2025 Data Structure HW4

A1125531 吳俊霆

1.

By utilizing a template linked list I created 2 lists with structures Album & Track containing “Album name, list of tracks” and “Track name, play count” respectively. The idea of my list is to create a class containing a private head pointer which type is a node structure I defined including a template defined data. For album methods I just went with the public functions, but for the track methods I nested the functions by first calling the find album function to locate the desired node and then do the actions on the track linked list within the album node.

Play: I first called the album level play which calls the track level play which then prints out the first track in the list.

Next: If the album has an empty track it’ll delete it. Otherwise it’ll pop the top of the track list.

Push: Create a new node then locate the desired position to insert.

Remove: By iterating down the list and remembering the previous node once the current node hits the end the previous node’s next would be set to nullptr or the address of the node after the deleted node hence deleting a node.

Reverse: Using 3 node pointers I first store the next node, then set the current node’s next to be the address of the previous node, then set the previous node to be the current node, then set the current node to be the next node stored in the first place, looping through and reversing the direction of the next pointers.

2.

Using recursion, I used the getmiddle function to halve the linked list, then sort the halves repeating the process until the list is sorted to then be merged in the end.

The time complexity of the linked list using merge sort is O(n logn) on average, same goes for a normal array. Merge sort uses divide & conquer splitting the lists into logn levels combining O(n) resulting in a time of O(n logn).

For space complexity linked lists outperform arrays when merging because normal arrays need a temp array taking up O(n) space whereas O(1) for linked lists where only pointers are changed.

For the linked list merge sort I choose bogo sort as a comparison to emphasize the massive time difference between the 2.

3.

AI helped me debugged and aided in programming the code to the comparison between bogo & merge sort. Ai also helped find out the linker errors caused by template functions not being in the same header file. These errors can be fixed by using Explicit Instantiation.

4.

這次資料結構作業讓我練習了 Template、Linked List、資料管理與排序等觀念。在 P1 的實作中，我學習如何用單向 Linked List 模擬專輯與歌曲的結構，並處理各種操作指令，增進了我對鏈結串列的操作熟悉度。P2 的 Merge Sort 則讓我實際了解遞迴、分治法的實作方式與其優點，也透過比較不同排序法進一步體會時間複雜度的重要性。

作業過程中最大的挑戰是維護資料結構的正確性，尤其是刪除節點與記憶體釋放的處理。Debug 時我也有使用 AI 輔助，幫助我釐清邏輯錯誤。

總體而言，這次作業讓我學到很多，也提升了解決問題的能力。