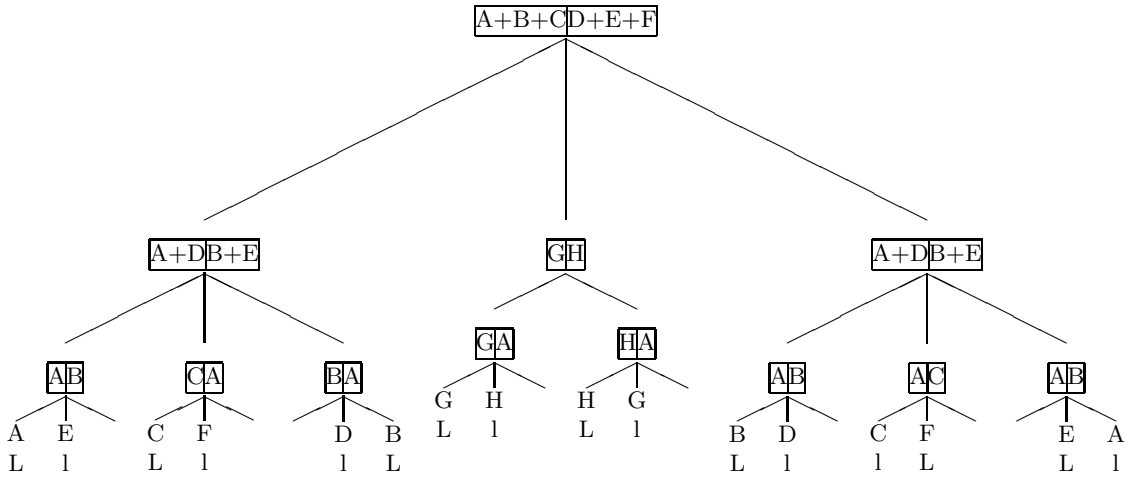
5.5 Arbres comme structures de contrôle

5.5.1 Arbres de décision

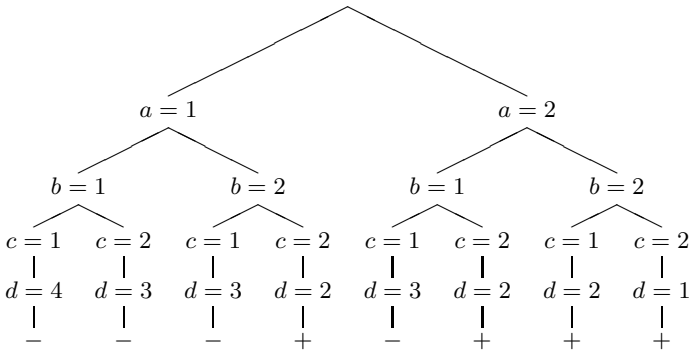
Concerne les problèmes pour lesquels on n'a pas d'autre méthode de recherche que d'énumérer les solutions, sans omission ni répétition.

Exemple 1 : pesée 8 pièces à peser, qui ont toutes le même poids, sauf une, qui est fausse (on ne sait pas si elle est plus lourde ou plus légère). On n'a qu'une balance sans poids de références à notre disposition.



Exemple 2 : résolution de l'équation $N = a + b + c + d$, $1 \leq a, b, c, d \leq k$

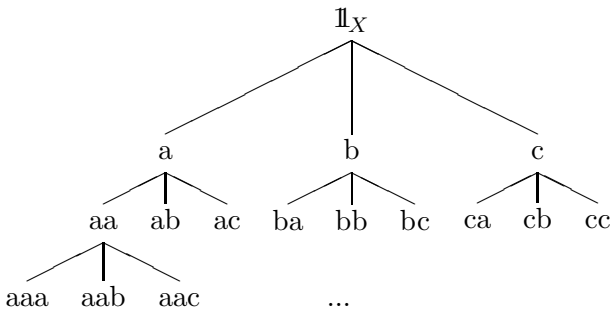
Avec $N = 7, k = 2$, l'arbre sera binaire (à cause de k)



Exemple 3 : Lister tous les mots de longueur n sur un alphabet à p lettres

On peut écrire n boucles imbriquées mais la structure du programme contiendrait alors la valeur n et le programme ne conviendrait pas pour $n+1$.

Solution : un arbre de profondeur n, chaque sommet ayant p fils. Par exemple avec $n = 3$ et $p = 3$:



Pour réaliser l'algorithme, il suffit alors de traduire les primitives utilisées dans l'agorithme standard de parcours :

existe_fils	Soit m la distance du sommet à la racine, alors `existe_fils := (m < n)`
premier_fils(x)	`m :=m+1 ; VAL(x) := 'a'`
existe_frère(x)	`existe_frère := (VAL(x) <= 'c') ;`
frère(x)	`VAL(x) := succ(VAL(x)) ;`
racine	`m = 0`

On ajoute dans l'algorithme un affichage à l'arrivée sur une feuille : en 4, afficher le contenu de la pile (« pile photographiable ») + `VAL(x)`.

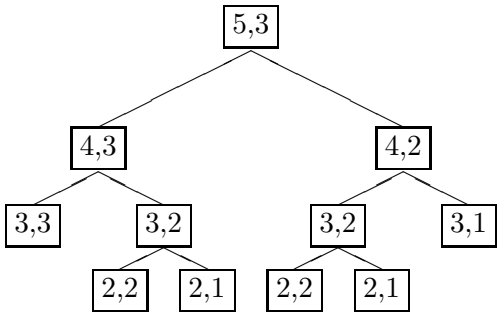
5.5.2 Arbres des appels récursifs

Exemple : C_n^p

Combinaisons $C(n,p)$ de n objets p à p :

$$C(n,p) = \begin{cases} n \text{ si } p = 1 \\ 1 \text{ si } p = n \\ C(n-1,p) + C(n-1,p-1) \text{ sinon} \end{cases}$$

```
fonction CNP(n,p : entier) : entier ;
début
  si p = 1 alors CNP := n
  sinon si p = n alors CNP := 1
        sinon CNP := CNP(n-1,p)
                   + CNP(n-1,p-1) ;
fin ;
```



Remarquer que l'évaluation de la fonction `CNP`, pour les valeurs (n,p), correspond au parcours de l'arbre de racine (n,p), en profondeur. Cela permet d'une part de bien comprendre le fonctionnement, et le déroulement temporel d'une telle fonction récursive ; cela permet aussi de **dé-récursiver** un sous-programme récursif : il suffit d'utiliser l'algorithme de parcours itératif de l'arbre des appels. Pour cela, on traduit les primitives habituelles : fils gauche de (n,p) : $(n-1,p)$; frère de (n,p) : $(n,p-1)$; feuille : $p = 1$ ou $n = p$, etc.