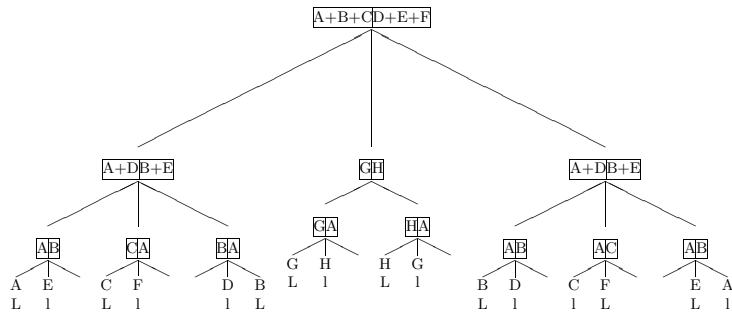


5.5 Arbres comme structures de contrôle

5.5.1 Arbres de décision

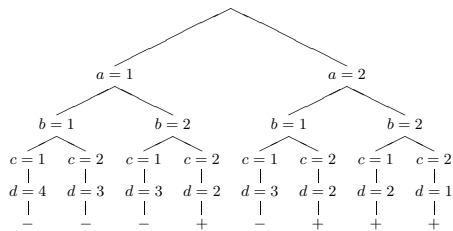
Concerne les problèmes pour lesquels on n'a pas d'autre méthode de recherche que d'énumérer les solutions, sans omission ni répétition.

Exemple 1 : pesée 8 pièces à peser, qui ont toutes le même poids, sauf une, qui est fautive (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 $N = a + b + c + d$, $1 \leq a, b, c, d \leq k$

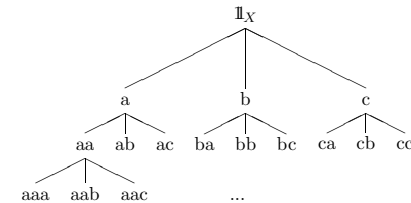
Avec $N = 7, k = 2$, l'arbre sera binaire (à cause de k)



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'algorithme standard de parcours :

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

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

Permet en particulier la dé-récursivation d'un (sous-)programme récursif.

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 ;
    
```

