B-Arbres

Dernière mise à jour: 15-06-2020[1]

Introduction

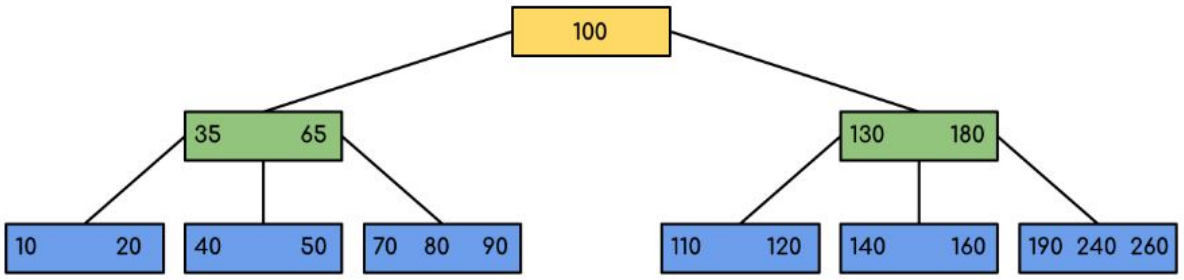
Un B-Arbre est un arbre de recherche auto-équilibré. Dans la plupart des autres arbres de recherche auto-équilibrés (comme les arbres AVL et Rouge-Noir), on suppose que tout est dans la mémoire principale. Pour comprendre l'utilisation des B-Arbres, nous devons penser à l'énorme quantité de données qui ne peuvent pas tenir dans la mémoire principale. Lorsque le nombre de noeuds de l'arbre est élevé, les données sont lues à partir du disque sous forme de blocs. Le temps d'accès au disque est très élevé par rapport au temps d'accès à la mémoire principale (la RAM). L'idée principale de l'utilisation des B-Arbres est de réduire le nombre d'accès au disque. La plupart des opérations dans un arbre (recherche, insertion, suppression, max, min, .. etc.) nécessitent des accès disque $O(h)$ où h est la hauteur de l'arbre. Un B-Arbre est un gros arbre. La hauteur des B-Abres est maintenue basse en plaçant le maximum de clés possibles dans un nœud. En général, la taille du nœud d'un B-Arbre est maintenue égale à la taille du bloc de disque. Étant donné que la hauteur d'un B-Arbre est faible, **les accès au disque pour la plupart des opérations sont considérablement réduits** par rapport aux arbres de recherche binaires équilibrés comme AVL, Rouge-Noir.

Propriétés d'un B-Arbre:

1. Toutes les feuilles sont au même niveau.
2. Un B-Arbre est défini par le terme *degré minimum* t. La valeur de t dépend de la taille du bloc de disque.
3. Chaque nœud sauf root doit contenir au moins $t-1$ clés. La racine peut contenir au minimum 1 clé.
4. Tous les nœuds (y compris la racine) peuvent contenir au plus $2t-1$ clés.
5. Le nombre d'enfants d'un nœud est égal au nombre de clés qu'il contient plus 1.
6. Toutes les clés d'un nœud sont triées par ordre croissant. L'enfant entre deux clés k1 et k2 contient toutes les clés comprises entre k1 et k2.
7. Un B-Arbre grandit et rétrécit à partir de la racine, contrairement à un ABR. Les ABR poussent vers le bas et rétrécissent également à partir du bas.
8. Comme d'autres arbres de recherche binaires équilibrés, la complexité du temps pour rechercher, insérer et supprimer est O (log n).

[1] https://www.geeksforgeeks.org/introduction-of-b-tree-2/

Voici un exemple de B-Arbre d'ordre minimum 5. Notez que dans les B-Arbres pratiques, la valeur de l'ordre minimum est bien plus que 5.



Nous pouvons voir dans le diagramme ci-dessus que tous les nœuds feuilles sont au même niveau et tous les non-feuilles n'ont pas de sous-arbre vide et ont une clé de moins que le nombre de leurs enfants.

Faits intéressants:

1. La hauteur minimale de l'arbre B qui peut exister avec n nombre de nœuds et m est le nombre maximum d'enfants d'un nœud peut avoir est:
   $h_{min} = \lceil \log_m(n+1) \rceil - 1$
2. La hauteur maximale de l'arbre B qui peut exister avec n nombre de nœuds et d est le nombre minimum d'enfants qu'un nœud non racine peut avoir est:
   $h_{max} = \lfloor \log_t \frac{n+1}{2} \rfloor$ et $t = \lceil \frac{m}{2} \rceil$

Opérations dans un B-Arbre

Parcours dans un B-Arbre

Traversal est également similaire à Inorder traversal of Binary Tree. Nous partons de l'enfant le plus à gauche, imprimons récursivement l'enfant le plus à gauche, puis répétons le même processus pour les enfants et les clés restants. À la fin, imprimez récursivement l'enfant le plus à droite.

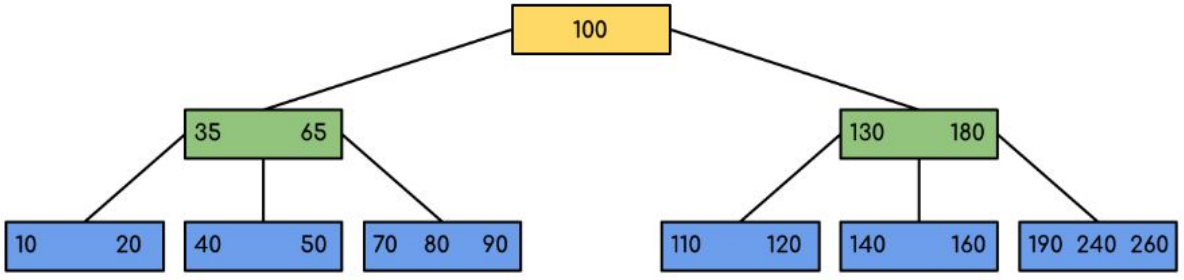
Opération de recherche dans B-Tree

recherche est similaire à la recherche dans Binary Search Tree. Soit la clé à rechercher k. Nous partons de la racine et descendons récursivement. Pour chaque nœud non feuille visité, si le nœud a la clé, nous renvoyons simplement le nœud. Sinon, nous revenons à l'enfant approprié (l'enfant qui est juste avant la première clé supérieure) du nœud. Si nous atteignons un nœud feuille et ne trouvons pas k dans le nœud feuille, nous retournons NULL.

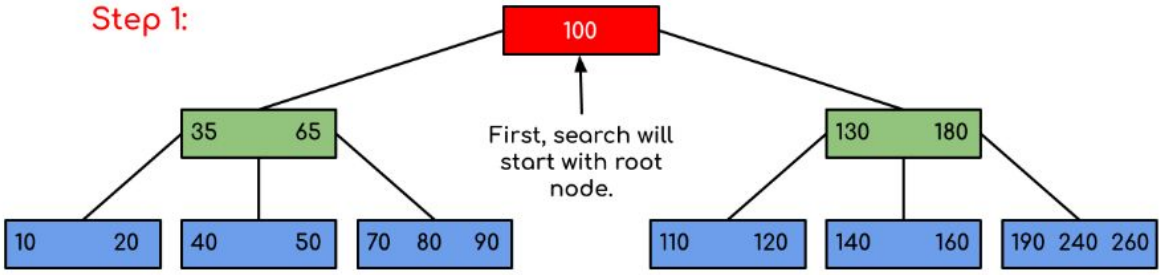
Logique

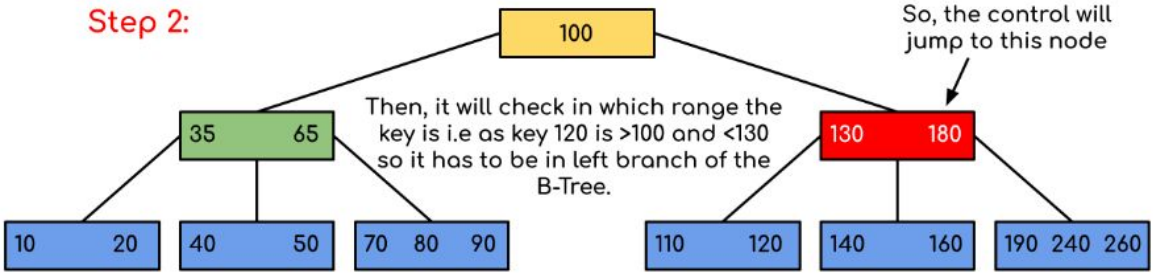
La recherche dans un B-Arbre est similaire à la recherche dans un arbre binaire. L'algorithme est similaire et va avec la récursivité. A chaque niveau, la recherche est optimisée comme si la valeur clé n'était pas présente dans la plage du parent alors la clé est présente dans une autre branche. Comme ces valeurs limitent la recherche, elles sont également appelées valeurs limites ou valeurs de séparation. Si nous atteignons un nœud feuille et ne trouvons pas la clé souhaitée, il affichera NULL.

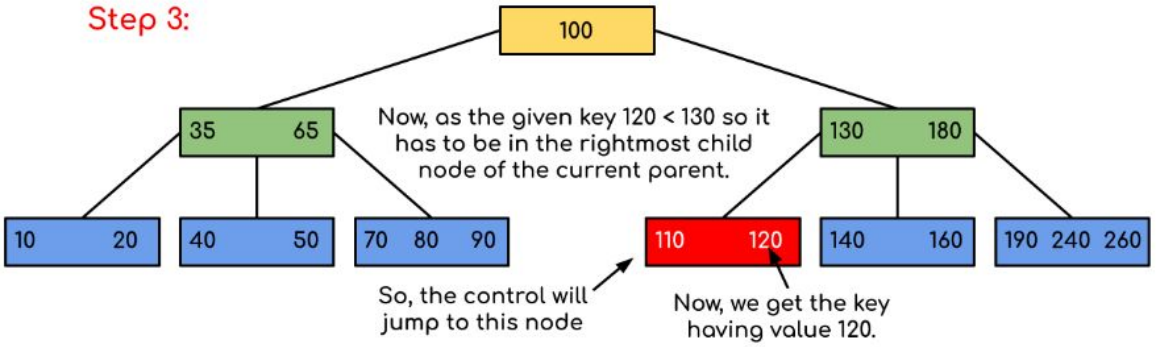
Exemple: recherche de 120 dans le B-Arbre ci-dessous.



Solution:







Dans cet exemple, nous pouvons voir que notre recherche a été réduite en limitant simplement les chances où la clé contenant la valeur pourrait être présente. De même, si dans l'exemple ci-dessus, nous devons rechercher 180, le contrôle s'arrêtera à l'étape 2 car le programme trouvera que la clé 180 est présente dans le nœud actuel. Et de même, s'il s'agit de rechercher 90, alors 90 <100, il ira automatiquement dans le sous-arbre droit et le flux de contrôle ira donc de la même manière, comme indiqué dans l'exemple.

Tâches à Faire

En vous basant sur vos propres recherches et sur les ressources du cours, proposez un programmes avec les fonctions suivantes:

- insertion dans un B-Arbre,
- suppression dans un B-Arbre,
- recherche dans un B-Arbre,
- transformation d'un arbre binaire de recherche (ABR) en un B-Arbre,
- menu de choix pour effectuer les différentes opérations dans un B-Arbre.

Afin de mieux tester l'utilité des B-Arbres,

1. créer un ABR;
2. faire une fonction itérant sur l'insertion successive de 1000 à 10000 valeurs aléatoires dans l'ABR, avec une intervalle de 1000, c'est-à-dire, vous testez d'abord sur 1000, ensuite 2000, ensuite 3000, …, jusqu'à 10000 (remarque: vous pouvez aller aussi au delà, rien ne vous empêche de tester avec 100000 noeuds);
3. pour chaque test à la question précédente, appeler une fonction de recherche dans cet ABR, et estimer la durée de calcul;
4. comparer les durées en temps de calcul les différentes cardinalités (1000, 2000, 3000, 4000, 5000, …, 10000);
5. Qu'observez-vous ?

6. Refaire les étapes 2 à 4 cette fois ci avec un B-Arbre.
7. Qu'observez-vous?
8. Comparer les durées obtenues avec les ABR et celles obtenues avec les B-Arbres.
9. Quelles conclusions tirez-vous de ces étapes et comparaisons ?