**作業一 排序演算法比較**

1. **測試環境**

作業系統：Microsoft Windows 10 版本20H2(OS組建19042.928)

測試環境：visual studio code

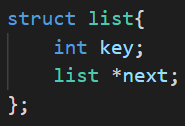
處理器：intel core i5-8265U

RAM：8GB

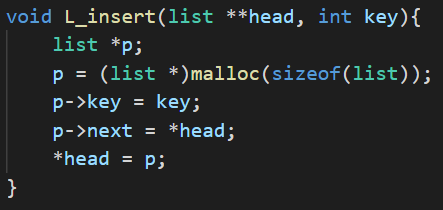
Git連結：https://github.com/kingslayer929/409410017-data-structure-compare

1. **解釋各種資料結構(linked list, array, array with binary search, BST, hash)**

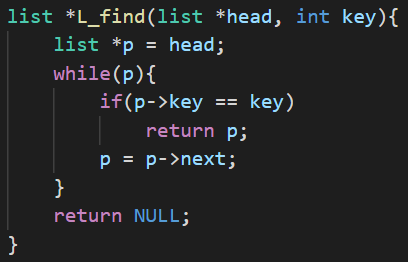
* **Linked list**



主要結構是由一個數值(key)和一個指向自身結構的指標(next)。



1. 輸入時給予串列的起頭與欲輸入的值。
2. 輸入時先取得一塊記憶體，將欲輸入的值放進此自己的key中。
3. 接著將整條串列接到自己之後，將自己改為整條串列的頭。
4. 不停重複1, 2, 3直到輸入完畢。



搜尋是否有欲搜尋的值時，給予串列開頭與欲搜尋的值。從頭開始找，不是就換下一個，直到找到後回傳位址，否則回傳空。

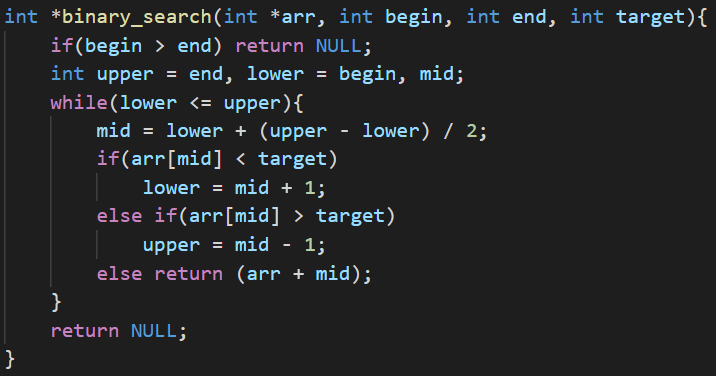
* Array

此為基本陣列操作，將值輸入，並從頭找尋是否有欲搜尋的值。

* Array with binary search

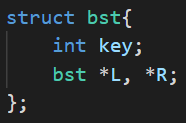
將上述的Array輸入後，進行排序。(一定要進行排序)

接著進行二分搜尋。

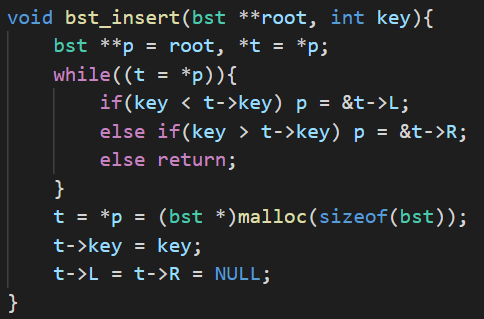


1. 給定欲搜尋陣列、陣列開頭、陣列結尾、搜尋目標。
2. 起始上界與下界為陣列開頭與結尾。
3. 中間值為上界與下界中間。從中間找，若目標比陣列中間的值小，則其值若在陣列中存在，必定在陣列中間以前，將上界改為中間值，反之則必定在陣列中間以後，將下界改為中間值。此操作可刪去目前搜尋範圍的一半，達到縮小搜尋範圍的功能。
4. 執行3直到目前中間值等於目標值，回傳目前中間值位址，否則回傳空。

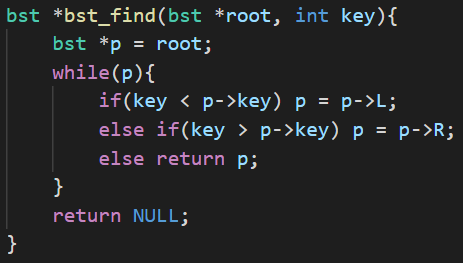
* BST (binary search tree)



主要結構是由一個數值(key)和兩個指向自身結構的指標。

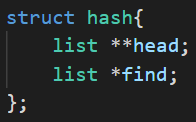


1. 輸入時給予二元搜尋樹的根與欲輸入的值
2. 與linked list不同，插入前先選擇位置，選擇方式為3, 4
3. 若此位置尚未插入，則選此位置。
4. 若此位置的值教育插入值大，位置移到左節點，反之一道右節點。
5. 重複3, 4直到找到位置，即可分配一塊空間，輸入數值。



1. 搜尋時給予二元搜尋樹的根與欲搜尋的值。
2. 與輸入時相同的方式找出此值應在的位置，若存在回傳目標位址，否則回傳空。

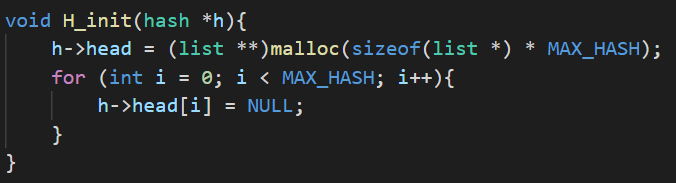
* Hash



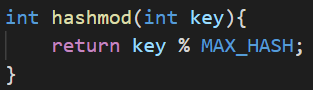
此種hash的方式為linked list陣列，並存在一個find用於儲存找尋後該值的位址。



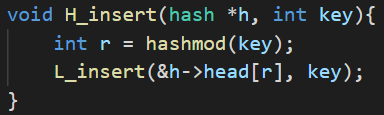
此定義hash值最多1,000,000。



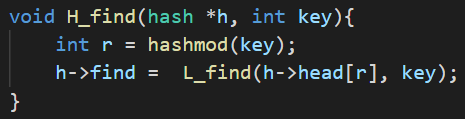
使用前必須先初始化，做出MAX\_HASH個linked list陣列，並將起始值都設為空。



此函式會以key回傳一個值，代表此key應存於哪個linked list中。



輸入時給予一個hash結構與欲輸入的值，並由hashmod得出英輸入哪條linked list。



搜尋與輸入類似，並將位址傳到結構中的find中。

1. **亂數產生方式**



在函數之外預開兩條陣列(全域)，分別為輸入陣列與搜尋陣列。