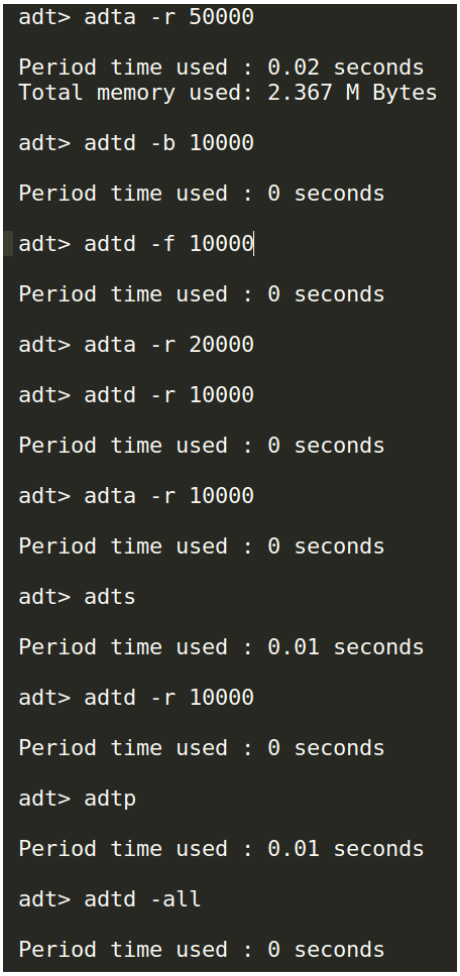
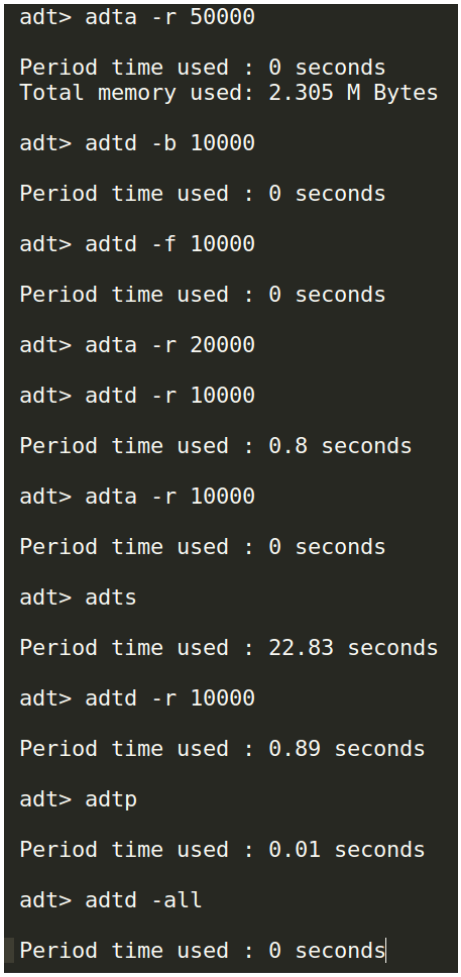
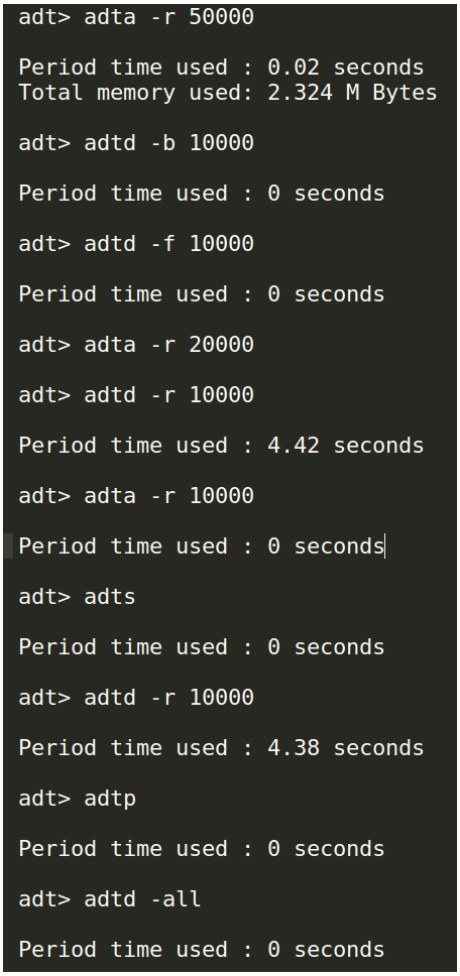


Array ↑



Dlist ↑



BST ↑

- Adta -r 50000:

  Array 需要 expand 整整 16 次才能容納 50000 筆資料,BST 需要判斷單筆資料要放置的位置，因此，相較於 Dlist 要花上較多時間。如果資料量顯著的上升，預計 array 所花費的時間要比 BST 更久，因為每次 expand 都要搬動所有資料。而耗費記憶體大小在此情況相差不遠，但 array 實際存有資料的空間並未到達 2.3M，大約有 15000 多個空餘空間，而 Dlist 每個 cell 要多存前後兩個指標，BST 更是要存取三個(_parent、_left、_right)。

- Adtd -b 10000 & Adtd -f 10000:

  三者在 pop_back 時都為 O(1)，前兩者在 pop_front 時也為 O(1)，因此不用耗費多時。但 BST 在 pop_front 時，要呼叫 begin()這個 iterator，如果是一路向左的 tree(worst case)，時間複雜度我猜測應為 O(n)，但由於看不見更小位數，在此無法確定是否較久。

- Adtd -r 10000:

  Dlist 在此要不斷的讓 iterator 進行跑動，並在刪除後重新連接前後的 pointer，因此需要耗費一些時間，但是相較於 BST 要訪問每個 node，並在判斷其 successor 的狀態後才能刪除，並重新連接，需要花費更大量的時間，如果是 worst case，我由於 successor 要遍歷所有 node，複雜度應該為 O(n)。

- Adts:

  此處三者皆為開-O3 在 compile，array 的 sort 為內建，Dlist 為$O(n^2)$的 bubble sort。理論上在排序後，array 的 find 耗費時間會小於 Dlist，但由於為了建立大量測資而使用 random 產生，不好進行特定資料刪除，因此僅能大略推測。

- Adtp:

  此處是為了檢查 iterator 跑動所有資料所花費的時間，比較意外的是 BST 與前兩者時間相差不多，為了盡量去除由於前面操作的影響，後來重新跑了 1000000 資料，array 與 DList 大約都在 0.2 秒左右，而 BST 則來到 0.3 秒多，較符合 iterator 在++時判斷所需多耗費的時間。

（附註：以上每個指令結束都有跑 usage，但為了符合螢幕大小，故予以刪除）

	Array	Dlist	BST
push_back	O(1), if no expand	O(1)	X
pop_front	O(1)	O(1)	O(n)
pop_back	O(1)	O(1)	O(1)
size	O(1)	O(n)	O(1)
empty	O(1)	O(1)	O(1)
insert(data)	X	X	O(n)
erase(pos)	O(1)	O(1)	O(n)
find(data)	O(n), if not sort	O(n)	O(n)