Tables dynamiques avec configuration pour ajouter/supprimer

Idée :

- ▷ doubler la taille quand on add et que t est remplie
- ▷ divise par 2 la taille lorsqu'on supp un elemt et que T est à moitié.

$n = 2^k$ operations commençant par $\frac{n}{2}$ "ajouter"...

$t = nb = \frac{n}{2}$

ajouter : taille $= 2t$, nbr $= nb+1$

supprimer : taille $= 2t$ nbr $= nb$

supprimer :

ajouter :

•
•
•

Supprimer(T) :

Si (2)(*) T.nb == T.taille) Alors

T' = newTable de taille $\frac{n}{2}$

recopier T dans T'

T'.nb = T.nb

Desallouer T

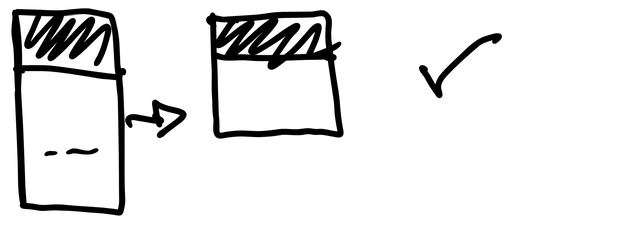
T = T''

supprimer un elemt

T.nb --

↝ bof

↳ (*) compresser la tab 4 en lien2



Potentiel :

$$\Phi(T) = \begin{cases} 2.nb - taille & \text{si } \alpha(T) \geq \frac{1}{2} \\ \frac{taille}{2} - nb & \text{si } \alpha(T) < \frac{1}{2} \end{cases}$$

$\alpha(T) = \frac{nb}{taille}$

Initialement : $nb = t = 0$ $\alpha(T) = 1$ $\Phi(T_0) = 0$

Après une expansion : $nb = \frac{t}{2}$. $\Phi(T) = 0$ [juste avant : $nb = t$, $\Phi(T) = nb$]

Après une expansion : $nb = \frac{t}{2}$ $\Phi(T) = 0$ [juste avant : $4nb = t$, $\alpha(T) = \frac{1}{4}$, $\Phi(T) = nb$]

Il reste à regarder les calculs cas par cas

Soit $nb_i, t_i, \alpha_i := \frac{nb_i}{t_i}$

Supposons que la $i^{ème}$ opération soit ... ajouter/supprimer et que α_i soit ... $< \frac{1}{2}$ ou $\geq \frac{1}{2}$

avec ou sans expansion/compression... et vérifier que $\hat{C} \in \Theta(1)$

Structure Union-Find

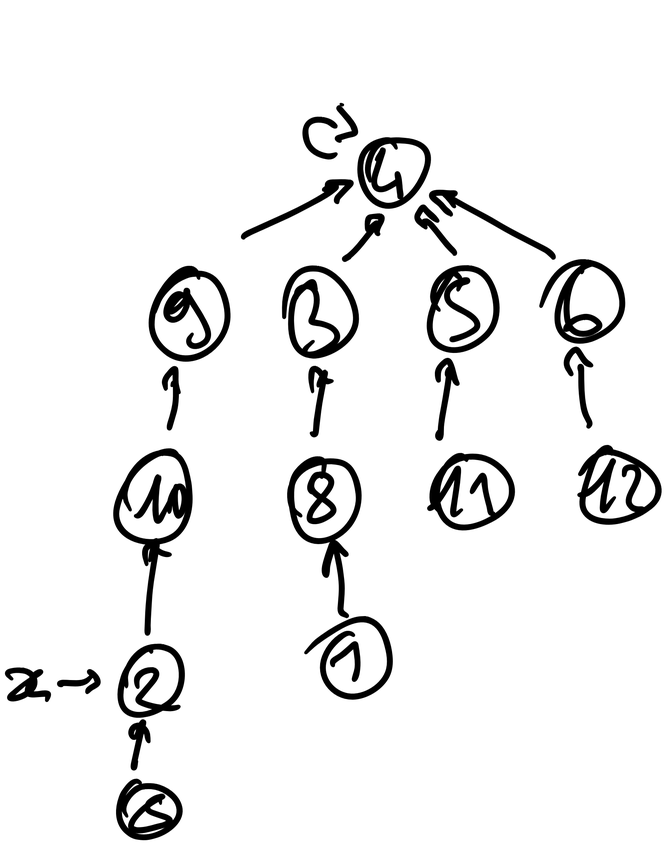
Objectifs : manipuler des partitions d'un ensemble avec deux opérations :
- Find(x) : pour trouver le sous ensemble de la partition où se trouve x.
- union(x,y) : pour fusionner les sous ensembles contenant x et y

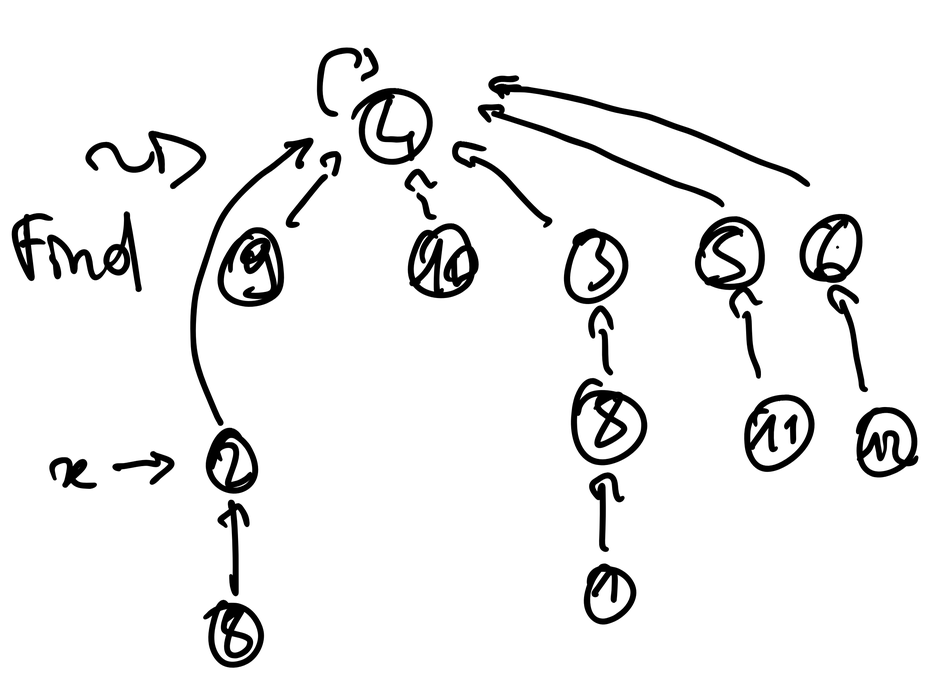
Les partitions de bases
chaque noeud a un champ (π) : [pointant vers soi | désignant son père]
un champ (r) : son "rang" [approximation de la hauteur de l'arbre de racine x]
(un champ valeur —)

MakeSet(x) : créer un arbre de singleton {x}
π(x) = x
r(x) = 0

⚠ x désigne un pointeur sur le noeud "x"
On dispose d'un pointeur sur chaque élé^t de l'ensemble

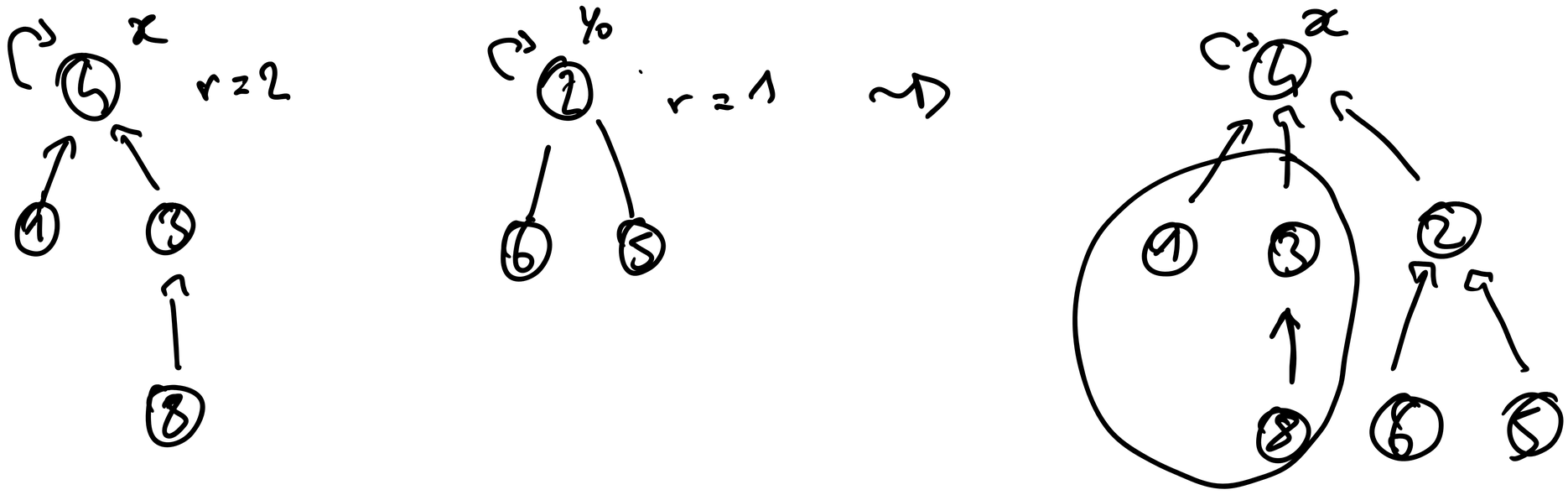
Operation Find(x) :
Exécuter à la racine du sous-arbre contenant x et "compression" le chemin...
Si (x ≠ π(x)) Alors :
π(x) = Find(π(x))
Return π(x)





Operation Union (x, y): Fusionner les deux sous ensembles --
| Link (Find (x), Find (y))

operation Link (x, y) Relier x à y ou l'inverse --)Union par niveaux!
| si r(x) > r(y) Alors π(y) = x
sinon | π(x) = y
| si r(x) == r(y) Alors r(y)++

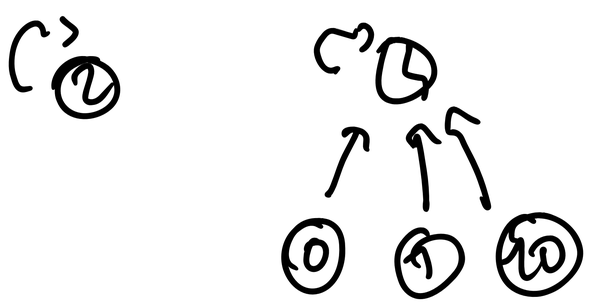


Structure :

Codage par structure arborescentes :

- un sous-ensemble de la partition = un arbre où chaque noeud pointe sur son père
  la racine est le représentant du sous ensemble (et pointe sur elle même).
- une partition = une forêt

ex : P = {{2}, {0, 1, 4, 10}, {7, 9}}



Ex : 1 2 3 4 5 6 7 8 9 10 ← rang

makeset(i) i=1..10

Union (1,3)

Union (1,4)

Union (5,6)

Union (2,7)

Union (5,7)

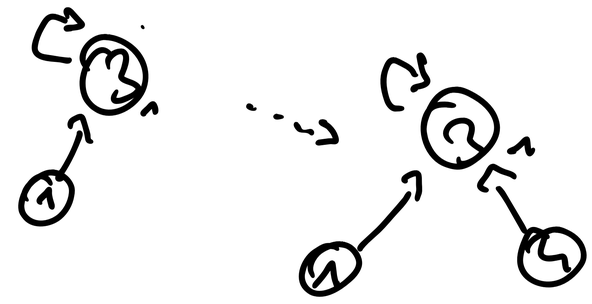
Union (1,7)

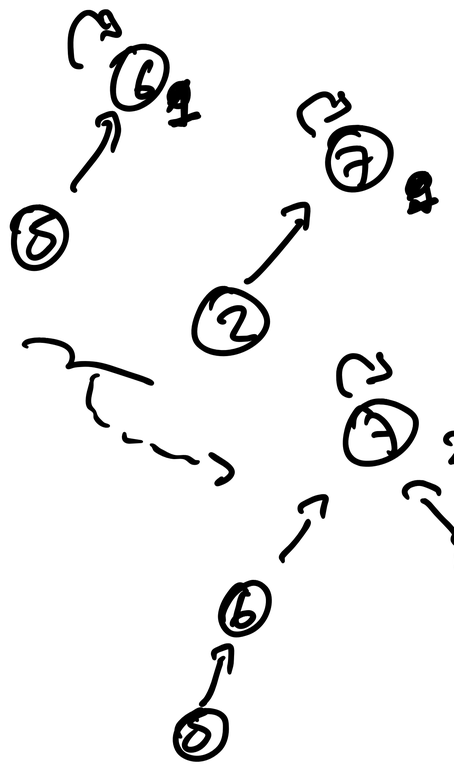
Union (8,9)

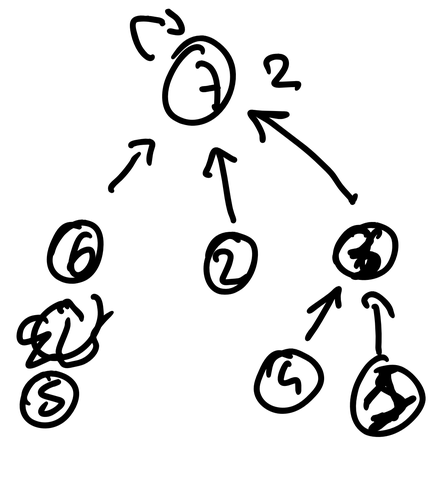
Union (10,8)

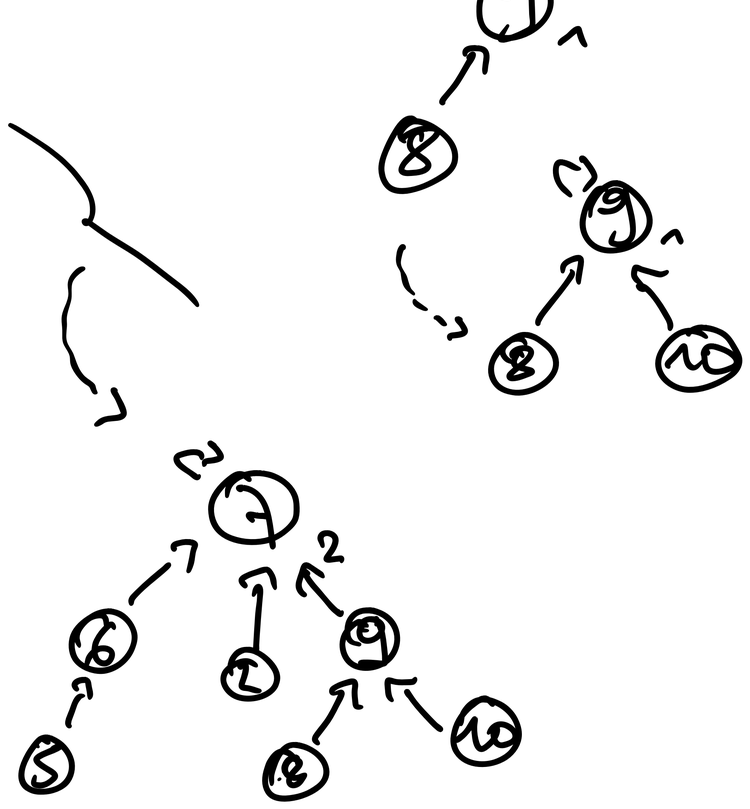
Union (4,8)

Find (5)









Complexité : | m opérations Makeset / Find / Pink dont n Makeset
| ont une complexité totale en $O(m \cdot \log^* n)$ dans le pire cas