Search部分

Linear (Sequential) Search(線性、循序搜尋)

定義:從頭到尾，一一比較各資料之鍵值，直到找到符合者，或搜尋完整資料皆找不到為止

分析:

1. Data不須事先經過牌序，即可Search
2. Data之保存在Random Access => Array，Sequential Access => Link List皆可
3. Time = O(n)

Binary Search(二分搜尋法)

前提:

1. Data必須事先排序過(小=>大)
2. Data保存在Random Access(例:Array)機制上

觀念:

每次都跟中間位置之Data比較:

1. “==”: 找到
2. “<”: 落在左半部
3. “>”: 落在右半部

N筆Data用Binary Search 找X最多比較次數?

Ans:最多比較次數 = 樹高，而Decision Tree是Binary Tree，且高度最小化，

所以n個Data最小高度=log2(n+1)取上限，也就是等於最多比較次數

N筆Data以Binary Search Tree Search X:

Worst case: O(n)，for skewed tree

Best case: O(logn)

N筆Data以Binary Search，Search X:

Worst case: O(logn)

Sort部分

Internal and External Sort

Internal Sorting: 資料量少，可以一次全部置於memory中，進行排序工作

External Sorting: 資料量太大，無法一次全部置入memory中，必須借助外部儲存體(for Disk)保存，再行排序工作

常用的External Sorting Method: Merge Sort、Selection Tree、m-way search Tree、B tree

Stable and Unstable Sorting Method

定義:

在input Data中，可能會存在多個鍵值相同之Data

例: input: …,k,k\*,….

排序後，若保證:…k,k\*,… => 此sorting method = stable

反之，k\*可能會在k前面 => 此sorting method = unstable

Note:

1. Unstable代表產生不必要的SWAP
2. 所有Unstable sorting method必定比 stable sorting method還慢
   * False，因為quicksort為unstable, but 初等排序:Insert、Bubble皆為stable，但都比quicksort慢

初等排序:

Insertion Sort(插入排序)

觀念: i = 2 to n 做(n-1)回合

將第i筆Data插入到前面(i-1)筆排好之串列中之正確位置，形成n筆排好之串列

Time:

1.Best case:O(n) => 當input Data剛好是小->大呈現時

2.Worst case:O(n平方) => 當input Data剛好是大->小呈現時

3.Avg case:O(n平方)

Space:

O(1)

Stable

Selection Sort(選擇排序)

觀念:

自第i到n筆Data中，找出最小值，然後再跟第i筆做交換

做(n-1)個合併

Time:

Best、Avg、Worst = O(n平方)

說明:任何情況下，A[j]<A[min]的比較次數皆為(n-1)+(n-2)+.....+(1) = O(n平方)

Space:

額外空間需求:固定O(1)，與n無關

Unstable

補充:

1. Selection Sort每一回合，頂多SWAP一次
2. 適用在排序大型紀錄(紀錄是由很多欄位組成)上 for database

Bubble Sort(氣泡排序)

定義:

由左而右，依序兩兩相互比較，若前者>後者，則SWAP(前、後)

每一回合的效果:最大值開到最高位置，上述回合數頂多做到(n-1)回合，因為如果某一回合如果都沒有SWAP，則代表Sort完成，可直接Exit

Time:

Best case:O(n)，當input為小->大呈現時

Worst case:O(n平方)，當input為大->小呈現時

Avg case:O(n平方)

Space:

O(1)，額外空間需求是固定的 => flag、Swap函數之temp

Stable