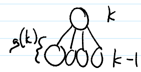
Algos de branchement

Technique de design d'algo FPT qui consiste à brancher sur un # limité de cas possibles.
Pour avoir un algo FPT, il faut que:



- le # d'appels récursifs est borné par une fct $g(k)$
- le param k doit réduire à chaque appel récursif
  $\Rightarrow$ hauteur de l'arbre bornée par une fct $h(k)$

Avec ces 2 conditions, le # de noeuds est $O(g(k)^{h(k)})$

- s'il y a une solution, un des noeuds de l'arbre de récursion la trouve

Vertex-cover
$g(k) = 2$
$h(k) = k$
$\Rightarrow O(2^k)$

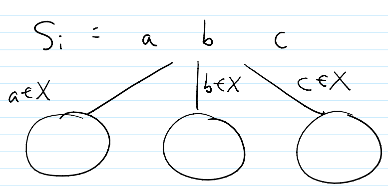
3-HITTING SET

Entrée: ensembles $S = \{S_1, S_2, \dots, S_m\}$ de taille 3 chacun
univers $U = \{u_1, \dots, u_n\}$
Param: k = taille de solution voulue
Sortie: $X \subseteq U$ (univers) tel que $|X| \leq k$
et tel que $\forall S_i$, $X \cap S_i \neq \emptyset$, ou null si un tel X n'existe pas

S_1	S_2	S_3	S_4	
abc	ade	dfg	deh	$k=2$

a d

On prend $S_i \in S$ arbitraire

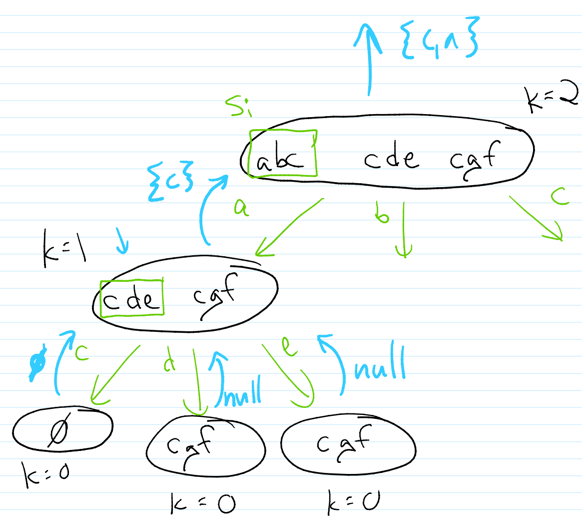


Toute solution X doit contenir a ou b ou c

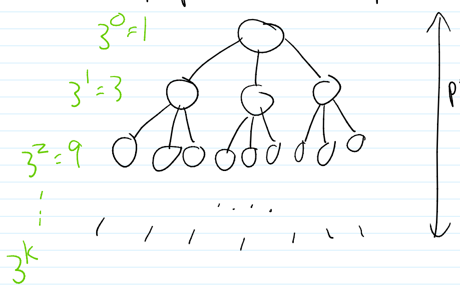
```
3hitset( S = {S1, ..., Sm}, k )
    si S = ∅, return ∅
    si k = 0, return null

    Si = ensemble de S
    Soit Si = {a,b,c}
    Sa = S \ { Sj ∈ S : a ∈ Sj }
    Xa = 3hitset( Sa, k-1 )
    Sb = S \ { Sj ∈ S : b ∈ Sj }
    Xb = 3hitset( Sb, k-1 )
    Sc = S \ { Sj ∈ S : c ∈ Sj }
    Xc = 3hitset( Sc, k-1 )

    si Xa = Xb = Xc = null, return null
    si Xa ≠ null, return Xa ∪ {a}
    si Xb ≠ null, return Xb ∪ {b}
    return Xc ∪ {c}
```



Cet algo crée un arbre récursif où chaque noeud a 3 enfants, Cet algo crée un arbre récursif où chaque noeud a 3 enfants, et sa prof. est bornée par k

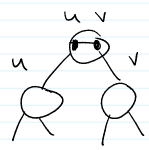


$O(3^k)$ noeuds

Chaque appel prend un temps $O(|S|)=O(m)$

$\Rightarrow$ Complexité: $O(3^k \cdot m)$

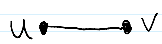
Vertex-Cover



$O(2^k)$

Autre idée de branchement $\rightarrow O(1{,}618^k)$

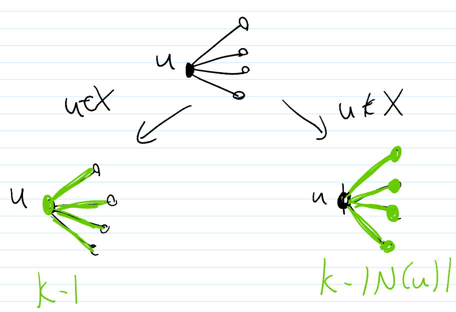
Soit $uv \in E$.



2 cas possibles pour une solution X

- $u \in X$ $\quad k-1$
- $u \notin X$ $\quad k$

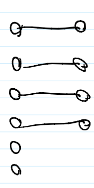
  └ si $u \notin X$, alors tous ses voisins sont dans X



- Pour réduire le + possible, on choisit u de degré max

- On voudrait que $k-|N(u)|$ soit mieux que $k-1$
- S'il $\exists u \in V$ t.q. $|N(u)| \ge 2$, on n'a pas ce pb.
- Il y a un pb si $|N(u)| \le 1 \quad \forall u \in V$

  Si c'est le cas, vertex-cover est facile car G est



Dans ce cas, le vertex-cover est $|E|$

vcFPT2 $(G=(V,E), k)$

si $|E|=0$, return $\emptyset$

vcFPT2 ($G=(V,E)$, k)

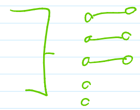
si $|E|=0$, return $\emptyset$

si $k=0$, return null

si $|N(u)| \leq 1 \ \forall u \in V$

 si $|E|>k$, return null

 return {un sommet par arête }



// à partir d'ici, $\exists u$ t.q. $|N(u)| \geq 2$

Soit u de degré maximum

$G_u = G - u$

$X_u = \text{vcFPT2}(G_u, k-1)$

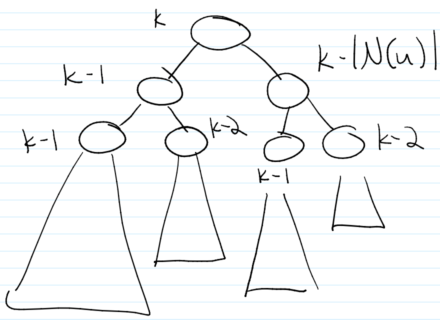
$G_{Nu} = G - N(u)$

$X_{Nu} = \text{vcFPT2}(G_{Nu}, k-|N(u)|)$

si $X_u \neq$ null, return $X_u \cup \{u\}$

si $X_{Nu} \neq$ null, return $X_{Nu} \cup N(u)$

return null



on peut supposer que $k-|N(u)| \leq k-2$ car $|N(u)| \geq 2$

Soit $f(k)$ le # de noeuds de l'arbre récursif

On a

$$f(k) = f(k-1) + f(k-2)$$

Ceci donne que le # de noeuds est $O(1.619^k)$

On ne sait pas $f(k)$ c'est quoi, mais c'est "sûrement" exponentiel.

$f(k) = \alpha^k$ on cherche le α

On a

$$f(k) = f(k-1) + f(k-2)$$

$$\alpha^k = \alpha^{k-1} + \alpha^{k-2} \qquad / \alpha^{k-2}$$

$$\alpha^2 = \alpha^1 + 1$$

$$\alpha^2 - \alpha^1 - 1 = 0$$

$$\alpha \in \{-0.61803..., \underline{1.618...}\}$$

$$\left(f(k) = c_1 \cdot 1.618^k + c_2 \cdot (-0.6)^k \right)$$

$\alpha \in \{-0{,}61803..., \underline{1.618...}\}$ $\quad$ $\left(f(k) = c_1 \cdot 1{,}618^{...} + c_2 \cdot (-0{,}6) \right.$

$\hookrightarrow \alpha = 1{,}618... \leq 1.619$

$\Rightarrow f(k) = 1{,}619^k$

$\Rightarrow \#\text{noeuds} \in O(1{,}619^k)$

Recette de résolution de récurrences homogènes linéaires

Forme: $f(k) = b_1 f(k-1) + b_2 f(k-2) + \dots + b_h f(k-h) \quad h \in O(1)$

① Poser $f(k) = \alpha^k$

② Remplacer $f(k)$ et les $f(k-i)$

$$\alpha^k = b_1 \alpha^{k-1} + b_2 \alpha^{k-2} + \dots + b_h \alpha^{k-h}$$

③ Tout mettre à gauche

$$\alpha^k - b_1 \alpha^{k-1} - b_2 \alpha^{k-2} - \dots - b_h \alpha^{k-h} = 0$$

④ Diviser par α^{k-h}

$$\alpha^h - b_1 \alpha^{h-1} - b_2 \alpha^{h-2} - \dots - b_h = 0$$

⑤ Trouver les racines (les valeurs de α) (logiciel)

Il y en aura h

Soit r la + grosse racine réelle

Alors $f(k) \in O(r^k)$

ex: bidon(k)

⋮

brancher sur

3 × bidon(k-1)

2 × bidon (k-3)

4 × bidon (k-6)

$$f(k) = 3f(k-1) + 2f(k-3) + 4f(k-6)$$

$$\alpha^k = 3\alpha^{k-1} + 2\alpha^{k-3} + 4\alpha^{k-6}$$

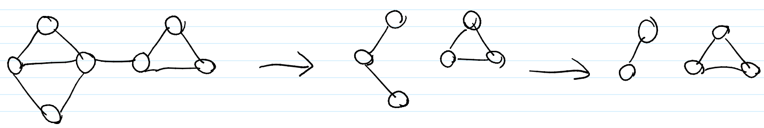
$$\alpha^k - 3\alpha^{k-1} - 2\alpha^{k-3} - 4\alpha^{k-6} = 0$$

$$\alpha^6 - 3\alpha^5 - 2\alpha^3 - 4 = 0$$

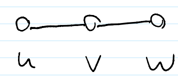
$$f(k) \in O(3.21^k)$$

CLUSTER-DELETION

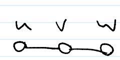
<u>Entrée</u>: graphe $G=(V,E)$
<u>Param</u>: k = # de sommets à supprimer
<u>Sortie</u>: $X \subseteq V$ tel que dans $G-X$, chaque composante (CC) connexe est une clique,
avec $|X| \leq k$,
ou null si X n'existe pas



Chaque CC est une clique $\iff \forall u,v,w \in V$
$uv \in E \wedge vw \in E \Rightarrow uw \in E$



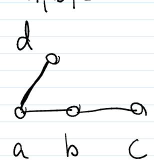
$\iff$ G n'a pas de P_3
P_3 = chemin sur 3 sommets sans raccourcis

cldel(G, k)
si G n'a aucun P_3, return $\emptyset$
si $k=0$, return null
Soit u,v,w un P_3 

Brancher sur 3 cas:
cldel$(G-u, k-1)$
cldel$(G-v, k-1)$
cldel$(G-w, k-1)$

$O(3^k \cdot n^3)$

Soit a,b,c un P_3. Soit d voisin d'un de a,b,c



Brancher sur toutes les façons d'éliminer tous les P_3 qui impliquent a,b,c,d.