Π, w ne peut être ni ancêtre ni descendant de $[v]$.
  Donc $[w] < f[w] < [v] < f[v]$.

Propriétés du Parcours en Profondeur

Propriété 19 A tout moment de l'algorithme, les sommets grisés forment un chemin reliés par des arcs de G_Π.

Propriétés du Parcours en Profondeur

Propriété 19 A tout instant de l'algorithme de parcours en profondeur, les sommets gris forment un chemin reliés par des arcs de G_Π.

A tout instant de l'algo, on peut trier les sommets gris $u_1, \ldots, u_k$ par $[\,\cdot\,]$ croissants.

$$[u_1] < [u_2] < \ldots < [u_k]$$

Propriétés du Parcours en Profondeur

Propriété 19 A tout moment de l'algorithme, le parcours en profondeur, les sommets gris forment un chemin reliés par les arcs de G_Π.

A tout moment de l'algo, on peut trier les sommets gris $u_1, \ldots, u_k$ par $[\,\cdot\,]$ croissants.

$$[u_1] < [u_2] < \ldots < [u_k] < [u_k] < \ldots < [u_2] < [u_1]$$

Prop. 16

Propriétés du Parcours en Profondeur

Propriété 19 A tout moment de l'algorithme, le parcours en profondeur, les sommets gris forment un chemin reliés par des arcs de G_Π.

A tout moment de l'algo, on peut trier les sommets gris $u_1, \ldots, u_k$ par $[\cdot]$ croissants.

$$[u_1] < [u_2] < \ldots < [u_k] < f[u_k] < \ldots < f[u_2] < f[u_1]$$

prop. 16

Donc $u_1 \to^*_{G_\Pi} u_2 \to^*_{G_\Pi} \ldots \to^*_{G_\Pi} u_k$ prop. 17

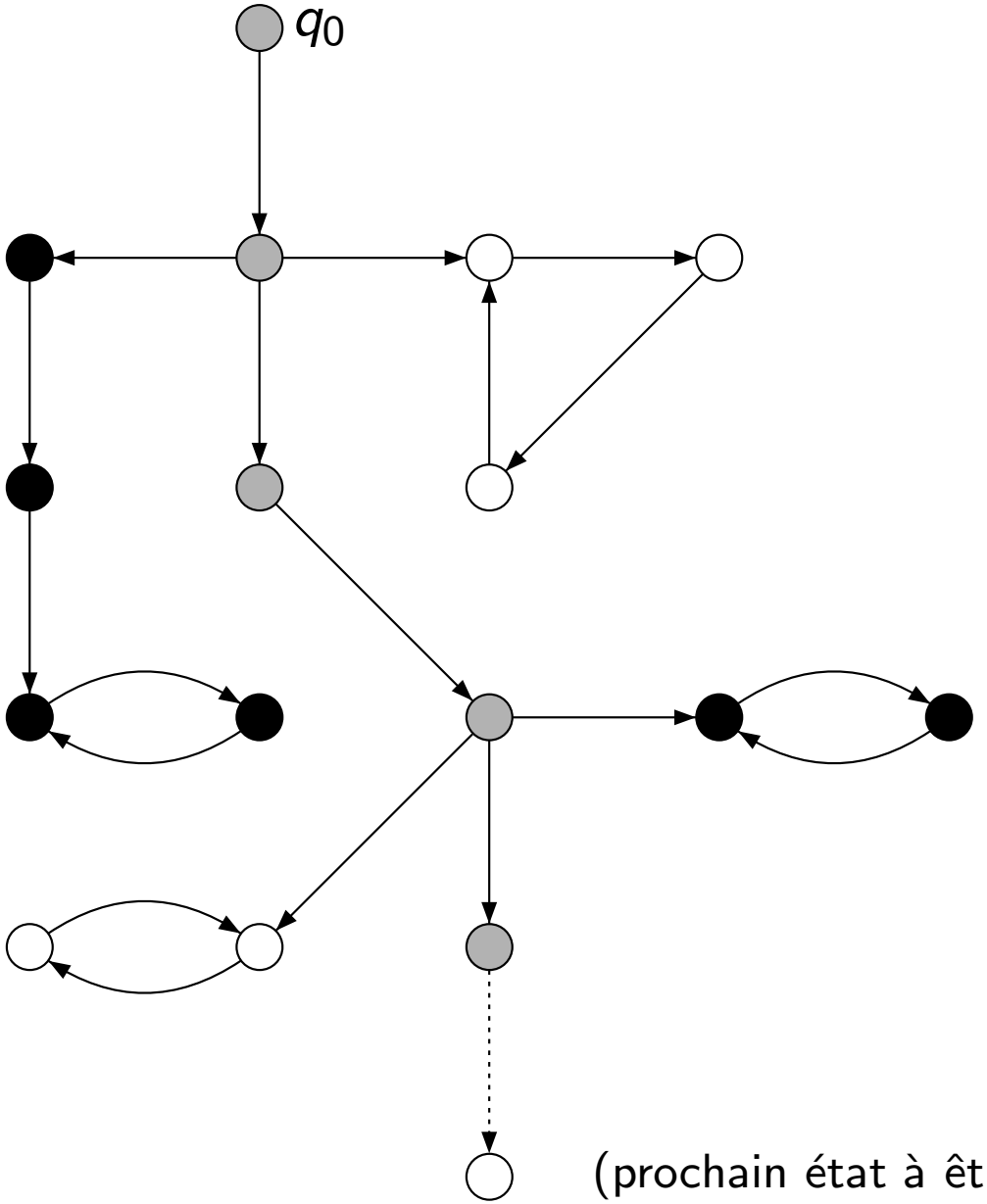
Propriétés du Parcours en Profondeur

Propriété 19 A tout moment de l'algorithme de parcours en profondeur, les sommets gris forment un chemin reliés par des arcs de G_Π.

A tout moment de l'algo, on peut trier les sommets gris $u_1, \ldots, u_k$ par $[\cdot]$ croissants.

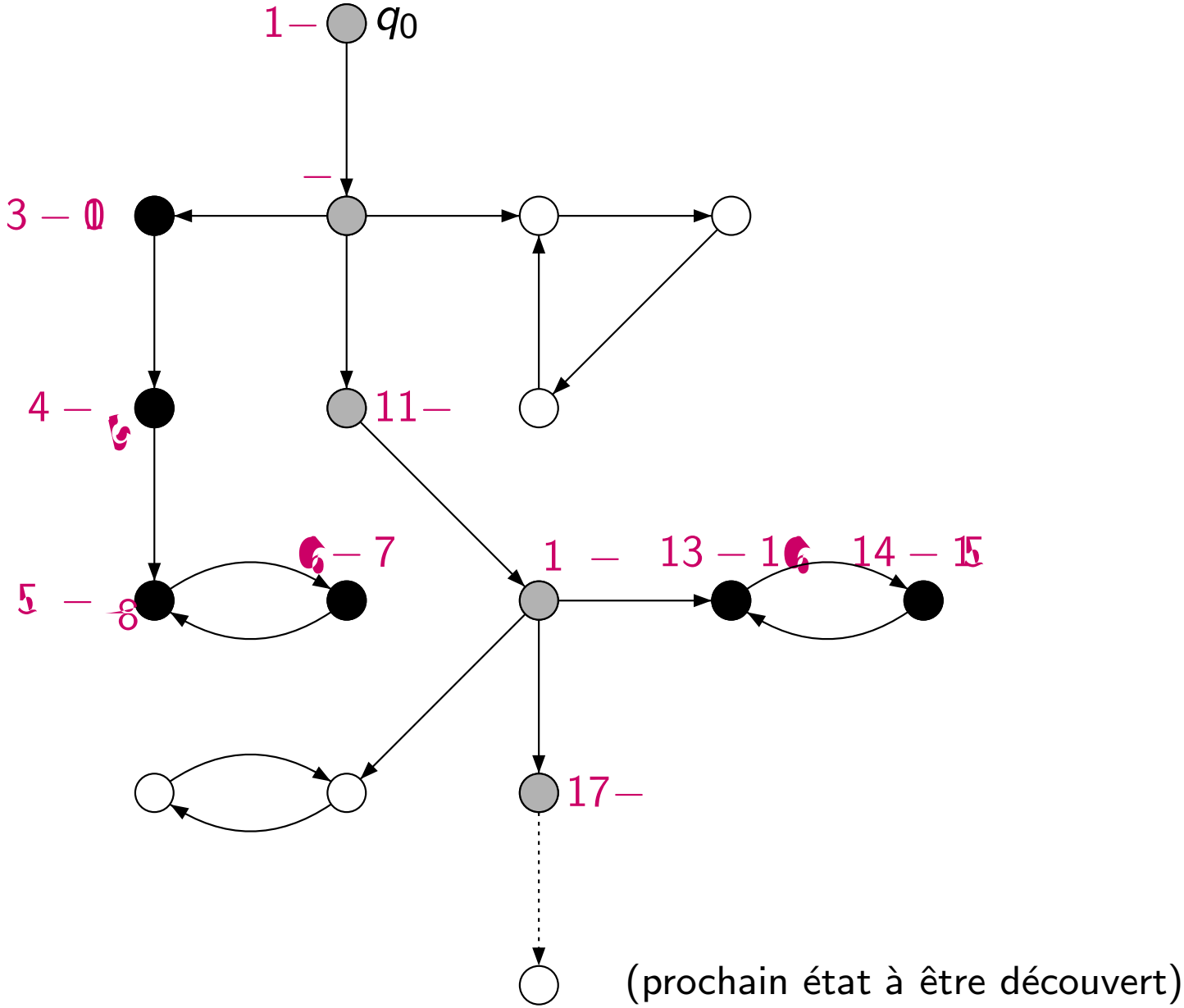
$$[u_1] < [u_2] < \ldots < [u_k] < [u_k] < \ldots < [u_k]$$

Vision globale du graphe



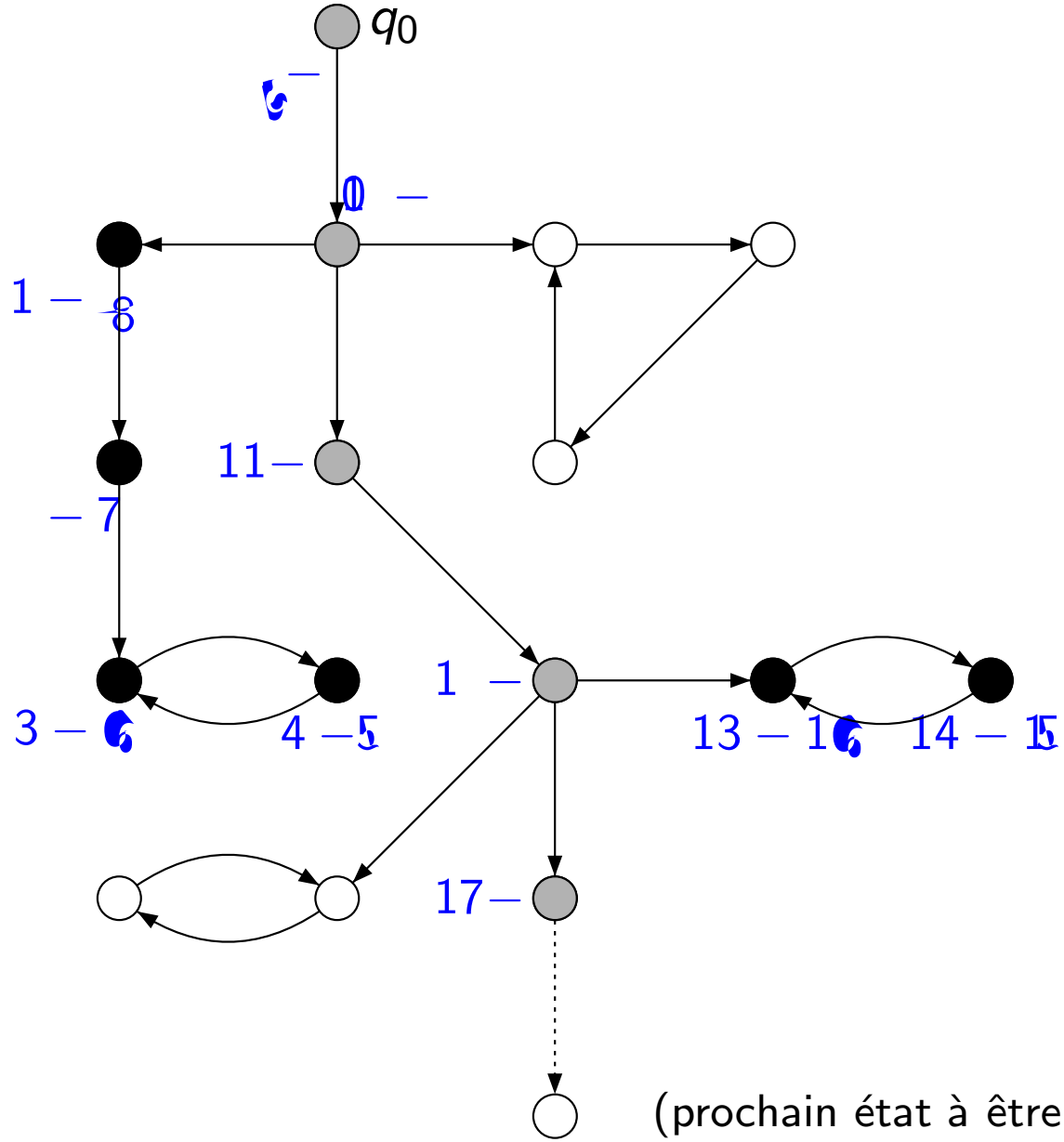
Exercice : Combien en est-on arrivé à

Vision globale du graphe



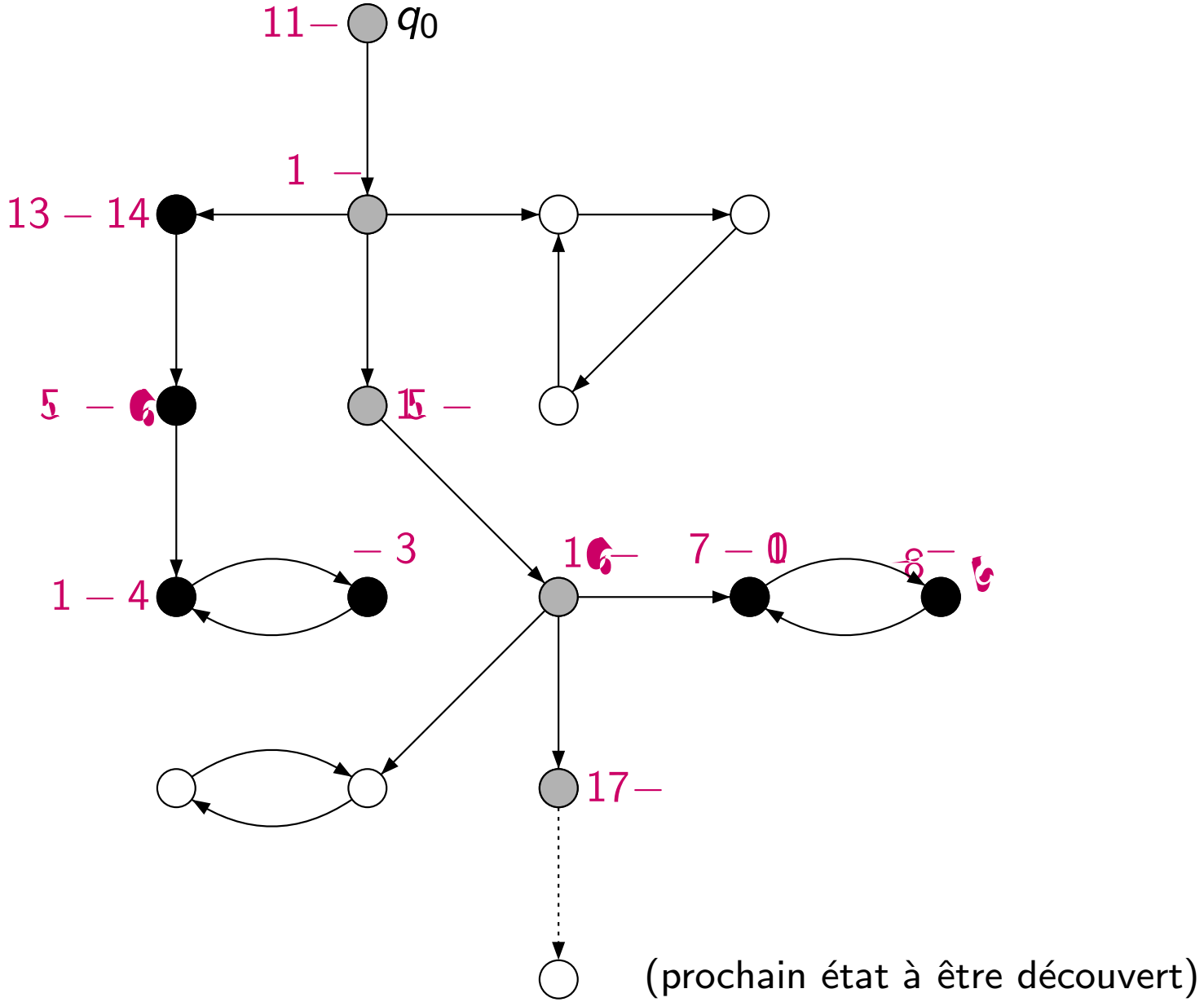
Exercice : Combien peut-on arriver à

Vision globale du graphe



Exercice : Combien peut-on arriver à

Vision globale du graphe



Exercice : Combien peut-on arriver à

Propriétés du Parcours en Profondeur

Enoncer la correction → PP-Visiter G, s

Propriétés du Parcours en Profondeur

Enoncer la correction : `PP-Visiter` G, s

Idée générale : tous les états accessibles depuis s et non encore découverts depuis le début de la procédure, seront découverts, et seront donc des ascendants dans G_Π...

Propriétés du Parcours en Profondeur

Enoncer la correction : `PP-Visiter` G, s

Idée générale : tous les états accessibles depuis s et non encore couverts depuis le but de la procédure, seront couverts, et seront donc des ancêtres de s dans G_Π...

Théorème du chemin blanc

v est un ancêtre de u dans G_Π **si et seulement si** à la date $[u]$, le sommet v était atteignable depuis u par un chemin composé uniquement de sommets blancs.

Preuve de la correction du PP

$u \to^*_{G_\Pi} v$ ssi si on [u], il y a un chemin blanc de u à v.

Preuve de la correction du PP

$u \to^*_{G_\Pi} v$ ssi [illegible] $[u]$, il y a un chemin blanc de u à v.

1 $\Rightarrow$ tout sous-mot strictement long d'un $u \to^*_{G_\Pi} v$ vérifie $[u] < [w]$ (prop. 17) donc chaque w est blanc en $[u]$.

Preuve de la correction du PP

$u \to^*_{G_\Pi} v$ ssi il y a un calcul en $[u]$, il y a un calcul en banc de u à v.

1 $\Rightarrow$ tout sous-ensemble strictement décroissant de $[u] \to^*_{G_\Pi} v$ est de longueur $\leq [u]$ (prop. 17). Donc chaque banc est banc en $[u]$.

$\Rightarrow$ 1 soit v atteignable par un calcul en banc ρ depuis u en $[u]$ tq $u \not\to^*_{G_\Pi} v$. On prend la plus petite valeur de c telle que ρ.

Preuve de la correction du PP

$u \to^*_{G_\Pi} v$ ssi en [u], il y a un chemin dans u à v.

1 $\Rightarrow$ tout sous-mot strictement long d'un chemin $u \to^*_{G_\Pi} v$ vérifie [u] $<$ [w] (prop. 17). Donc chaque w est dans [u].

$\Rightarrow$ 1 soit v atteignable par un chemin dans ρ puis u en [u] tq $u \not\to^*_{G_\Pi} v$. On prend l'préfixe v de type long ρ. Pour tout processeur w de v sur ρ, on a $u \to^*_{G_\Pi} w$.

Preuve de la correction du PP

$u \to^*_{G_\Pi} v$ ssi il existe $[u]$, il y a un chemin blanc de u à v.

$1 \Rightarrow$ tout sous-mot strictement long de u chemin $u \to^*_{G_\Pi} v$ vérifie $[u] < [w]$ (prop. 17) donc chaque w est blanc en $[u]$.

$\Rightarrow 1$ soit v atteignable par un chemin blanc ρ puis u en $[u]$ tq $u \not\to^*_{G_\Pi} v$. On prend le préfixe de v de type strictement long de ρ. Pour tout processus sur w de v sur ρ, on a $u \to^*_{G_\Pi} w$.
Donc $[u] < [w] < [w] < [u]$ (prop. 17)

Preuve de la correction du PP

$u \to^*_{G_\Pi} v$ ssi en $[u]$, il y a un chemin blanc de u à v.

$1 \Rightarrow$ tout sous-mot w long de u chemin $u \to^*_{G_\Pi} v$ vérifie $[u] < [w]$ (prop. 17) donc chemin aqué w est blanc en $[u]$.

$\Rightarrow 1$ soit v atteignable par un chemin blanc ρ puis u en $[u]$ tq $u \not\to^*_{G_\Pi} v$. On prend ρ préfixe propre de v de type-long ρ. Pour tout préfixe sur w de v sur ρ, on a $u \to^*_{G_\Pi} w$.

Donc $[u] < [w] < [w] < [u]$ (prop. 17)

Considérons le préfixe sur ρ de cas w tq $\exists w, v \in A$.

Cad $w \not\to^*_{G_\Pi} v$, v étant strictement gris avant $[w]$: $[v] < [w]$.

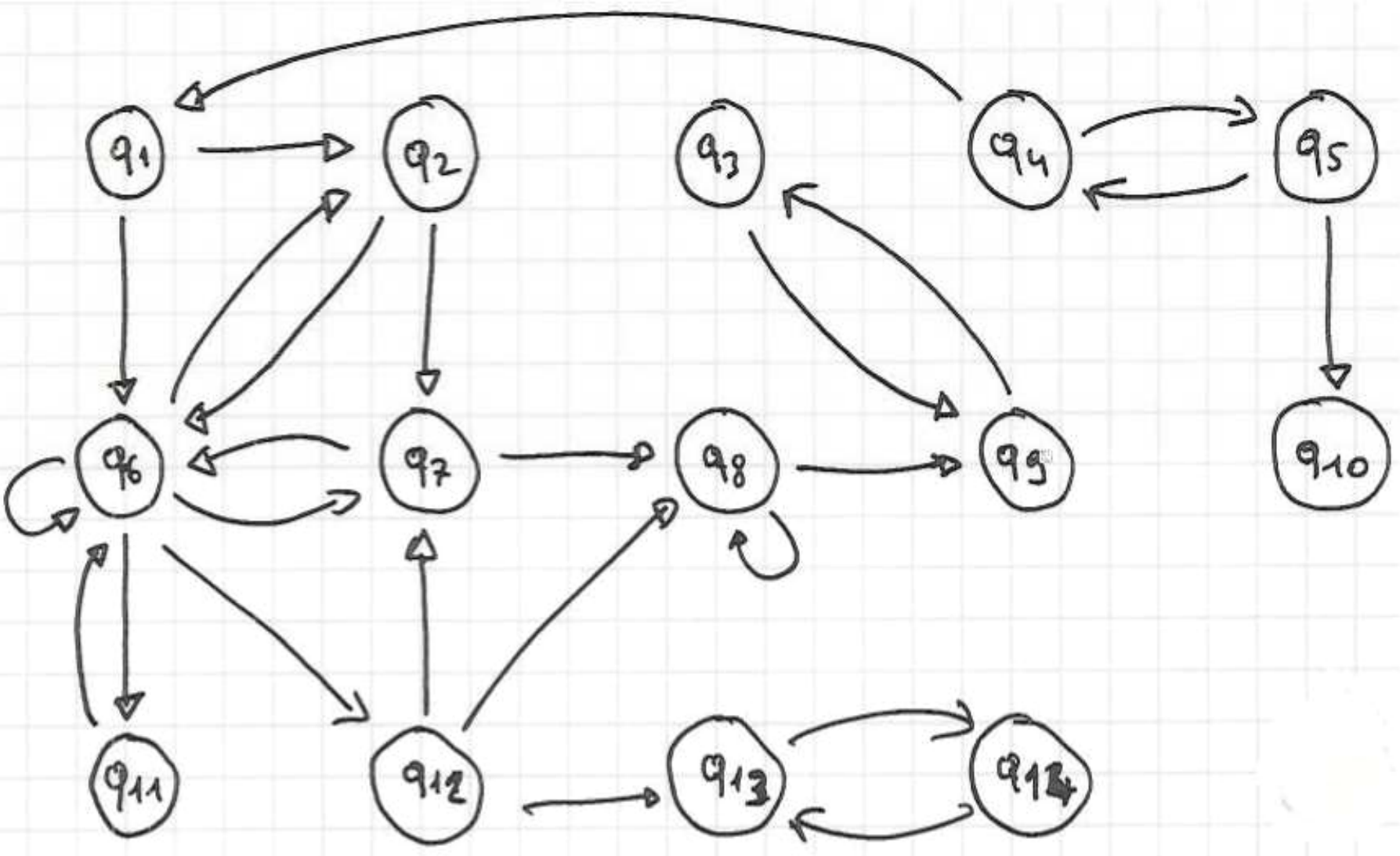
D'après le lemme 16, on a deux cas :

$$[w] < [v] < [v] < [w] \quad \textbf{ou} \quad [v] < [v] < [w] < [w]$$

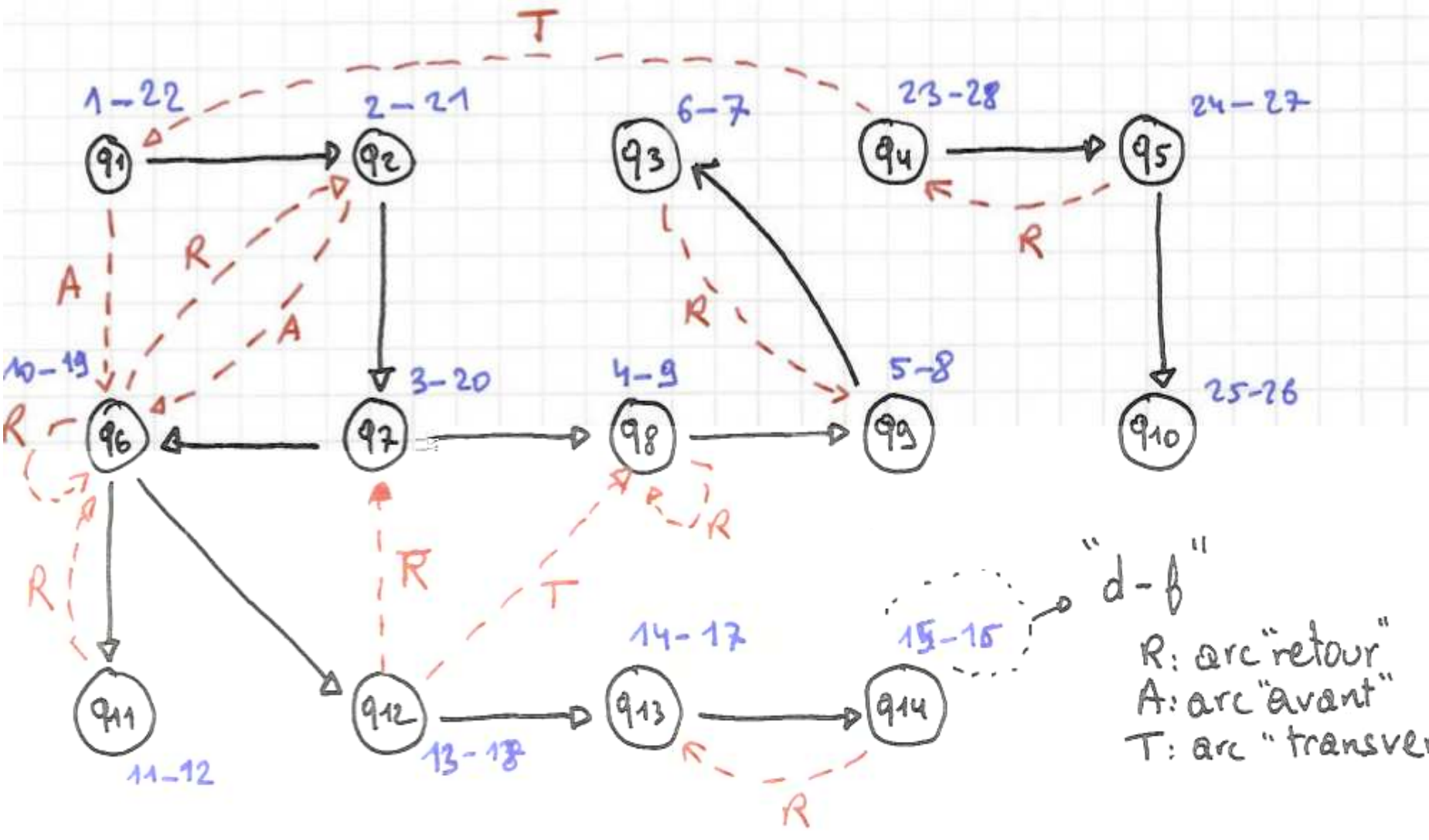
D'où $[[v], [v]] \subset [[u], [u]]$ et par le lemme prop. 17, on a $u \to^*_{G_\Pi} v$.

Contradiction.

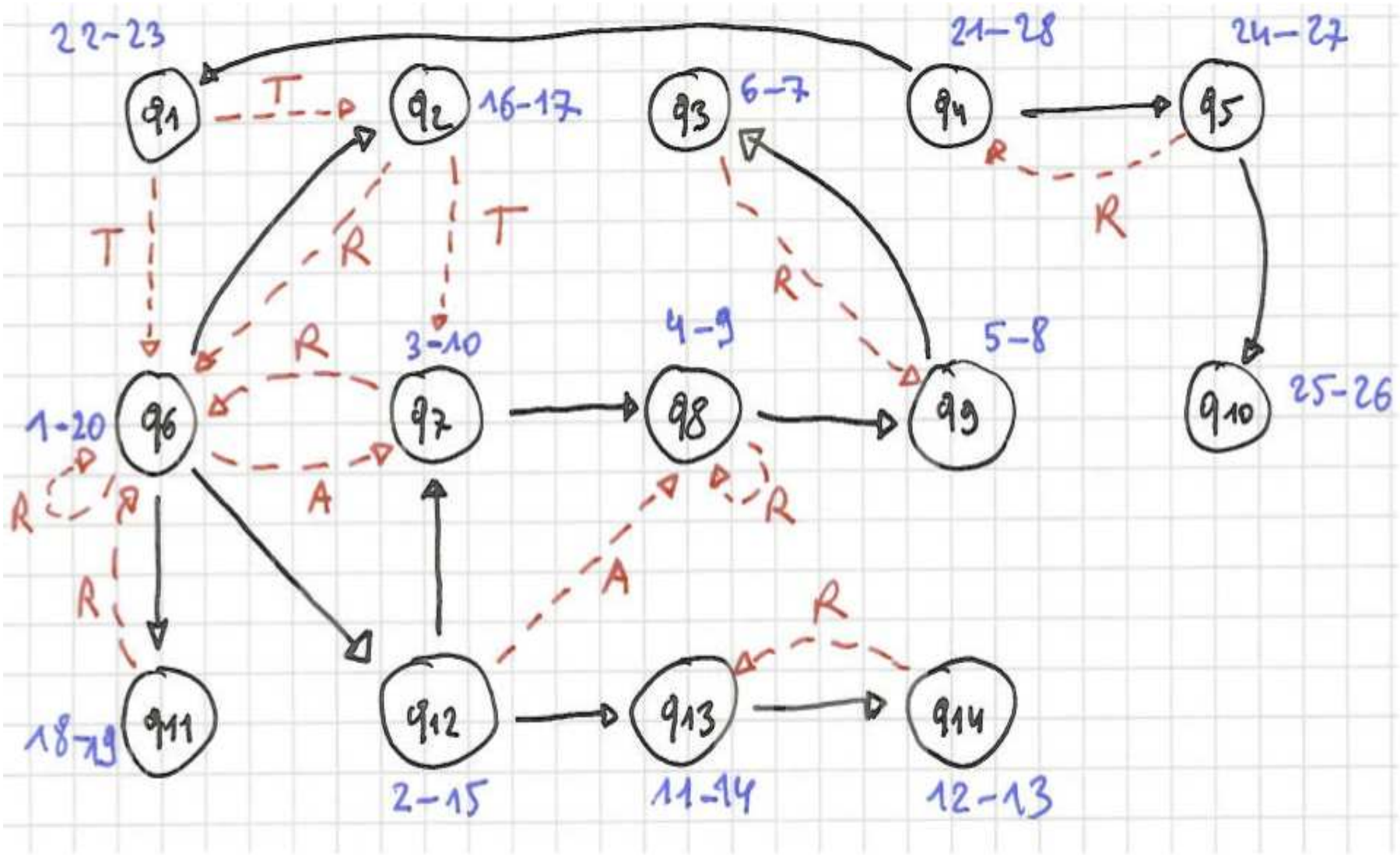
Exemple



Exemple


T
1-22
2-21
6-7
23-28
24-27
q1
q2
q3
q4
q5
A
R
A
R
R
R
10-19
3-20
4-9
5-8
25-26
R
q6
q7
q8
q9
q10
R
R
R
T
14-17
15-15
"d-f"
q11
q12
q13
q14
11-12
13-18
R
R: arc "retour"
A: arc "avant"
T: arc "transver

Exemple


22-23
21-28
24-27
q1
T
q2
16-17
q3
6-7
q4
q5
T
R
T
R
R
R
4-9
3-10
5-8
1-20
q6
R
q7
q8
q9
q10
25-26
R
R
A
R
R
A
R
18-19
q11
q12
q13
q14
2-15
11-14
12-13

P

Extension linéaire (tri topologique)

Soit $G = (S, A)$ un graphe orienté acyclique (DAG).

Définition Une extension linéaire de G est un ordre total $\leq$ sur les sommets **compatible** avec A, c'est-à-dire tel que pour tout u et v, on a : $(u, v) \in A \Rightarrow u \leq v$.

Extension linéaire (tri topologique)

Soit $G = (S, A)$ un graphe orienté acyclique DAG.

Définition Une extension linéaire de G est un ordre total $\leq$ sur les sommets **compatible** avec A, c'est-à-dire tel que pour tout u et v, on a $(u, v) \in A \Rightarrow u \leq v$.

Si G acyclique $\Rightarrow$ la relation $\rightarrow^*$ induit un ordre partiel.

Comment en faire un ordre total

Applications

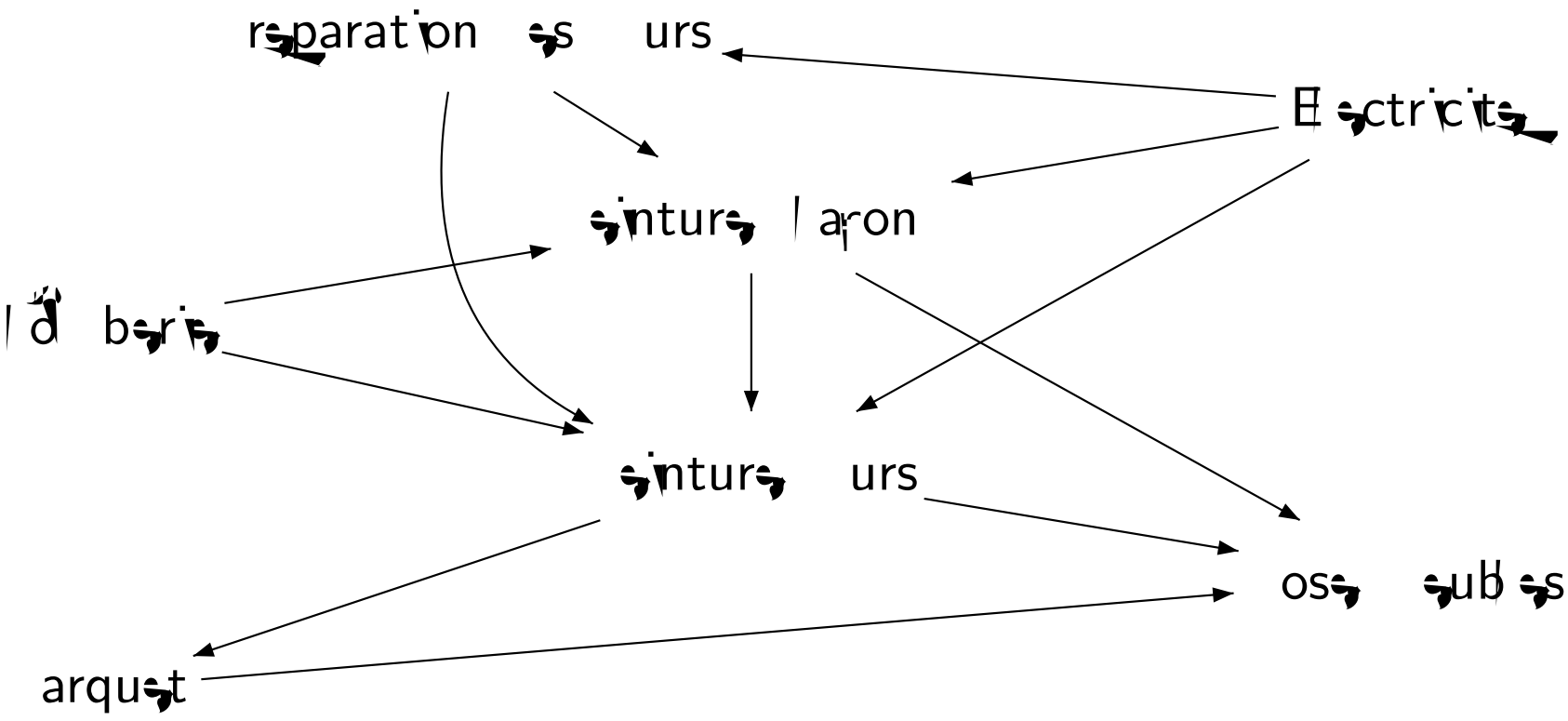
Ordonnancements de tâches...

n tâches à planifier en respectant des contraintes du genre « T_i doit être traitée avant T_j »...

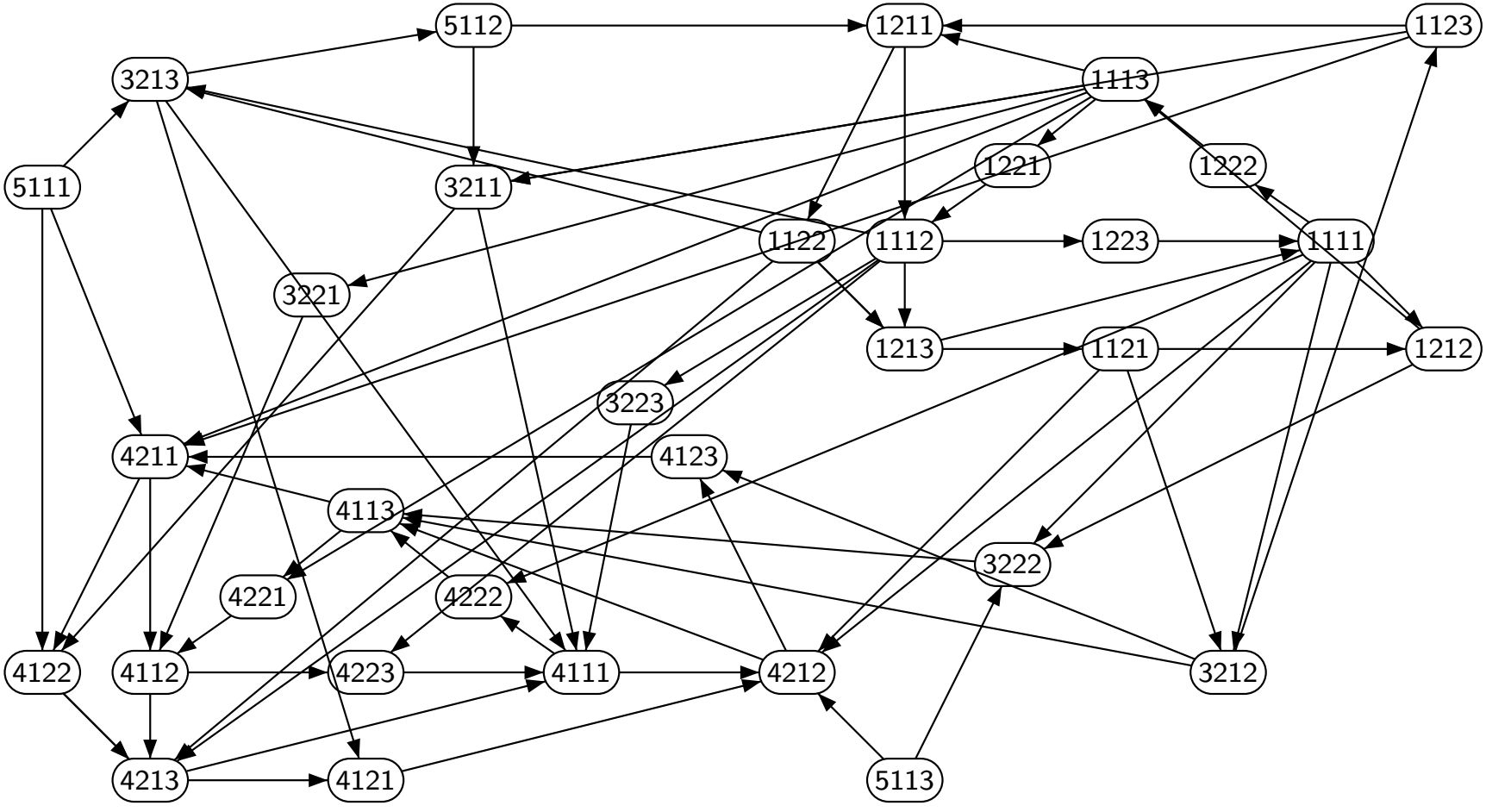
Applications

Li or onnanc nt tac s...

n tac s à p an r n respectant s contra ntes u genre « T_i oit tre trait avant T_j »...



Et là ?


5112
1211
1123
3213
1113
1221
1222
5111
3211
1122
1112
1223
1111
3221
1213
1121
1212
3223
4211
4123
4113
3222
4221
4222
4122
4112
4223
4111
4212
3212
4213
4121
5113

Algorithme de recherche d'extension linéaire

1. Appeler PP G ...
2. Retenir le plus court sommet trouvé lorsque l'on est contraint de noircir un nouveau sommet

La pile contient tous les sommets dans l'ordre croissant.

Les sommets de la pile sont les sommets $\leq$ au nouveau sommet noirci. pour $\leq$ il n'a pas de prédécesseur dans G.

Algorithme de recherche d'extension linéaire

1. Appeler PP G ...
2. Retirer de P tous les sommets minimaux lorsqu'il en existe un noir

La preuve contient tous les sommets dans l'ordre croissant.

Les sommets de P a plus petit sommet v tel que pour $\leq_A$ il n'a pas de prédécesseur dans G.

ou :

1. Appeler PP G ...
2. Ranger les sommets par $\mathrm{rang}_A[\,]$ croissants.

L'ordre $\leq_A$ calculé est donc : $u \leq_A v$ si $\mathrm{rang}_A[u] > \mathrm{rang}_A[v]$

Les deux définitions de $\mathrm{rang}_A[\,]$ sont équivalentes.

Algo. de recherche d'extension linéaire

Procédure `Tri-Topo` G
pour chaque $x \in S$ **faire** $Couleur[x] = blanc$;
$P = PileVide$;
pour chaque $x \in S$ **faire**
  si $Couleur[x] = blanc$ **alors** `PP-Visiter2` G, x ;

Avec :

Procédure `PP-Visiter2` G, s
$Couleur[s] = gris$;
pour chaque $s, u \in A$ **faire**
  si $Couleur[u] = blanc$ **alors** `PP-Visiter2` G, u ;
$Couleur[s] = noir$;
$P.Empiler(s)$;

Correction de l'algorithme

Théorème

Soit G un graphe orienté acyclique. L'ordre $\leq_A$ calculé par l'algorithme est une extension linéaire de G.

Correction de l'algorithme

Théorème

Soit G un graphe orienté acyclique. L'ordre $\leq_A$ calculé par l'algorithme est une extension linéaire de G.

Dans l'algorithme, on a montré :

Propriété 19

G est acyclique ssi l'algo. PP ne trouve aucun arc retour.

Correction de l'algorithme

Théorème

Soit G un graphe orienté acyclique. L'ordre $\leq_A$ calculé par l'algorithme est une extension linéaire de G.

Démontrons :

Propriété 19

G est acyclique ssi l'algo. PP ne trouve aucun arc retour.

$1 \Rightarrow$ Si u, v est un arc retour, alors il y a un chemin gris reliant v à u, l'arc u, v ferme le cycle.

Correction de l'algorithme

Théorème

Soit G un graphe orienté acyclique. L'ordre $\leq_A$ calculé par l'algorithme est une extension linéaire de G.

Dans un premier temps, on montre :

Propriété 19

G est acyclique ssi l'algo. PP ne trouve aucun arc retour.

$1 \Rightarrow$ Si u, v est un arc retour, alors il y a un chemin gris reliant v à u, l'arc u, v ferme le cycle.

$\Rightarrow 1$ Soit ρ un cycle. Soit v le premier sommet de ρ découvert par PP. Le sommet u du chemin blanc garantit $v \rightarrow^*_{G_\Pi} u$ et donc u, v sera vu comme un arc retour.

Correction de l'algorithme

Théorème

Soit G un graphe orienté acyclique. L'ordre $\leq_A$ calculé par l'algorithme est une extension linéaire de G.

Correction de l'algorithme

Théorème

Soit G un graphe orienté acyclique. L'ordre $\leq_A$ calculé par l'algorithme est une extension linéaire de G.

Etant donnés $u, v \in S$ tels que $u, v \in A$.

Objectif : il faut montrer $u \leq_A v$, et donc $r[u] > r[v]$.

Correction de l'algorithme

Théorème

Soit G un graphe orienté acyclique. L'ordre $\leq_A$ calculé par l'algorithme est une extension linéaire de G.

Etant donnés $u, v \in S$ tels que $u, v \in A$.

Objectif : montrer $u \leq_A v$, et donc $f[u] > f[v]$.

Lorsque l'arête u, v est étudiée dans PP-Visiter2, alors v ne peut pas être gris car u, v serait un arc retour et G aurait un cycle. Donc :

- soit v est blanc et sera visité après u et alors on aura $f[u] > f[v]$;
- soit v est noir et alors $f[v] < f[u]$.

Le résultat recherché est donc vrai.