Thời gian còn lại 0:21:24

Câu hỏi 1

Không hoàn thành

Chấm điểm của 2,00

The best way to sort a singly linked list given the head pointer is probably using merge sort.

Both Merge sort and Insertion sort can be used for linked lists. The slow random-access performance of a linked list makes other algorithms (such as quick sort) perform poorly, and others (such as heap sort) completely impossible. Since worst case time complexity of Merge Sort is O(nLogn) and Insertion sort is O(n^2), merge sort is preferred.

Additionally, Merge Sort for linked list only requires a small constant amount of auxiliary storage.

To gain a deeper understanding about Merge sort on linked lists, let's implement **mergeLists** and **mergeSortList** function below

Constraints:

0 <= list.length <= 10^4
0 <= node.val <= 10^6

Use the nodes in the original list and don't modify ListNode's val attribute.

```
struct ListNode {
    int val;
    ListNode* next;
    ListNode(int _val = 0, ListNode* _next = nullptr) : val(_val), next(_next) { }
};

// Merge two sorted lists
ListNode* mergeSortList(ListNode* head);

// Sort an unsorted list given its head pointer
ListNode* mergeSortList(ListNode* head);
```

For example:

Test	Input	Result

`int arr1[] = {1, 3, 5, 7, 9};`
`int arr2[] = {2, 4, 6, 8};`
`unordered_map<ListNode*, int> nodeAddr;`
`ListNode* a = init(arr1, sizeof(arr1) / 4, nodeAddr);`
`ListNode* b = init(arr2, sizeof(arr2) / 4, nodeAddr);`
`ListNode* merged = mergeLists(a, b);`
`try {`
`    printList(merged, nodeAddr);`
`}`
`catch(char const* err) {`
`    cout << err << '\n';`
`}`
`freeMem(merged);`
		1 2 3 4 5 6 7 8 9

Test	Input	Result

`int size;`
`cin >> size;`
`int* array = new int[size];`
`for(int i = 0; i < size; i++) cin >> array[i];`
`unordered_map<ListNode*, int> nodeAddr;`
`ListNode* head = init(array, size, nodeAddr);`
`ListNode* sorted = mergeSortList(head);`
`try {`
`    printList(sorted, nodeAddr);`
`}`
`catch(char const* err) {`
`    cout << err << '\n';`
`}`
`freeMem(sorted);`
`delete[] array;`
	9
	9 3 8 2 1 6 7 4 5
		1 2 3 4 5 6 7 8 9

Answer: (penalty regime: 0 %)

Reset answer

```
// You must use the nodes in the original list and must not modify ListNode's val attribute.
// Hint: You should complete the function mergeLists first and validate it using our first testcase exampl

// Merge two sorted lists
ListNode* mergeLists(ListNode* a, ListNode* b) {
    return nullptr;
}

// Sort and unsorted list given its head pointer
ListNode* mergeSortList(ListNode* head) {
    return nullptr;
}
```

Precheck Kiểm tra