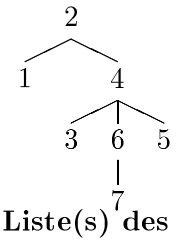
6.3.1 Implémentation contigüe

Tableau `père`

Exemple :



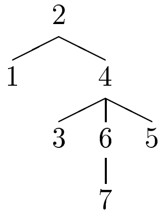
	1	2	3	4	5	6	7
Père	2	0	4	2	4	4	6

Coût important de presque toutes les primitives ; et même impossibilité pour l'ordre des fils (à moins de poser une convention d'ordre représentée par la numérotation).

Liste(s) des fils

La totalité des informations nécessaires peut être représentée par un tableau contenant la liste des fils de chaque nœud.

Exemple :



Soit une numérotation arbitraire des sommets. On concatène dans un tableau `LS` les listes des fils de chaque nœud, chaque liste se terminant par 0.

1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	4	0	0	3	6	5	0	0	7	0	0
$\alpha(1)$	$\alpha(2)$			$\alpha(3)$	$\alpha(4)$				$\alpha(5)$	$\alpha(6)$		$\alpha(7)$

Occupation mémoire : $2N-1$, $N=$ nombre de sommets (chaque sommet apparaît une fois — sauf la racine, et chaque liste de fils occupe une case en plus pour la fin de liste (0)).
Traduction des primitives :

`existe_fils(x)`	la x$^{\mathrm{e}}$ liste commence par zéro.
`premier_fils(x)`	premier élément (normalement $\neq 0$) de la x$^{\text{ième}}$ liste
`existe_frère(x)`	Il faut trouver x dans le tableau, et regarder sa case de droite.
`frère(x)`	*Idem*
`racine(x)`	x n'apparaît pas dans le tableau
`père(x)`	`i`, tel que `x` apparaît dans la `i`$^{\mathrm{e}}$ liste dans le tableau

Toutes les primitives sont au moins d'ordre N.
Pour diminuer la complexité, on peut ajouter de l'information. Par exemple, tableau `PS` qui indique pour chaque sommet la position de sa liste de fils dans `LS`.

Ici, `PS :`

1	2	3	4	5	6	7
1	2	5	6	10	11	13

`existe_fils(x)` devient : `LS[PS[x]] !=0`.

Mais les primitives concernant le passage au frère restent en $O(N)$. On peut ajouter encore de l'information. Soit le tableau `PLACE` qui indique pour chaque sommet à quel endroit il apparaît dans le tableau `LS`.

Ici, `PLACE :`

1	2	3	4	5	6	7
2	0	6	3	8	7	11

`existe_frère(x)` devient : `LS[PLACE[x]+1] !=0`.

Cette fois, toutes les primitives (sauf `racine`, à moins d'un dispositif spécial) sont en temps constant, et on peut implémenter l'algorithme de parcours itératif (occupation mémoire totale : $4N-1$).