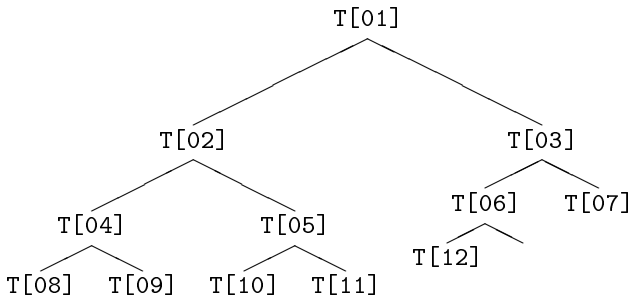
Amélioration du tri arborescent tri par tas (maximier, *heapsort*) (J. Williams, 64).

L'idée est d'essayer de conserver l'information stockée dans l'arbre précédent, sans avoir ce coût de stockage $(2n-1)$.

Principe : on stocke un arbre dans un tableau, en appliquant la règle suivante : un *tas* est défini par une suite `T[1]...T[N]` telle que $\forall i \in [1..N/2]$, `T[i]`$\leq$`T[2*i]` et `T[i]`$\leq$`T[2*i+1]`.

Graphiquement, un tableau de 12 cases, `T[1]...T[12]`, par exemple, correspondrait donc à l'arbre suivant, où chaque nœud est plus petit que tous ses fils.



Le tri arborescent défini précédemment construit d'abord l'arbre. Dans cette version, on va donc construire d'abord un tas. Principe : chaque nouvel élément est d'abord introduit à gauche du tableau, c'est-à-dire en position de racine. Ensuite, il est *glissé* — par échanges successifs — jusqu'à sa position finale dans le tableau.

Exemple : soit le tableau

	42	06	55	94	18	12

qui forme un tas. Supposons que nous voulons ajouter la valeur 44. Les étapes successives sont représentées ci-après :

44	42	06	55	94	18	12
06	42	44	55	94	18	12
06	42	12	55	94	18	44

Le résultat est un nouveau tas.

On peut envisager de créer le tas dans le tableau de départ lui-même. En effet, dans ce tableau, la seconde moitié, qui correspond aux feuilles de l'arbre, vérifie trivialement la définition d'un tas. Alors on peut supposer que notre 1/2 tableau droit est un tas, et considérer chaque élément comme inséré à son tour.

On a donc un algorithme de construction du tas. Mais cela ne donne qu'un ordre partiel. Pour obtenir l'ordre total, il faut faire ces opérations de glissement n fois. On peut une fois encore le faire dans le tableau de départ. Par exemple : à chaque étape on « extrait » la plus petite clé : comme elle est à gauche, on l'échange avec le dernier élément du tableau, et on considère que le tableau fait une case de moins. Alors il faut placer correctement la nouvelle « racine ». Puis on recommence... le résultat est à l'envers, mais ça peut s'arranger.

Exemple avec comme tableau initial :	07	12	44	91	23	31	10	67
La construction du tas donne :	07	10	12	23	44	31	91	67
Les n étapes de glissement :								
On échange :	67	10	12	23	44	31	91	07
... et on fait glisser :	10	23	12	67	44	31	91	07
On échange :	91	23	12	67	44	31	10	07
... et on fait glisser :	12	23	31	67	44	91	10	07
etc...								
... jusqu'à obtention du tableau trié :	91	67	44	31	23	12	10	07

Algorithme complet : [Wirth, 1987, p. 87]