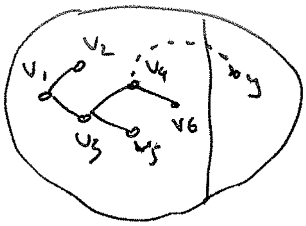
<u>Propriété</u> : pour tout graphe connexe G et tout sommet v de G, il existe un parcours de G de racine v.

• <u>Pr</u> : de façon incrémentale : si on a déjà calculé un parcours partiel $v_1 = v, \dots, v_i$ de G.



Comme G est connexe, il existe une arête xy entre $\{x_1, \dots, x_i\}$ et $\overline{\{x_1, \dots, x_i\}}$ avec $x = v_j$ et $y \notin \{x_1 \dots x_i\}$

on pose $v_{i+1} = y$ et $\text{père}(v_{i+1}) = v_k$

II) <u>Un algorithme de parcours.</u>

On détaille l'algorithme précédent.

<u>ENTRÉE</u> : $G = (V, E)$ connexe et r un sommet de G (racine). G par liste de voisins

<u>SORTIE</u> : L'énumération $(v_1 \dots v_n)$ et la fonction père.

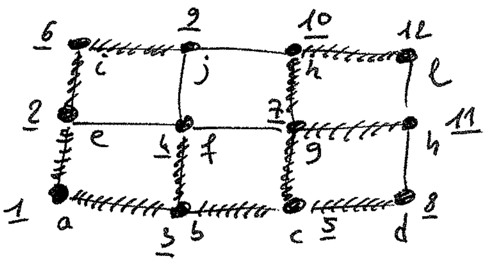
<u>PARCOURS(G, r)</u>

```
Initialisation
  Pour tout v ∈ V, dv(v) ← 0 ;      // sommets déjà vus.
  dv(r) ← 1 ;
  ordre(r) ← 1 ;    // le numéro de r dans (v1, ..., vn).
  t ← 2 ;           // pour coder les prochains numéros.
  père(r) ← r ;
  AT ← {r} ;  // les sommets que l'on doit traiter.
Tant que AT ≠ ∅
    Choisir v ∈ AT ;
    AT ← AT \ {v}
    Pour x ∈ Vois(v)
        Si dv(x) = 0 alors
```

L12 | $dv(x) \leftarrow 1$;
L13 | $AT \leftarrow AT \cup \{x\}$
L14 | $ordre(x) \leftarrow t$;
L15 | $t \leftarrow t+1$;
L16 | $père(x) \leftarrow v$;

L17 | Retourner ordre et père ;

Exemple : On note les valeurs de "ordre" et on marque les "déjà vu"



AT : ~~a e b f c i g d j k h l~~

Analyse de l'algo :

Correction : découle de la propriété I.

Terminaison / complexité :

Initialisation : $O(n)$.

L_7 : chaque sommet apparaît au plus une fois dans AT (grâce au marqueur dv). $O(n)$

L_{10} : chaque arête du graphe est examinée 2 fois (1 fois par extrémité) donc L_{10} à $L_{16} \rightarrow O(m)$.

En tout $O(m+n)$

Rem : . Si on teste en fin d'algo si $t = n+1$ alors on a un algo qui décide si G est connexe en $O(n+m)$.

. On va voir des parcours particuliers.

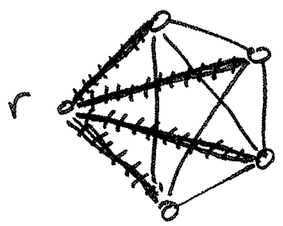
III). Arbre de plus courts chemins

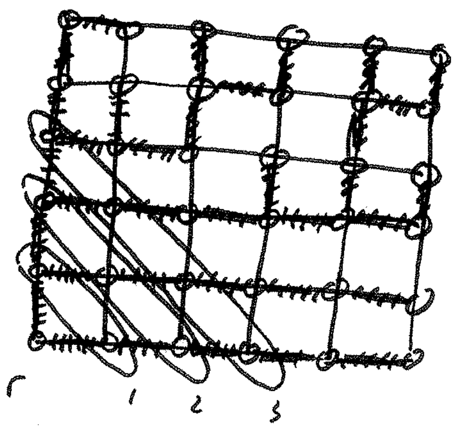
- But : trouver un plus court chemin entre un sommet r et un sommet x (on doit parfois lire tout le graphe).

- Soit G connexe et $r \in V$ (racine), un arbre de plus court chemin (apcc) de racine r est un arbre couvrant $T=(V,A)$ de G tel que pour tout $x \in V$ on ait : $d_T(r,x) = d_G(r,x)$ (les distances sont égales dans T et G).

Ce qui signifie que l'unique chemin de r à x dans T est un chemin de longueur min dans G.

Ex :

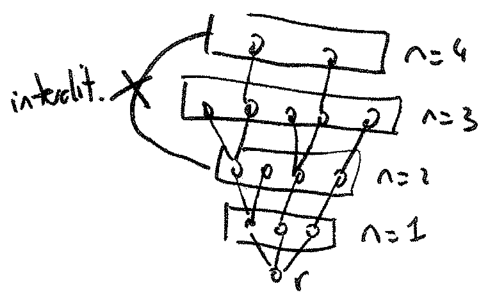




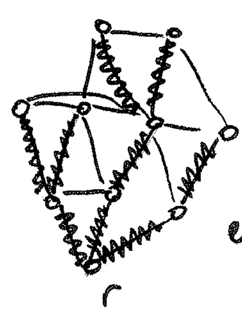
On traite "tout un niveau" avant de passer au suivant.
On va obtenir ça en se servant d'une file lors d'un parcours.

- Dans un arbre enraciné T et de racine r, $d_T(r,x)$ est appelé niveau du sommet x dans T et est noté $n_T(x)$.

Propriété : Un arbre est un apcc ssi $\forall xy \in E(G)$ $|n_T(x) - n_T(y)| \leq 1$ (P)



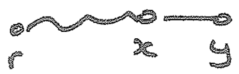
On a :



les arêtes sont soit dans un même niveau, soit entre 2 niveaux consécutifs.

Preuve : • puisque xy est une arête de G, on a :

$$d_G(r,y) \leq d_G(r,x)+1$$



Comme T apcc, on a : $n_T(y) = d_G(r,y) \leq d_G(r,x)+1 = n_T(x)+1$.

Symétriquement : $n_T(x) \leq n_T(y)+1$.

• Réciproquement si T vérifie (?) alors pour un sommet x de G, il existe un chemin P de r à x de longueur $n_T(x)$ (dans l'arbre) et tout autre chemin : $r = x_1 x_2 \dots x_k = x$ vérifie :

$$(n_T(x_2)-n_T(x_1)) + (n_T(x_3)-n_T(x_2)) + \dots + (n_T(x_k) - n_T(x_{k-1})) \leq k$$

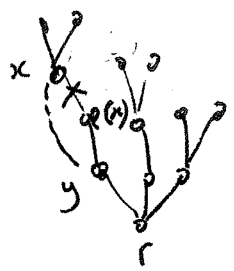
$$\Rightarrow n_T(x_k) - n_T(x_1) \leq k$$

$$\Rightarrow n_T(x) \leq k$$

Donc P est un plus court chemin de r à x. ■

• Lemme : tout graphe connexe admet un ap.c.c.

Pr : on choisit un arbre T avec $\sum_x n_T(x)$ minimum.



Si une arête xy vérifie $n_T(x) \geq n_T(y)+2$

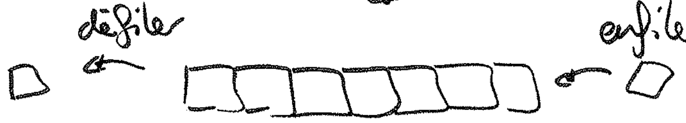
On change T : $T \leftarrow (T \setminus \{x\, \text{père}(x)\}) \cup \{xy\}$

$n_T(x)$ chute de 1 au moins (ainsi que ses successeurs). ■

• Rq : Ça donne un algo pour construire un a.p.c.c. (mais pas bon !)

IV) Calcul effectif d'un a.p.c.c : Parcours en largeur.

• On va utiliser une file (comme à la boulangerie) Premier Entré, premier sorti (FIFO).



On reprend l'algo de parcours et on gère AT comme une file :

L_6 : Enfiler r sur AT

L_8 : | Défiler AT, on note v le sommet défilé

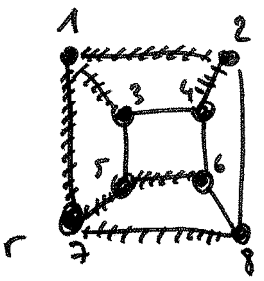
L_9 : |

L_{13} : Enfiler x sur AT.

Pour calculer le niveau, on rajoute :

L_6' : $n_T(r) \leftarrow 0$ et L_{16}' : $n_T(x) \leftarrow n_T(v) + 1$.

. <u>Exemple</u> :



AT : ~~7~~ ~~1~~ ~~8~~ ~~8~~ ~~2~~ 3 6

	1	2	3	4	5	6	7	8
père	7	1	1	2	7	5	7	7
ordre	2	5	6	8	3	7	1	4
niveau	1	2	2	3	1	2	0	1

. <u>Analyse</u> :

. <u>Terminaison / Complexité</u> : L'usage de la pile ne change rien comme PARCOURS. $O(m+n)$

. <u>Correction</u> : On montre que l'arbre de parcours est un a.p.c.c. :

. <u>Fait 1</u> : $\text{ordre}(x) < \text{ordre}(y) \Rightarrow \text{ordre}(\text{père}(x)) \leq \text{ordre}(\text{père}(y))$.

Sinon, $\text{ordre}(\text{père}(y)) < \text{ordre}(\text{père}(x))$, mais lorsqu'on