Parcours de graphes

Algorithmique – L3

François Laroussinie

2 novembre 2010

```
Procédure Parcours(G)
//G = (S, A)
begin
    pour chaque x ∈ S faire Couleur[x] := blanc
    Choisir s ∈ S
    Couleur[s] := gris
    répéter
        Choisir x tq Couleur[x] = gris
        pour chaque (x, y) ∈ A faire
            si Couleur[y] = blanc alors Couleur[y] := gris
        Couleur[x] := noir
    jusqu'à ∀x.Couleur[x] ≠ gris ;
end
```

Parcours

Idées :

• On choisit un sommet s dont tous les successeurs n'ont pas encore été découverts.

• On explore les transitions issues de s et on cherche des successeurs de s pas encore découverts.

Parcours

Idées :

• On choisit un sommet s dont tous les successeurs n'ont pas encore été découverts.

• On explore les transitions issues de s et on cherche des 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.

$\rightarrow$ trois couleurs (blanc/gris/noir).

Plan

Plan

Parcours en largeur

on considère un graphe non orienté.

Arborescence de parcours

Pour chaque sommet, on va garder en mémoire par quelle arête il a été découvert, c.-à-d. depuis quel sommet. . .

Arborescence de parcours

Pour chaque sommet, on va garder en mémoire par quelle arête il a été découvert, c.-à-d. depuis quel sommet...

$\Pi : S \to S \cup \{\text{nil}\}$
$\Pi(u) = v \quad \text{ssi} \quad u$ a été découvert depuis v.

Arborescence de parcours

Pour chaque sommet, on va garder en mémoire par quelle arête il a été découvert, c.-à-d. depuis quel sommet...

$\Pi : S \to S \cup \{\text{nil}\}$
$\Pi(u) = v \quad \text{ssi} \quad u$ a été découvert depuis v.

On construit donc un arbre $G_\Pi = (S, A_\Pi)$ de racine s.

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

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 chaque sommet à l'origine s, et
2. les chemins dans G_Π seront des plus courts chemins de G.

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

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

Parcours en largeur

Le sens du coloriage 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 immédiats n'ont pas encore été tous découverts ;
- noir : les sommets découverts dont tous les successeurs immédiats ont aussi été découverts.

Terminaison et complexité

Tout ajout de u dans F...

- est conditionné par « $\text{Couleur}[u] = \text{blanc}$ », et
- est suivi par un coloriage en gris de u.
- Un sommet n'est colorié en blanc qu'à l'initialisation.

Terminaison et complexité

Tout ajout de u dans F...

- est conditionné par « $\text{Couleur}[u] = \text{blanc}$ », et
- est suivi par un coloriage en gris de u.
- Un sommet n'est colorié en blanc qu'à l'initialisation.

$\Rightarrow$ u ne peut être ajouté (et extrait) qu'au plus une fois.

$\Rightarrow$ l'algorithme termine !

Terminaison et complexité

Tout ajout de u dans F...

- est conditionné par « Couleur$[u]$ = blanc », et
- est suivi par un coloriage en gris de u.
- Un sommet n'est colorié en blanc qu'à l'initialisation.

$\Rightarrow$ u ne peut être ajouté (et extrait) qu'au plus une fois.
$\Rightarrow$ l'algorithme termine !
Et sa liste d'adjacence n'est parcourue qu'au plus une fois.

Terminaison et complexité

Tout ajout de u dans F...

- est conditionné par « Couleur$[u] =$ blanc », et
- est suivi par un coloriage en gris de u.
- Un sommet n'est colorié en blanc qu'à l'initialisation.

$\Rightarrow$ u ne peut être ajouté (et extrait) qu'au plus une fois.
$\Rightarrow$ l'algorithme termine !
Et sa liste d'adjacence n'est parcourue qu'au plus une fois.
$\Rightarrow$ Complexité totale :

- de la boucle principale : $O(|A|)$;
- de l'initialisation : $O(|S|)$.

$\Rightarrow$ Complexité en $O(|S| + |A|)$ 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 $\mathtt{PL}(G, s)$:

1. découvre tous les sommets atteignables depuis s et uniquement eux ;
2. termine avec $\mathrm{Dist}[v] = \delta(s, v)$ pour tout $v \in S$;
3. construit la table Π de telle sorte que pour tout sommet $u \neq s$ atteignable depuis s, il existe un plus court chemin de s à u dans G dont la dernière transition est $(\Pi(u), u)$.

NB : $\delta(s, v) \stackrel{\text{def}}{=}$ longueur d'un plus court chemin entre s et v

Propriétés – distances

Définition $G = (S, A), \quad u, v \in S$

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

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 : $\mathrm{Dist}[u] \geq \delta(s, u)$.

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

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

Plan

Idée générale

On considère des graphes orientés.

Ici on choisit le sommet gris découvert le plus récemment.
Au lieu d'une file, on utilise une pile.

Idée générale

On considère des graphes orientés.

Ici on choisit le sommet gris découvert le plus récemment.
Au lieu d'une file, on utilise une pile.
On déroule donc un chemin (gris) le plus loin possible.

Idée générale

On considère des graphes orientés.

Ici on choisit le sommet gris découvert le plus récemment.
Au lieu d'une file, on utilise une pile.
On déroule donc un chemin (gris) le plus loin possible.

Les trois couleurs signifient :

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

Algorithme - 1

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

Algorithme - 1

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

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

- une date de coloriage en gris $\mathrm{d}[u]$; et
- une date de coloriage en noir $\mathrm{f}[u]$

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

Algorithme - 1

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

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

- une date de coloriage en gris $\mathrm{d}[u]$; et
- une date de coloriage en noir $\mathrm{f}[u]$

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

$\mathrm{d}[u] < \mathrm{f}[u]$

Et...

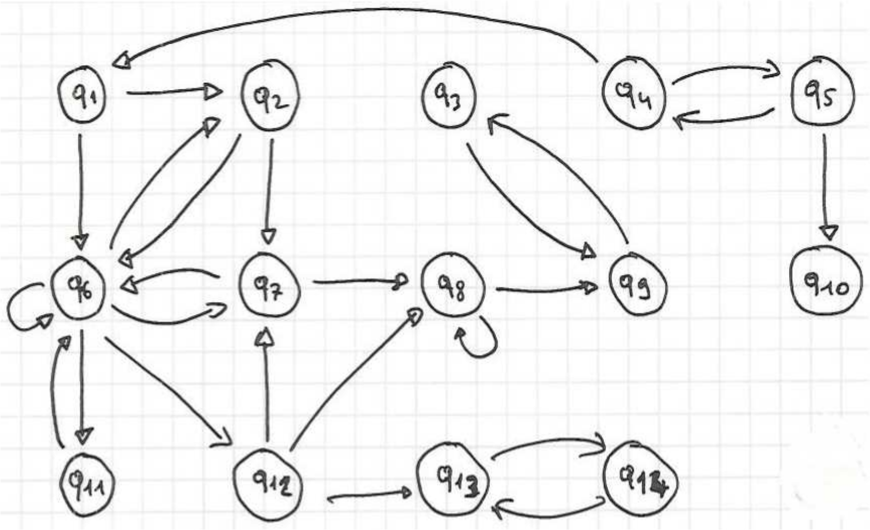
- avant la date $\mathrm{d}[u]$, u est en blanc ;
- entre $\mathrm{d}[u]$ et $\mathrm{f}[u]$, u est en gris ;
- après la date $\mathrm{f}[u]$, u est en noir.

Parcours en profondeur (init)

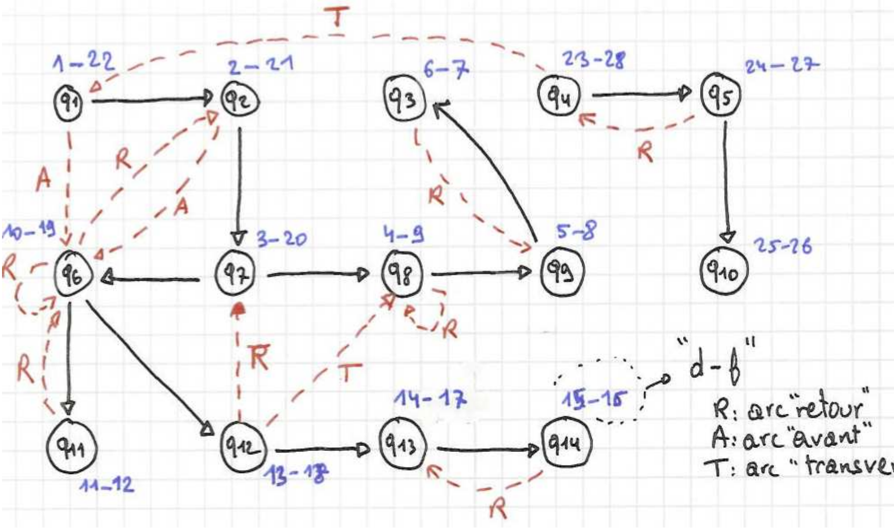
```
Procédure PP(G)
//G = (S, A)
begin
    pour chaque x ∈ S faire
        Couleur[x] := blanc;
        Π[x] := nil;
    temps := 0;
    pour chaque x ∈ S faire
        si Couleur[x] = blanc alors
            PP-Visiter(G, x);
end
```

Procédure `PP-Visiter`(G, s)
Couleur$[s]$:= gris;
temps $++$;
d$[s]$:= *temps*;
pour chaque $(s, u) \in A$ **faire**
    si *Couleur*$[u]$ = *blanc* **alors**
        $\Pi[u] := s$;
        `PP-Visiter`(G, u) ;
Couleur$[s]$:= noir ;
temps $++$;
f$[s]$:= *temps* ;

Exemple

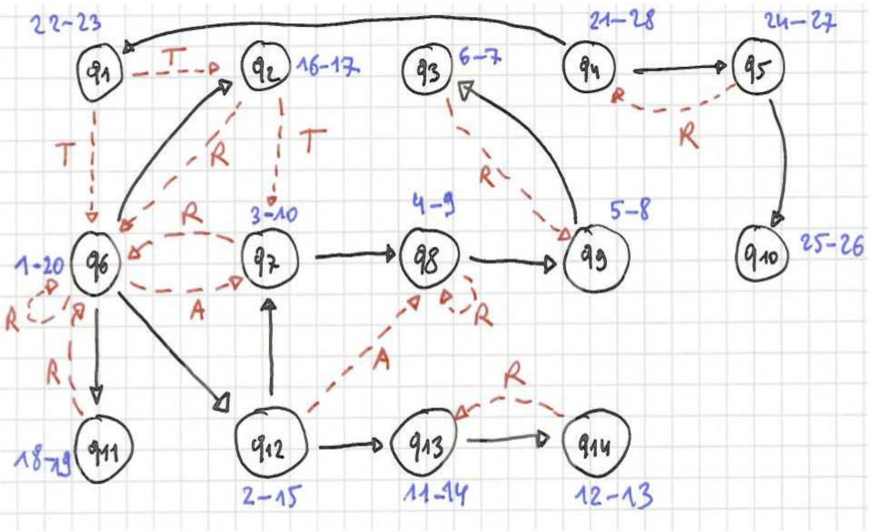


Exemple



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

Exemple



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

Complexité

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

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

La complexité des boucles « **Pour chaque** $(s, u) \ldots$ » de 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. . .

Tout arc (v, w) de A est soit un arc de G_Π, soit. . .

- un arc de G_Π (*i.e.*$\Pi[w] = v$) ;
- un arc « retour » : v est un descendant de w dans G_Π (*) ;
- un arcs « avant » : w est un descendant de v dans G_Π (mais $\Pi[w] \neq v$) (*) ;
- un arc « transverse » (tous les autres cas !).

(*) lors de l'examen de (v, w) dans `PP-Visiter`.

Propriétés du Parc. en Profondeur - 1

Propriété 16 Pour tout sommet u et v, on a :

- soit les intervalles $[\mathrm{d}[u]; \mathrm{f}[u]]$ et $[\mathrm{d}[v]; \mathrm{f}[v]]$ sont disjoints ;
- soit $[\mathrm{d}[u]; \mathrm{f}[u]]$ est contenu dans $[\mathrm{d}[v]; \mathrm{f}[v]]$;
- soit $[\mathrm{d}[v]; \mathrm{f}[v]]$ est contenu dans $[\mathrm{d}[u]; \mathrm{f}[u]]$.

Propriétés du Parc. en Profondeur - 1

Propriété 16 Pour tout sommet u et v, on a :

- soit les intervalles $[\mathrm{d}[u]; \mathrm{f}[u]]$ et $[\mathrm{d}[v]; \mathrm{f}[v]]$ sont disjoints ;
- soit $[\mathrm{d}[u]; \mathrm{f}[u]]$ est contenu dans $[\mathrm{d}[v]; \mathrm{f}[v]]$;
- soit $[\mathrm{d}[v]; \mathrm{f}[v]]$ est contenu dans $[\mathrm{d}[u]; \mathrm{f}[u]]$.

Supposons $\mathrm{d}[u] < \mathrm{d}[v]$.

- $\mathrm{d}[v] < \mathrm{f}[u]$:

Propriétés du Parc. en Profondeur - 1

Propriété 16 Pour tout sommet u et v, on a :

- soit les intervalles $[\mathrm{d}[u]; \mathrm{f}[u]]$ et $[\mathrm{d}[v]; \mathrm{f}[v]]$ sont disjoints ;
- soit $[\mathrm{d}[u]; \mathrm{f}[u]]$ est contenu dans $[\mathrm{d}[v]; \mathrm{f}[v]]$;
- soit $[\mathrm{d}[v]; \mathrm{f}[v]]$ est contenu dans $[\mathrm{d}[u]; \mathrm{f}[u]]$.

Supposons $\mathrm{d}[u] < \mathrm{d}[v]$.

- $\mathrm{d}[v] < \mathrm{f}[u]$:
  - En $\mathrm{d}[v]$, u est gris : `PP-Visiter`(G, u) n'est pas fini.

Propriétés du Parc. en Profondeur - 1

Propriété 16 Pour tout sommet u et v, on a :

- soit les intervalles $[\mathrm{d}[u]; \mathrm{f}[u]]$ et $[\mathrm{d}[v]; \mathrm{f}[v]]$ sont disjoints ;
- soit $[\mathrm{d}[u]; \mathrm{f}[u]]$ est contenu dans $[\mathrm{d}[v]; \mathrm{f}[v]]$;
- soit $[\mathrm{d}[v]; \mathrm{f}[v]]$ est contenu dans $[\mathrm{d}[u]; \mathrm{f}[u]]$.

Supposons $\mathrm{d}[u] < \mathrm{d}[v]$.

- $\mathrm{d}[v] < \mathrm{f}[u]$:
  - En $\mathrm{d}[v]$, u est gris : `PP-Visiter`(G, u) n'est pas fini.
  - on va appeler `PP-Visiter`(G, v') pour tout $(v, v') \in A$ t.q. $\mathrm{Couleur}[v'] = \mathrm{blanc}$.

Propriétés du Parc. en Profondeur - 1

Propriété 16 Pour tout sommet u et v, on a :

- soit les intervalles $[\mathrm{d}[u]; \mathrm{f}[u]]$ et $[\mathrm{d}[v]; \mathrm{f}[v]]$ sont disjoints ;
- soit $[\mathrm{d}[u]; \mathrm{f}[u]]$ est contenu dans $[\mathrm{d}[v]; \mathrm{f}[v]]$;
- soit $[\mathrm{d}[v]; \mathrm{f}[v]]$ est contenu dans $[\mathrm{d}[u]; \mathrm{f}[u]]$.

Supposons $\mathrm{d}[u] < \mathrm{d}[v]$.

- $\mathrm{d}[v] < \mathrm{f}[u]$:
  - En $\mathrm{d}[v]$, u est gris : `PP-Visiter`(G, u) n'est pas fini.
  - on va appeler `PP-Visiter`(G, v') pour tout $(v, v') \in A$ t.q. $\mathrm{Couleur}[v'] = \mathrm{blanc}$.
  - puis colorier v en noir ... et affecter $\mathrm{f}[v]$,

Propriétés du Parc. en Profondeur - 1

Propriété 16 Pour tout sommet u et v, on a :

- soit les intervalles $[\mathrm{d}[u]; \mathrm{f}[u]]$ et $[\mathrm{d}[v]; \mathrm{f}[v]]$ sont disjoints ;
- soit $[\mathrm{d}[u]; \mathrm{f}[u]]$ est contenu dans $[\mathrm{d}[v]; \mathrm{f}[v]]$;
- soit $[\mathrm{d}[v]; \mathrm{f}[v]]$ est contenu dans $[\mathrm{d}[u]; \mathrm{f}[u]]$.

Supposons $\mathrm{d}[u] < \mathrm{d}[v]$.

- $\mathrm{d}[v] < \mathrm{f}[u]$:
  - En $\mathrm{d}[v]$, u est gris : `PP-Visiter`(G, u) n'est pas fini.
  - 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 $\mathrm{f}[v]$,
  - finir les `PP-Visiter`(G, u') des autres descendants de u,

Propriétés du Parc. en Profondeur - 1

Propriété 16 Pour tout sommet u et v, on a :

- soit les intervalles $[\mathrm{d}[u]; \mathrm{f}[u]]$ et $[\mathrm{d}[v]; \mathrm{f}[v]]$ sont disjoints ;
- soit $[\mathrm{d}[u]; \mathrm{f}[u]]$ est contenu dans $[\mathrm{d}[v]; \mathrm{f}[v]]$;
- soit $[\mathrm{d}[v]; \mathrm{f}[v]]$ est contenu dans $[\mathrm{d}[u]; \mathrm{f}[u]]$.

Supposons $\mathrm{d}[u] < \mathrm{d}[v]$.

- $\mathrm{d}[v] < \mathrm{f}[u]$:
  - En $\mathrm{d}[v]$, u est gris : `PP-Visiter`(G, u) n'est pas fini.
  - on va appeler `PP-Visiter`(G, v') pour tout $(v, v') \in A$ t.q. $\mathrm{Couleur}[v'] = \mathrm{blanc}$.
  - puis colorier v en noir ... et affecter $\mathrm{f}[v]$,
  - finir les `PP-Visiter`(G, u') des autres descendants de u,
  - et enfin affecter $\mathrm{f}[u]$!

Propriétés du Parc. en Profondeur - 1

Propriété 16 Pour tout sommet u et v, on a :

- soit les intervalles $[d[u]; f[u]]$ et $[d[v]; f[v]]$ sont disjoints ;
- soit $[d[u]; f[u]]$ est contenu dans $[d[v]; f[v]]$;
- soit $[d[v]; f[v]]$ est contenu dans $[d[u]; f[u]]$.

Supposons $d[u] < d[v]$.

- $d[v] < f[u]$:
  - En $d[v]$, u est gris : `PP-Visiter`(G, u) n'est pas fini.
  - 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 $f[v]$,
  - finir les `PP-Visiter`(G, u') des autres descendants de u,
  - et enfin affecter $f[u]$!
- $d[v] > f[u]$: alors $d[u] < f[u] < d[v] < f[v]$...

Propriétés du Parc. en Profondeur - 2

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

Propriétés du Parc. en Profondeur - 2

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

- (1) $\Rightarrow$ (2) $u_0 = u \rightarrow_{G_\Pi} u_1 \rightarrow_{G_\Pi} u_2 \ldots \rightarrow_{G_\Pi} u_k = v$

Propriétés du Parc. en Profondeur - 2

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

- (1) $\Rightarrow$ (2) $u_0 = u \rightarrow_{G_\Pi} u_1 \rightarrow_{G_\Pi} u_2 \ldots \rightarrow_{G_\Pi} u_k = v$
  - u_i est découvert depuis u_{i-1} : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_{i-1}]$.

Propriétés du Parc. en Profondeur - 2

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

- (1) $\Rightarrow$ (2) $u_0 = u \to_{G_\Pi} u_1 \to_{G_\Pi} u_2 \ldots \to_{G_\Pi} u_k = v$
  - u_i est découvert depuis u_{i-1} : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_{i-1}]$.
  - par la Prop. 16 : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_i] < \mathrm{f}[u_{i-1}]$, ...

Propriétés du Parc. en Profondeur - 2

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

- (1) $\Rightarrow$ (2) $u_0 = u \rightarrow_{G_\Pi} u_1 \rightarrow_{G_\Pi} u_2 \ldots \rightarrow_{G_\Pi} u_k = v$
  - u_i est découvert depuis u_{i-1} : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_{i-1}]$.
  - par la Prop. 16 : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_i] < \mathrm{f}[u_{i-1}]$, ...
- (2) $\Rightarrow$ (1) soient u et v tq $\mathrm{d}[u] < \mathrm{d}[v] < \mathrm{f}[v] < \mathrm{f}[u]$ et $u \not\rightarrow^*_{G_\Pi} v$. On choisit v de manière à minimiser $\mathrm{d}[v]$.

Propriétés du Parc. en Profondeur - 2

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

- (1) $\Rightarrow$ (2) $u_0 = u \rightarrow_{G_\Pi} u_1 \rightarrow_{G_\Pi} u_2 \ldots \rightarrow_{G_\Pi} u_k = v$
  - u_i est découvert depuis u_{i-1} : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_{i-1}]$.
  - par la Prop. 16 : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_i] < \mathrm{f}[u_{i-1}]$, ...
- (2) $\Rightarrow$ (1) soient u et v tq $\mathrm{d}[u] < \mathrm{d}[v] < \mathrm{f}[v] < \mathrm{f}[u]$ et $u \not\rightarrow^*_{G_\Pi} v$. On choisit v de manière à minimiser $\mathrm{d}[v]$.

Propriétés du Parc. en Profondeur - 2

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

- (1) $\Rightarrow$ (2) $u_0 = u \rightarrow_{G_\Pi} u_1 \rightarrow_{G_\Pi} u_2 \ldots \rightarrow_{G_\Pi} u_k = v$
  - u_i est découvert depuis u_{i-1} : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_{i-1}]$.
  - par la Prop. 16 : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_i] < \mathrm{f}[u_{i-1}]$, ...
- (2) $\Rightarrow$ (1) soient u et v tq $\mathrm{d}[u] < \mathrm{d}[v] < \mathrm{f}[v] < \mathrm{f}[u]$ et $u \not\rightarrow^*_{G_\Pi} v$. On choisit v de manière à minimiser $\mathrm{d}[v]$.
  Considérons $w = \Pi(v)$ (il existe car $\mathrm{d}[u] < \mathrm{d}[v] < \mathrm{f}[u]$!)
  - $\mathrm{d}[u] < \mathrm{d}[w]$?

Propriétés du Parc. en Profondeur - 2

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

- (1) $\Rightarrow$ (2) $u_0 = u \rightarrow_{G_\Pi} u_1 \rightarrow_{G_\Pi} u_2 \ldots \rightarrow_{G_\Pi} u_k = v$
  - u_i est découvert depuis u_{i-1} : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_{i-1}]$.
  - par la Prop. 16 : $\mathrm{d}[u_{i-1}] < \mathrm{d}[u_i] < \mathrm{f}[u_i] < \mathrm{f}[u_{i-1}]$, ...
- (2) $\Rightarrow$ (1) soient u et v tq $\mathrm{d}[u] < \mathrm{d}[v] < \mathrm{f}[v] < \mathrm{f}[u]$ et $u \not\rightarrow^*_{G_\Pi} v$. On choisit v de manière à minimiser $\mathrm{d}[v]$.
  Considérons $w = \Pi(v)$ (il existe car $\mathrm{d}[u] < \mathrm{d}[v] < \mathrm{f}[u]$!)
  - $\mathrm{d}[u] < \mathrm{d}[w]$? Alors $\mathrm{d}[u] < \mathrm{d}[w] < \mathrm{f}[u]$ (car $\mathrm{d}[w] < \mathrm{d}[v]$)
    Prop. 16 $\Rightarrow$ $\mathrm{d}[u] < \mathrm{d}[w] < \mathrm{f}[w] < \mathrm{f}[u]$ et donc $u \rightarrow^*_{G_\Pi} w$ car v a été « bien » choisi... Donc $u \rightarrow^*_{G_\Pi} v$, contradiction !
  - $\mathrm{d}[w] < \mathrm{d}[u]$?

Propriétés du Parc. en Profondeur - 2

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

- (1) $\Rightarrow$ (2) $u_0 = u \rightarrow_{G_\Pi} u_1 \rightarrow_{G_\Pi} u_2 \ldots \rightarrow_{G_\Pi} u_k = v$
  - u_i est découvert depuis u_{i-1} : $d[u_{i-1}] < d[u_i] < f[u_{i-1}]$.
  - par la Prop. 16 : $d[u_{i-1}] < d[u_i] < f[u_i] < f[u_{i-1}]$, ...
- (2) $\Rightarrow$ (1) soient u et v tq $d[u] < d[v] < f[v] < f[u]$ et $u \not\rightarrow^*_{G_\Pi} v$. On choisit v de manière à minimiser $d[v]$.
  Considérons $w = \Pi(v)$ (il existe car $d[u] < d[v] < f[u]$!)
  - $d[u] < d[w]$? Alors $d[u] < d[w] < f[u]$ (car $d[w] < d[v]$)
    Prop. 16 $\Rightarrow$ $d[u] < d[w] < f[w] < f[u]$ et donc $u \rightarrow^*_{G_\Pi} w$ car v a été « bien » choisi... Donc $u \rightarrow^*_{G_\Pi} v$, contradiction !
  - $d[w] < d[u]$? u est découvert par un `PP-Visiter`(G, w')...
    avant l'appel `PP-Visiter`(G, v) qui découvrira v.
    Donc en $d[v]$, `PP-Visiter`(G, u) est terminé et
    $d[u] < f[u] < d[v] < f[v]$, contradiction !

Propriétés des arcs

Un 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

Un 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 $\mathrm{d}[w] < \mathrm{d}[v] < \mathrm{f}[v] < \mathrm{f}[w]$; et
2. arc "avant" ssi $\mathrm{d}[v] < \mathrm{d}[w] < \mathrm{f}[w] < \mathrm{f}[v] \ \wedge\ \Pi[w] \neq v$;
3. arc "transverse" ssi $\mathrm{d}[w] < \mathrm{f}[w] < \mathrm{d}[v] < \mathrm{f}[v]$.

Propriétés des arcs

Un 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 $\mathrm{d}[w] < \mathrm{d}[v] < \mathrm{f}[v] < \mathrm{f}[w]$; et
2. arc "avant" ssi $\mathrm{d}[v] < \mathrm{d}[w] < \mathrm{f}[w] < \mathrm{f}[v] \ \wedge \ \Pi[w] \neq v$;
3. arc "transverse" ssi $\mathrm{d}[w] < \mathrm{f}[w] < \mathrm{d}[v] < \mathrm{f}[v]$.

- (1) et (2) : Propriété 17...

Propriétés des arcs

Un 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 $d[w] < d[v] < f[v] < f[w]$; et
2. arc "avant" ssi $d[v] < d[w] < f[w] < f[v] \ \wedge \ \Pi[w] \neq v$;
3. arc "transverse" ssi $d[w] < f[w] < d[v] < f[v]$.

- (1) et (2) : Propriété 17...
- (3) on a soit $d[v] < f[v] < d[w] < f[w]$ ou $d[w] < f[w] < d[v] < f[v]$.
  Mais comme (v, w) n'est pas dans G_Π, w ne peut être blanc en $d[v]$.
  Donc $d[w] < f[w] < d[v] < f[v]$.

Propriétés du Parcours en Profondeur

Propriété 19 A tout moment de l'algorithme de parcours en profondeur, les sommets gris forment un chemin relié par des arcs de G_{Π}.

Propriétés du Parcours en Profondeur

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

A tout moment de l'algo, on peut trier les sommets gris $u_1, \ldots, u_k$ par $\mathrm{d}[-]$ croissante.

$$\mathrm{d}[u_1] < \mathrm{d}[u_2] < \ldots < \mathrm{d}[u_k]$$

Propriétés du Parcours en Profondeur

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

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

$$\mathrm{d}[u_1] < \mathrm{d}[u_2] < \ldots < \mathrm{d}[u_k] < \mathrm{f}[u_k] < \ldots < \mathrm{f}[u_2] < \mathrm{f}[u_1]$$

(Prop. 16)

Propriétés du Parcours en Profondeur

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

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

$$\mathrm{d}[u_1] < \mathrm{d}[u_2] < \ldots < \mathrm{d}[u_k] < \mathrm{f}[u_k] < \ldots < \mathrm{f}[u_2] < \mathrm{f}[u_1]$$

(Prop. 16)

Donc : $u_1 \to^*_{G_\Pi} u_2 \to^*_{G_\Pi} \ldots \to^*_{G_\Pi} u_k$ (Prop. 17)

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

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

$$\mathrm{d}[u_1] < \mathrm{d}[u_2] < \ldots < \mathrm{d}[u_k] < \mathrm{f}[u_k] < \ldots < \mathrm{f}[u_2] < \mathrm{f}[u_1]$$

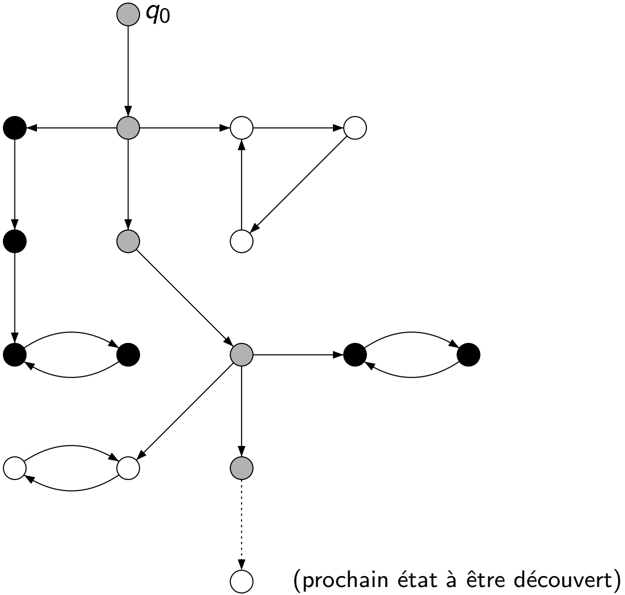
(Prop. 16)

Donc : $u_1 \rightarrow^*_{G_\Pi} u_2 \rightarrow^*_{G_\Pi} \ldots \rightarrow^*_{G_\Pi} u_k$ (Prop. 17)

Chaque étape de ce chemin est élémentaire car il n'y a ...

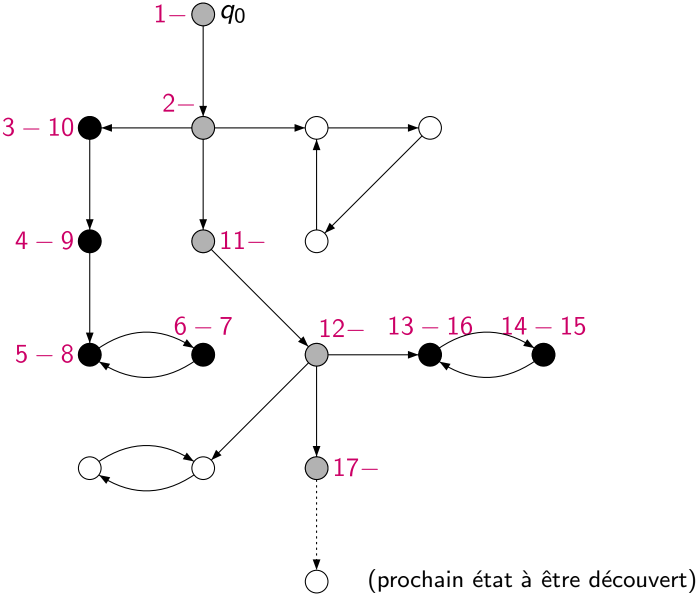
1. ni sommet blanc (il faut que $\Pi[-]$ soit def.),
2. ni sommet noir (ils n'ont pas de sommets gris comme descendants dans G_Π).

Vision globale du graphe



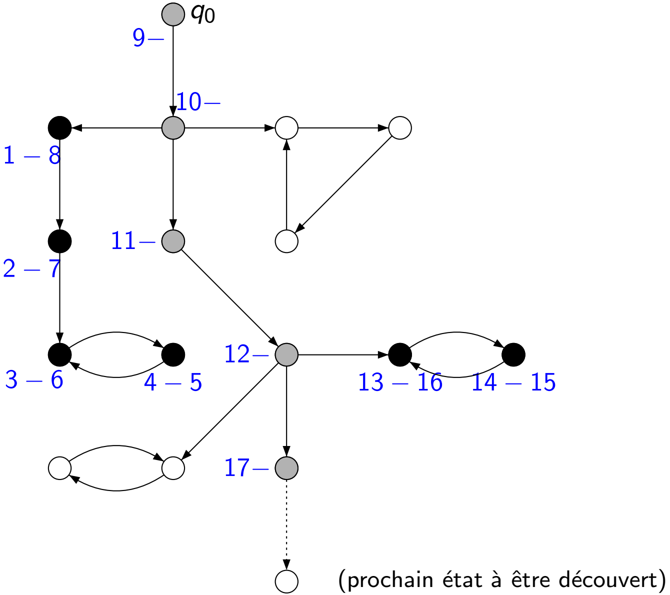
Exercice : Comment en est-on arrivé là ?

Vision globale du graphe



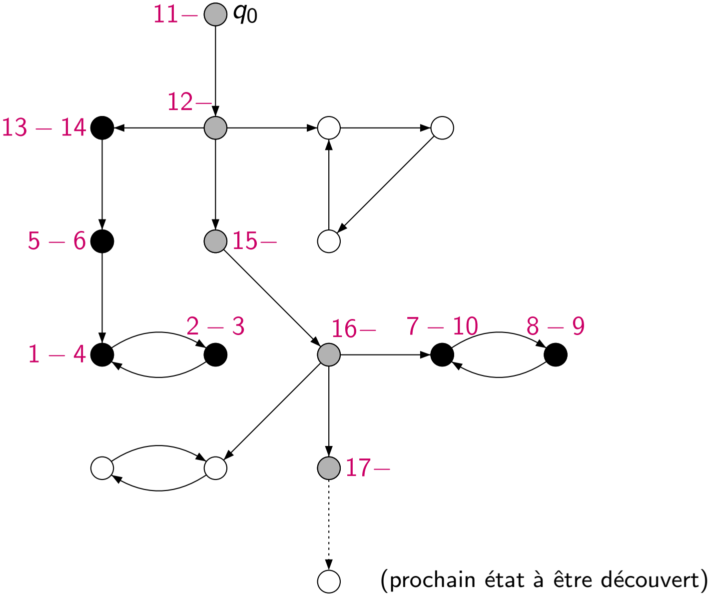
Exercice : Comment en est-on arrivé là ?

Vision globale du graphe



Exercice : Comment en est-on arrivé là ?

Vision globale du graphe



Exercice : Comment en est-on arrivé là ?

Propriétés du Parcours en Profondeur

Énoncer la correction de `PP-Visiter`(G, s) ?

Propriétés du Parcours en Profondeur

Énoncer la correction de `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 descendants de s dans G_Π...

Propriétés du Parcours en Profondeur

Énoncer la correction de `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 descendants de s dans G_Π...

Théorème du chemin blanc

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

Preuve de la correction du PP

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

Preuve de la correction du PP

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

(1) $\Rightarrow$ (2) tout sommet w le long du chemin $u \rightarrow^*_{G_\Pi} v$ vérifie d$[u] <$ d$[w]$ (Prop. 17) : donc chaque w est blanc en d$[u]$.

Preuve de la correction du PP

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

(1) $\Rightarrow$ (2) tout sommet w le long du chemin $u \to^*_{G_\Pi} v$ vérifie d$[u] <$ d$[w]$ (Prop. 17) : donc chaque w est blanc en d$[u]$.

(2) $\Rightarrow$ (1) soit v atteignable par un chemin blanc ρ depuis u en d$[u]$ tq $u \not\to^*_{G_\Pi} v$. On prend le premier v de ce type le long de ρ.

Preuve de la correction du PP

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

(1) $\Rightarrow$ (2) tout sommet w le long du chemin $u \rightarrow^*_{G_\Pi} v$ vérifie d$[u] <$ d$[w]$ (Prop. 17) : donc chaque w est blanc en d$[u]$.

(2) $\Rightarrow$ (1) soit v atteignable par un chemin blanc ρ depuis u en d$[u]$ tq $u \not\rightarrow^*_{G_\Pi} v$. On prend le premier v de ce type le long de ρ. Pour tout prédécesseur w de v sur ρ, on a : $u \rightarrow^*_{G_\Pi} w$.

Preuve de la correction du PP

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

(1) $\Rightarrow$ (2) tout sommet w le long du chemin $u \rightarrow^*_{G_\Pi} v$ vérifie $d[u] < d[w]$ (Prop. 17) : donc chaque w est blanc en $d[u]$.

(2) $\Rightarrow$ (1) soit v atteignable par un chemin blanc ρ depuis u en $d[u]$ tq $u \not\rightarrow^*_{G_\Pi} v$. On prend le premier v de ce type le long de ρ.
Pour tout prédécesseur w de v sur ρ, on a : $u \rightarrow^*_{G_\Pi} w$.
Donc $d[u] < d[w] < f[w] < f[u]$ (Prop. 17)

Preuve de la correction du PP

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

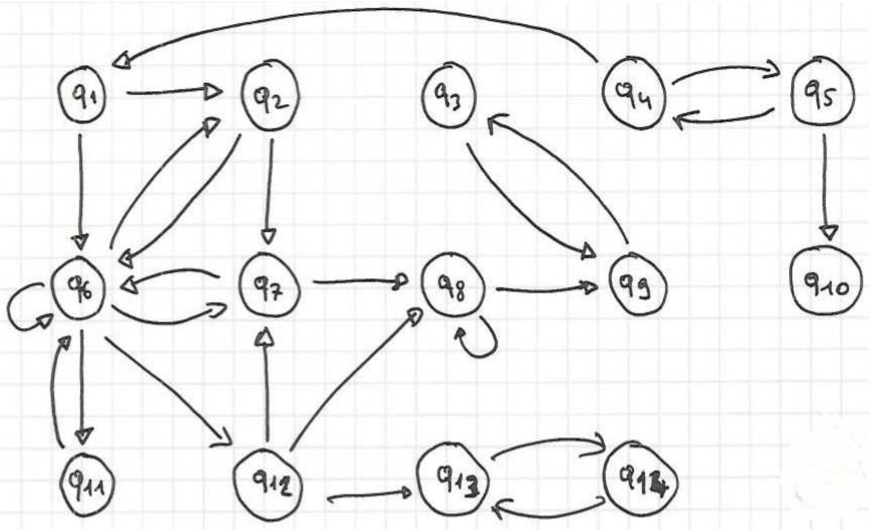
(1) $\Rightarrow$ (2) tout sommet w le long du chemin $u \rightarrow^*_{G_\Pi} v$ vérifie d$[u] <$ d$[w]$ (Prop. 17) : donc chaque w est blanc en d$[u]$.

(2) $\Rightarrow$ (1) soit v atteignable par un chemin blanc ρ depuis u en d$[u]$ tq $u \not\rightarrow^*_{G_\Pi} v$. On prend le premier v de ce type le long de ρ.
Pour tout prédécesseur w de v sur ρ, on a : $u \rightarrow^*_{G_\Pi} w$.
Donc d$[u] <$ d$[w] <$ f$[w] <$ f$[u]$ (Prop. 17)
Considérons le premier (sur ρ) de ces w tq $\exists(w, v) \in A$.
Comme $w \not\rightarrow^*_{G_\Pi} v$, v doit être gris avant f$[w]$: d$[v] <$ f$[w]$.
D'après la Prop. 16, on a 2 cas :

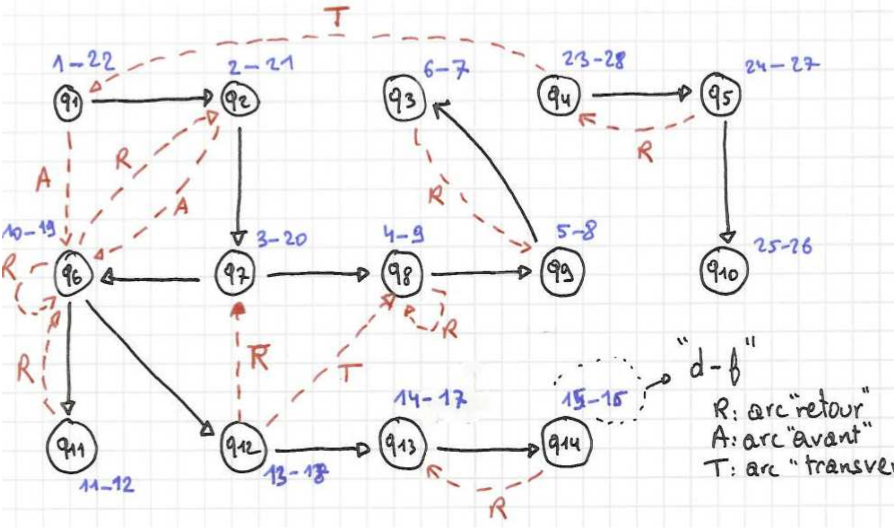
$$\mathrm{d}[w] < \mathrm{d}[v] < \mathrm{f}[v] < \mathrm{f}[w] \quad \textbf{ou} \quad \mathrm{d}[v] < \mathrm{f}[v] < \mathrm{d}[w] < \mathrm{f}[w]$$

D'où $[\mathrm{d}[v]; \mathrm{f}[v]] \subset [\mathrm{d}[u]; \mathrm{f}[u]]$ et par la Prop. 17, on a $u \rightarrow^*_{G_\Pi} v$.
Contradiction !

Exemple

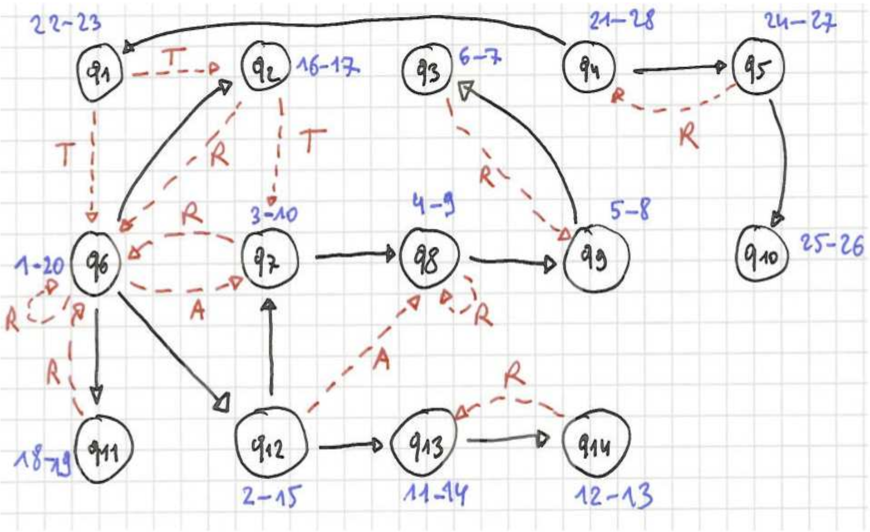


Exemple



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

Exemple



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

Plan

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 : $(u, v) \in A \ \Rightarrow \ u \leq v$.

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

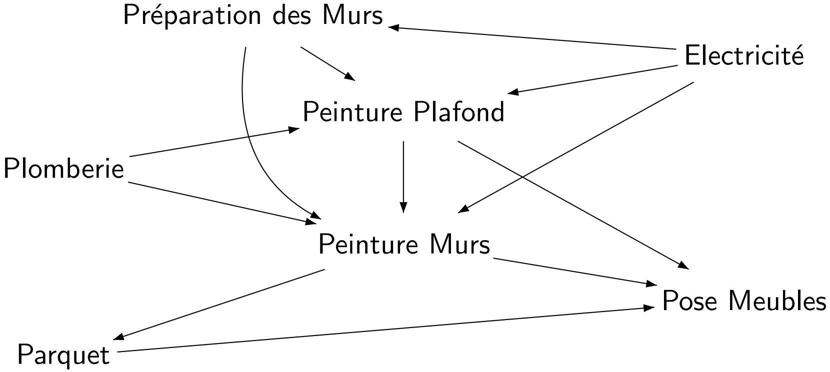
Q ? comment en déduire un ordre total ?

Applications

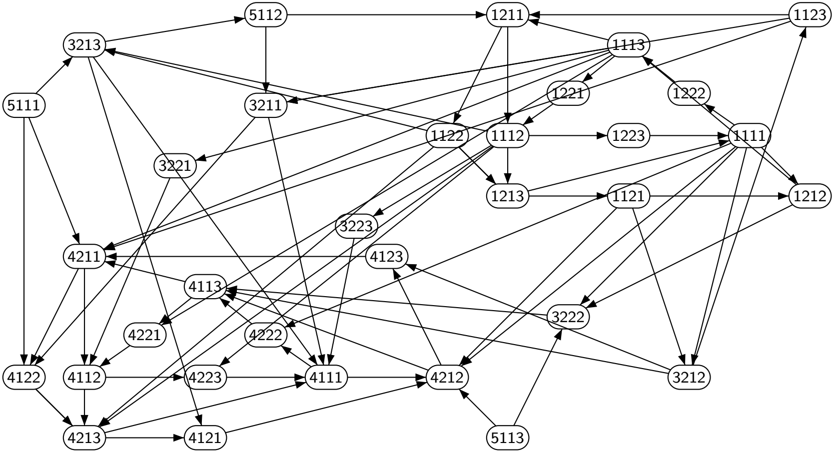
L'ordonnancement de tâches. . .
n tâches à planifier en respectant des contraintes du genre « T_i doitêtre traitée avant T_j ». . .

Applications

L'ordonnancement de tâches...
n tâches à planifier en respectant des contraintes du genre « T_i doitêtre traitée avant T_j »...



Et là ?


5112
1211
1123
3213
1113
5111
3211
1221
1222
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. et empiler chaque sommet lorsqu'il est colorié en noir

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

(Le sommet de la pile est le sommet min. pour $\leq$: il n'a pas de prédécesseur dans G.)

Algorithme de recherche d'extension linéaire

1. Appeler PP(G)...
2. et empiler chaque sommet lorsqu'il est colorié en noir

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

(Le sommet de la pile est le sommet min. pour $\leq$: il n'a pas de prédécesseur dans G.)

ou :

1. Appeler PP(G)...
2. Trier les sommets par date f[.] décroissantes.

L'ordre $\leq_A$ calculé est donc : $u \leq_A v$ ssi $f[u] > f[v]$

(les deux définitions Pile/f[ ] sont équivalentes !)

Algo. de recherche d'extension linéaire

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

Avec :

Procédure `PP-Visiter2`(G, s)
Couleur$[s]$:= gris;
pour chaque $(s, u) \in A$ **faire**
  si $\mathit{Couleur}[u] = \mathit{blanc}$ **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 linéairede G.

Correction de l'algorithme

Théorème

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

D'abord, on montre :

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 linéairede G.

D'abord, on montre :

Propriété 19

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

(1) $\Rightarrow$ (2) Si (u, v) est un arc retour, alors il y a un chemin gris reliant v à u, l'arc (u, v) ferme le cyle !

Correction de l'algorithme

Théorème

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

D'abord, on montre :

Propriété 19

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

(1) $\Rightarrow$ (2) Si (u, v) est un arc retour, alors il y a un chemin gris reliant v à u, l'arc (u, v) ferme le cyle !

(2) $\Rightarrow$ (1) Soit ρ un cycle. Soit v le premier sommet de ρ découvert par PP. Le théorème 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 linéairede G.

Correction de l'algorithme

Théorème

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

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

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

Correction de l'algorithme

Théorème

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

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

Obj : il faut 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 depuis 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 !