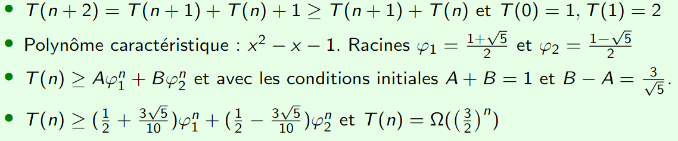
Variant de boucle: Propriétés inductives **–**while: après la k e itération, j vaut x −1 −k et 0 ≤j < x, j = j – 1, donc j ≤n −2 **–**for : après la k e itération, i vaut k + 1 et i < n

`<<`: 相当于乘2^n, n为移动位数，对2进制操作 `>>`：相当于除以2^n, `|`：按位或运算符, `&`：按位与运算符

**terminaison :** Dans la boucle **while** d’insert, la valeur de j est dans N (par l’invariant qu’on a prouvé) et décroit strictement à chaque itération. **Récursif :** Par induction/récurrence sur len(A) - i : la fonction termine et renvoit le maximum du tableau à partir de l’indice.

On trouve un invariant qui doit être satisfait par les itérations de tout hypothétique algorithme ;mais qui n’est pas satisfait par la configuration finale recherchée. Complexité asymptotique:

meilleur cas pire cas **fib:**

Quick: O(n2), O(nlogn), O(nlogn)

Heap : O(nlogn), O(1)

Merge: 同heap,O(n)

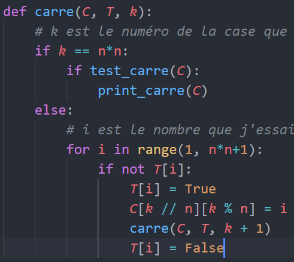
cas moyen (espérance) :

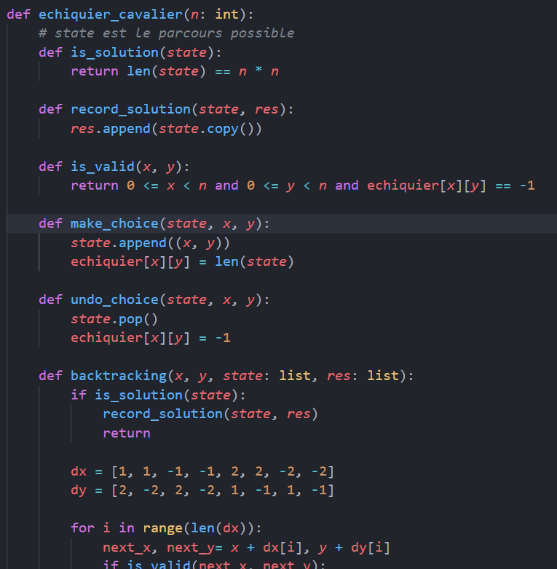
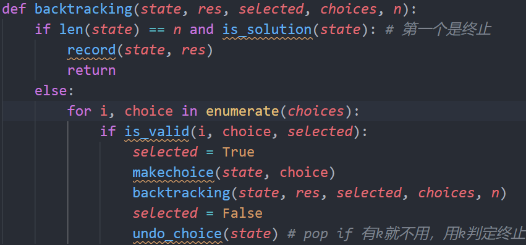
Sans information supplémentaire,

on suppose ici que chaque permutation des éléments de A est **équiprobable**.

Nombre d’inversions:

Démontrer l’invariant suivant de la boucle pour : Après l'itération 0, … , Supposons que l'invariant soit vrai après l'itération k-1, … , lors de l'itération k, …分治算法考虑合起来的时候是否跟回溯相似

**Grill :** , or 存取点的坐标，比如(i, j) 二分查找不一定是递归，可以循环

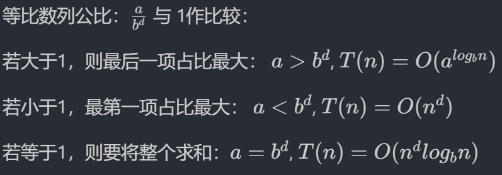
Backtracking: 如果有k就不用弹出，因为是用k判断终止而不是len

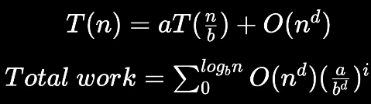
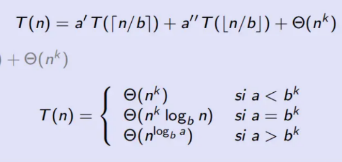
Diviser er régner :

先分到最小，

然后合起来的时候一定要注意从整

体考虑，比如两部分，该如何选择

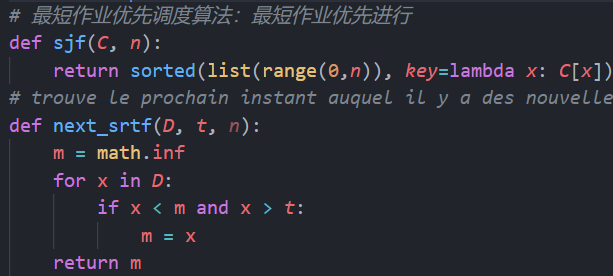
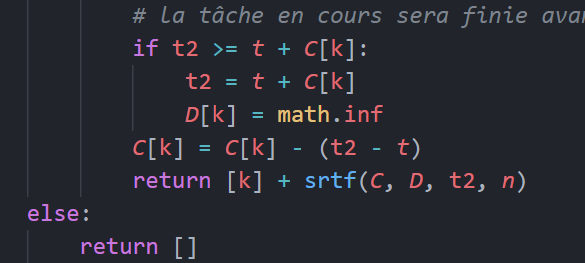
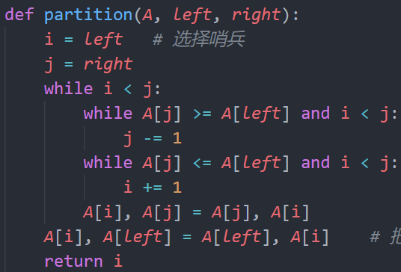
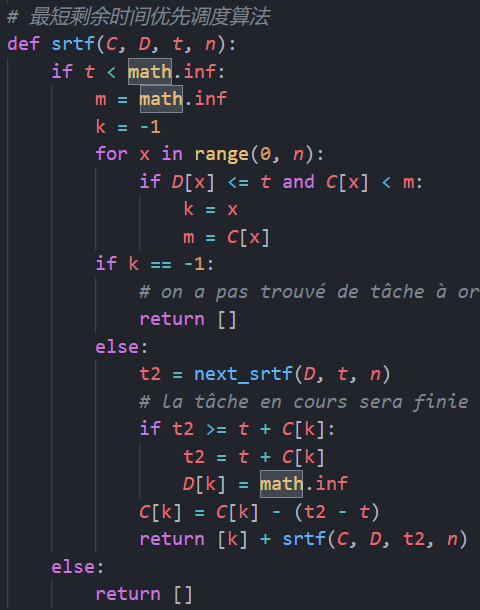
其中一部分。可能选择的这一步要多写一个函数，如merge\_sort

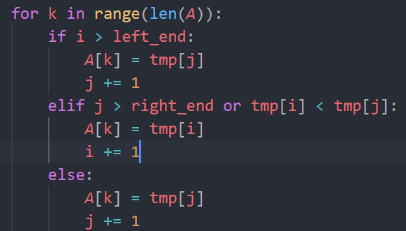
Master theorem:

Dp:类似分治，找

到合适的递归方程，然后从小往大开始用循环

并用矩阵存储值，每一个i都是这个i的解。sous-structure optimale, sous-problèmes qui se chevauchent

Algo glouton : à chaque étape on fait le choix qui optimise localement l’objectif



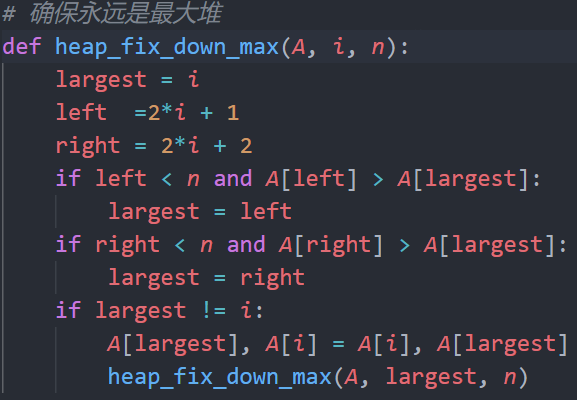
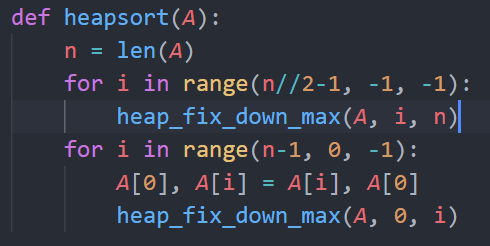
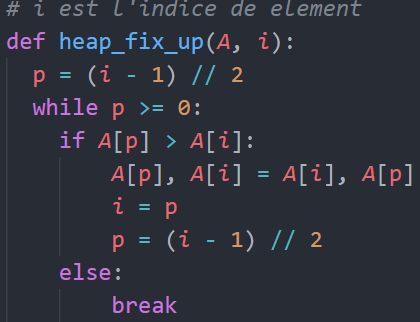
File : enqueue, dequeue—FIFO

Pile : empiler, depiler(pop)—LIFO

Dérécursification : ?

Tas de binaire == Files de priorité: Toujours sortir le plus petit élément de la file

重点问题是分清楚是max heap还是min heap，反正就是要注意性质和的运用。插入删除都涉及往前回溯还是往后回溯，堆排序涉及到往后回溯。是叶子, 是最后一个叶子上面的父节点的索引。





Arbre binaire

Un arbre n-aire a au

plus n sous-arbres qui sont eux-aussi n-aires. La hauteur de l’arbre est

la longueur du plus long chemin de la racine à une feuille

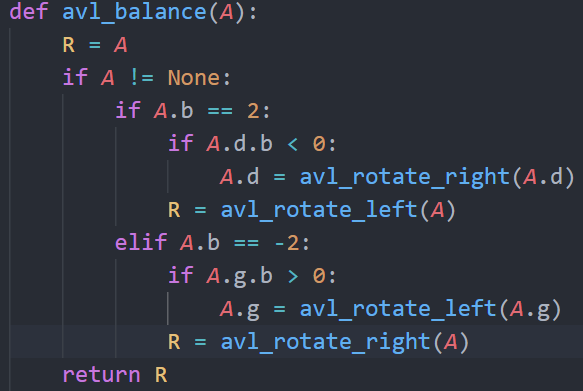
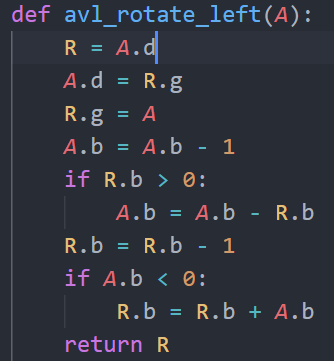
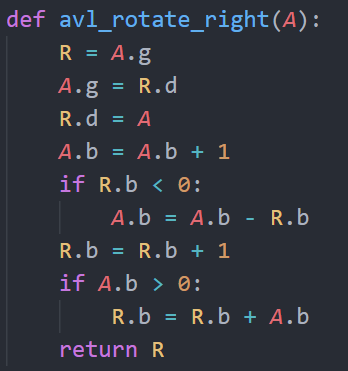
Un arbre est équilibré en hauteur si les hauteurs de ses sous-arbre

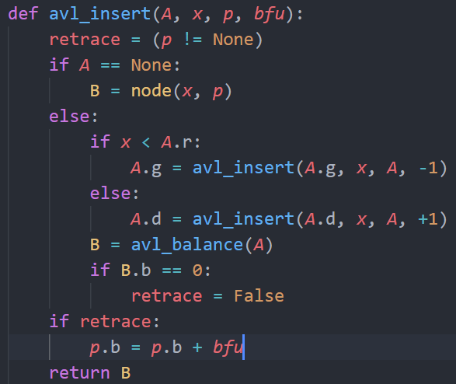
diffèrent d’au plus 1. Un arbre n-aire équilibré = une hauteur d’au plus

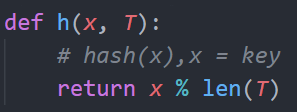
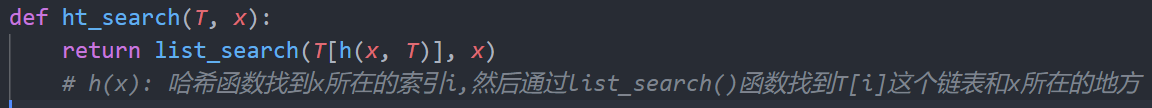
二叉搜索树先注意定义，左边的永远小于根节点，右边的永远大于根节点。并且区分prefix, inorder, postfix，这三种遍历方式。递归时要注意终止条件，和哪些时候到左边或者右边，分清楚多种情况，什么时候该添加或者删除

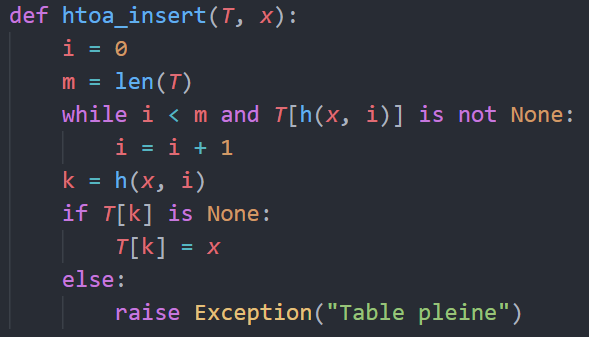
删除：三种情况—某一边没有，两边都有—让右子树最小的替换到x

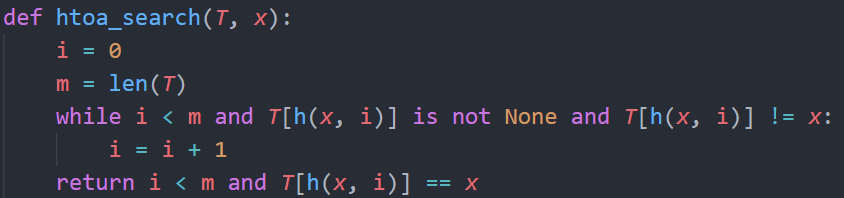
AVL : est un ABR dans lequel pour tout nœud x , le sous-arbre enraciné en x est équilibré (en hauteur)

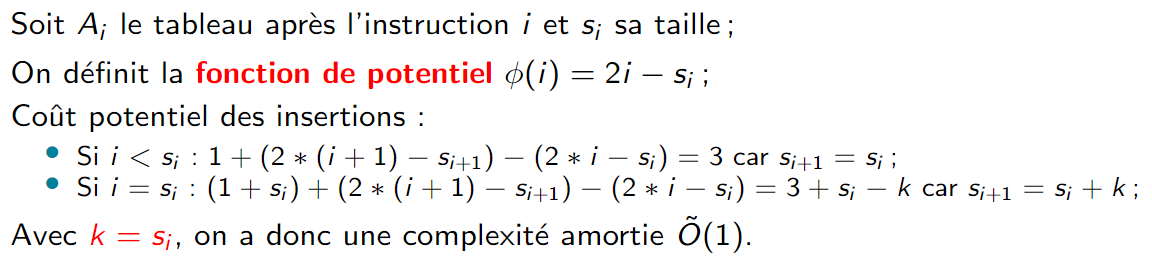
le facteur d’équilibre : A.b = hauteur(A.d) - hauteur(A.g) [-2, 2] 操作结合ABR



Tables de hachage: collisions T是一个list, h映射—x在T的位置

La complexité pire cas ne dépend pas de la taille m de la table, 去链表搜索

adressage ouvert :

如果遇到符号判定，直接 « if是不是 »就行

