作業五 report

b05901168 陳冠豪

一、資料結構的實做

1. Dlist 是由許多 node 所組成,利用每個 node 除了紀錄資料之外,還會紀錄下一個 node 的 pointer, 利用這個 pointer 把所有的 node 串在一起,優點是插入或刪除 node 很容易,且 node 的記憶體空間不需要連續,但缺點是讀取時需要重頭開始讀,而且要存取下一個 node 需要讀取 pointer 所指的位置,相較連續的 array 較慢.
實作的方法,如上所述,每個 node 會有下一個 node 及下一個 node 的 pointer,並且加入一個 dummy node, 此 dummy 的 next 是 list 的 head,prev 是 list 的 tail. 可以藉由此 node 來判斷此 array 是否為 empty 和是否尋訪到 list 的結尾.當 list 需要 sort 的時候,我是採用 bubble sort 的方法,考量的點是因為 bubble sort 不需要額外的記憶體空間來暫存,可以直接利用 node 的 data 交換即可完成 sort,雖然速度會比 merge sort 還要慢.

2.array 是把資料存在一個連續的記憶體空間,因此一開始要決定 array 的大小,如果資料增加到超過 array 的大小時,需要重新分配一塊記憶體,並把資料複製過去,優點為存取很容易且可以隨機存取,缺點為插入及移動資料較麻煩
實作的方法,使用動態分配記憶體,每次 data size=array capacity 時,就要一個兩倍大的記憶體,並把資料複製過去,而在刪除 array data 時,為了方便,我的實作方法為直接把 array 最後的 data 填入.

3.bst 是一種 tree,這種 tree 的每個 node 最多只能有兩個 child,且在左邊的 child 一定比該 node 小,反之再右邊的比 node 大,這種資料型態的優點為隨時都是 sort 好的,小的會在左邊,大的會在右邊,因此在存取時需要的時間會比 Dlist 還要短,缺點是需要額外的空間紀錄 node 之間的連結.
實作方法,每個 node 紀錄了 parent,left child,right child, 並有一個 dummy node 當作 tree 的 end,使用 parent 的理由是當作一些 tree 的 search 時比較直觀,但缺點是要 maintain parent 真的有點麻煩,常常會更新 parent 的值造成 bug , 而 dummy node 也會造成一些可能的 bug, 但是用這兩個方法寫起來 bst 結構就有點像 dlist, 讀起來蠻直觀的.

二、 實 驗 比 較

分別對這些資料結構作 push_back,delete,find,sort 並紀錄所花的時間(每個都有 100000 筆資料)

結果預測:
push_back: 我覺得 bst 會最久,因為每次要插入 node 時都要找到適當的 insert 地方,最多可能要花 log(height) 的時間,而 dlist 跟 array 我覺得應該差不多.

sort: 我覺得應該是 list 最久,因為他的複雜度為 log(n^2),而 bst 因為本身就是 sort 好的,因此為 0

find: 我覺得應該是 list 最久,因為他要從第一個 node 開始找起,而 bst 應該最快.

Delete: 我覺得應該是 bst 最久, 因為每次刪除 node 還需要考慮 tree 的結構,所以會花較長的時間,而 array 應該會最快

實驗結果:

	push_back(100000)	sort	Find	Delete (random x100)
Dlist	0.01s	62.13	0.03	0.02
Array	0.02s	0.05	0.01	<0.01
Bst	0.04s	0	<0.01	0.12

結果比較與討論:

由結果來看跟預測的差不多,除了 push_back,list 時間為 array 兩倍,我猜應該是因為 array 一開始大小是 0,因此要長到 100000 需要經過多次的記憶體複製.

而效能方面,可以看出 link list sorting 的時間很長,因此如果資料常常需要 sorting,不建議用 link list,可以考慮用 bst 來存資料,而如果資料需要多次的查找和存取而不需要常常 sorting 的話,其實用 array 效果會不錯,其在上述的比較來看,綜合的表現是最好的.