Arbres 2-3-4 (2-3-4 trees)

Pour simplifier les explications : dans un nœud, pour chaque clé n° i on nommera *fils gauche* le fils n° i, et *fils droit* le fils n° $i+1$.

1 Préliminaires

Exercice 1.1 (Arbres : de recherche, B, B+. . .)

Qu'est-ce ? A quoi ça sert ?

1. Un arbre (général) de recherche (A m-way search tree or (general) search tree)
2. Un arbre B (A B-tree), un arbre B+ (a B+ tree)

Exercice 1.2 (Et les arbres 2-3-4 ?)

1. Quelles sont les propriétés d'un arbre 2-3-4 ?
2. Donner un type permettant de représenter des arbres 2-3-4.

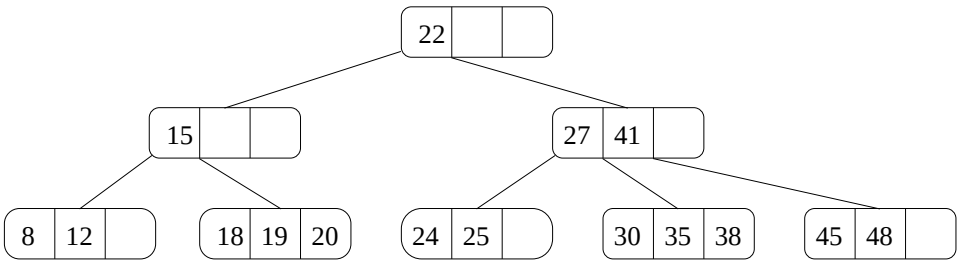

FIGURE 1 – Arbre 2-3-4

Exercice 1.3 (Minimum et maximum)

1. Où se trouvent la valeur minimum et la valeur maximum d'un arbre 2-3-4 ?
2. Écrire les deux fonctions permettant de récupérer la valeur minimum (resp. maximum) d'un arbre 2-3-4 non vide.

Exercice 1.4 (Recherche d'un élément – *contrôle nov. 12*)

1. Écrire un algorithme qui recherche un élément dans un arbre 2-3-4, donne un pointeur sur le nœud contenant l'élément si la recherche est positive, la valeur NUL sinon.
2. Si l'algorithme écrit est récursif, le "dé-récursiver". Le rendre récursif dans le cas contraire.

Exercice 1.5 (Intervalle – *contrôle nov. 12*)

L'objectif de cet exercice est d'afficher un intervalle de clés dans un arbre 2.3.4. Nous utiliserons ici la procédure `ecrire` (plutôt que de construire une chaîne) qui est capable de *convertir* les clés de nos arbres pour les afficher.

Écrire la procédure `range (`A`,` b_i`,` b_s`)` qui affiche (en ordre croissant) l'ensemble des clés se trouvant dans l'arbre 2.3.4 `A` et comprises dans l'intervalle $[bi; bs]$.

---

2 Insertions – Suppressions

Exercice 2.1 (Inserer un nouvel élément : la méthode classique)

1. (a) Où peut-on insérer un nouvel élément dans un arbre 2-3-4 afin qu'il conserve ses propriétés ?
   (b) Quel problème cela pose ?
   (c) Quelle transformation peut être appliquée à l'arbre pour le résoudre ? Dans quelles conditions peut-on effectuer cette transformation ?
   (d) Écrire l'algorithme réalisant cette transformation dans le *cas idéal*[1].
2. On peut utiliser cette transformation de deux manières : à la descente (*principe de précaution*) ou à la remontée.

   Pour chacune de ces méthodes :
   (a) Donner le principe d'insertion.
   (b) Construire un nouvel arbre 2-3-4 par insertions successives des éléments suivants : 5, 15, 40, 25, 18, 45, 38, 42, 9.
3. Écrire l'algorithme d'insertion dans un arbre 2-3-4 en appliquant le principe de précaution.

Exercice 2.2 (Suppression d'un élément : à la descente)

1. Comment supprimer une clé lorsque l'on n'est pas en feuille ? Utiliser comme exemple la suppression de la valeur 27 dans l'arbre de la figure 1 (on préférera ici systématiquement le coté gauche, mais ce choix pourra être remis en cause plus tard !).
2. (a) Quel problème pose la suppression de la valeur 24 dans l'arbre obtenu ? Quelle transformation permet de résoudre le problème (s'inspirer des transformations sur les AVL) ? Supprimer cette valeur, puis la valeur 25, en utilisant cette nouvelle transformation.
   (b) Écrire les deux algorithmes de cette transformation (à gauche, à droite), en précisant les conditions d'appels.
3. (a) Quel problème pose la suppression de la valeur 35 dans le dernier arbre ? Quelle nouvelle transformation faut-il appliquer ?
   (b) Écrire l'algorithme de la nouvelle transformation, en précisant les conditions d'appel.
4. En appliquant un principe de précaution similaire à celui vu pour l'insertion, supprimer successivement les valeurs 15, 22, 8, 20, 30, 18, 45, 12, 48, 19, 41. Essayer de limiter les destructions de nœuds.
   Construire en même temps le principe de la suppression d'un élément dans un arbre 2-3-4.
5. Écrire l'algorithme de suppression.

1. Pas la racine de l'arbre, pas de soucis de "place".

Bonus

Exercice 2.3 (Insertion, une nouvelle méthode : pour changer un peu. . .)

L'insertion vue ci-dessus entraîne quelquefois des modifications de la structure de l'arbre inutiles. Le but ici est de trouver d'autres transformations qui permettent l'insertion sans pour autant augmenter le nombre de nœuds de l'arbre.

1. Une nouvelle transformation ?
   (a) Comment éviter d'effectuer un éclatement lors de l'insertion de l'avant-dernier élément (42) dans l'exemple de l'exercice 2.1 ?
   (b) Utiliser la même méthode pour ajouter successivement les valeurs 33 et 36 dans l'arbre de la figure 1 (préférer le côté gauche). Peut-on utiliser la même méthode pour insérer la valeur 42 ?
2. L'ajout :
   (a) Ajouter à l'arbre obtenu les valeurs 42 et 16. Préciser les conditions dans lesquelles on pourra utiliser les transformations pour une insertion.
   (b) Ajouter enfin les valeur 37 puis 23. Doit-on utiliser le principe de précaution ?
   (c) En déduire le nouveau principe de l'algorithme d'insertion.
   (d) Écrire l'algorithme d'insertion.

Exercice 2.4 (Suppression d'un élément : à la remontée !)

L'algorithme de suppression utilisant le principe de précaution est trop coûteux en modifications : par exemple, la suppression d'une clé qui n'existe pas dans l'arbre 2-3-4 peut quand même modifier l'arbre.

Ici la suppression se fera sans utiliser le principe de précaution, c'est à dire en effectuant les modifications seulement lorsque cela est nécessaire : à la remontée.

1. On reprend ici les suppressions effectuées à l'exercice 2.2 sur l'arbre de la figure 1, mais cette fois-ci elles seront effectuées avec modifications à la remontée.
   (a) Commencer par supprimer 27, 24, 25, 35. Des remarques ?
   (b) Continuer les suppressions : 15, 22, 8, 20, 30, 18, 45, 12, 48, 19, 41.
2. En déduire le principe de la suppression d'un élément dans un arbre 2-3-4 avec modifications à la remontée.
3. Écrire cet algorithme de suppression.