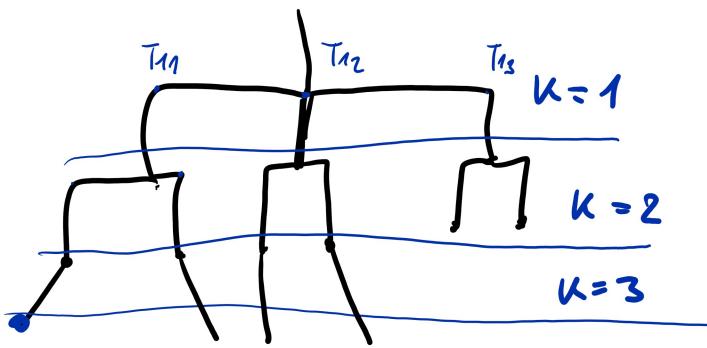


backtracking

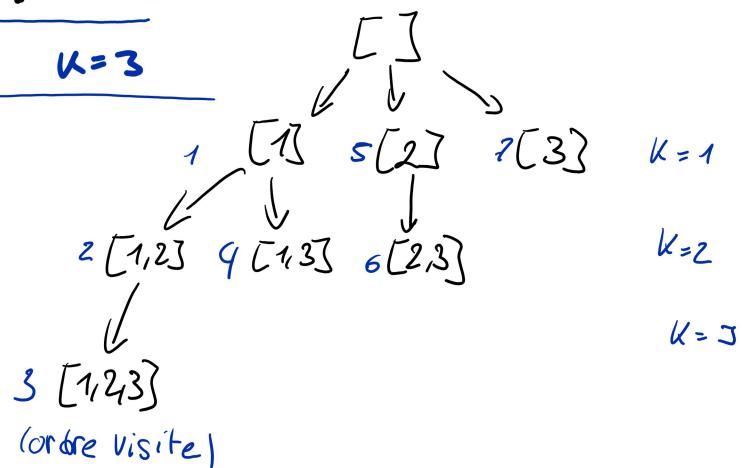
1. Structure

- Itérer^(à l'étape k) sur l'ens. des val. courantes possibles (déf par les cmds explicites T_k)
- Tester si $\{x_i\}_{i=0}^n$ est une solution locale (B_k)
(i.e. si c'est un candidat potentiel de sol glob.)
 - Si Oui: Tester si c'est une sol glob (P)
(Δ \exists sol glob pour $k < n$)
 - ↳ Si oui: Sol trouvée
 - ↳ Si non: passer au step $k+1$
(fait naturellement un arbre, $k = \text{depth}$)
 - Si non: continuer l'itération de la step k
- Une fois l'itération step k finie, rec call se fini et remonte (backtrack) à celle (encours) de la step $k-1$



$N=3$ (ici)

e.g. powerset de $\{1, 2, 3\}$



2. Résolution cas général

def BT(...):

determined d'après $X[0:k]$

def T(x, k, N): ↙

def B(x, k, N):] determined d'après $X[0:k+1]$

def P(x, k, N):]

oat = None

def rBT(x, k, N): ^{non local oat}

for y in T(x, k, N):

$X[k] = y$

if B(x, k, N):

if P(x, k, N):

$oat = X[:k+1]$

return oat $\{x_i\}_{i=0}^n$

return oat

exit point of rec call. 1st stmt after exit

rBT(x, k+1, N) ^{Sts seem as out is not None : direct returned}

\hookrightarrow break for & call stack

if oat is not None: return oat

return rBT([None], N, 0, N)

NB : pour return toutes les solutions :

enlever les return oat / if oat is not ... et

remplacer par "oat.append(X[:k+1])". Ici oat sera