

|  |  |
| --- | --- |
|  |  |
| **JIANGSU UNIVERSITY** |

数据结构作业

2023-2024-1

学 号： 3220604077

姓 名： 唐熙晟

班 级： 信息安全2203

2023年 9 月 17 日

# 题目1

## 1 题目描述

设有一个双链表h，每个结点中除了有prior .data和next几个域以外,还有一个访问频度域freq，在链表被启用之前，其值均初始化为0。每当进行LocateNode(h,x)运算时，令元素值为x的结点中freq域的值加1,并调整表中结点的次序,使其按访问频度的递减次序排列，以便使频繁访向的结点总是靠近表头。试编写一个符合上述要求的LocateNode运算的算法。

## 2 系统功能

双链表且有排序，访问频度等功能

## 3运行界面截图与说明

5：访问次数

8：freq8:1

4：freq4:1

13：freq13:1

8：freq8:2

7：freq7:1

## 4 代码

1. #include <bits/stdc++.h>
2. **using** **namespace** std;
4. **struct** ListNode
5. {
6. **int** data;
7. ListNode\* next;
8. ListNode\* prior;
9. **int** freq;
11. ListNode(**int** data) : data(data), next(nullptr), prior(nullptr), freq(0) {}
12. };
14. **class** DoubleLinkList {
15. ListNode\* head;
17. **public**:
18. DoubleLinkList() : head(nullptr) {}
20. **void** add(**int** data) {
21. ListNode\* NewNode = **new** ListNode(data);
22. **if** (head == nullptr) {
23. head = NewNode;
24. }
25. **else** {
26. ListNode\* currentNode = head;
27. **while** (currentNode->next != nullptr) {
28. currentNode = currentNode->next;
29. }
30. currentNode->next = NewNode;
31. NewNode->prior = currentNode;
32. }
33. }
35. **void** printList() {
36. **if** (head == nullptr) {
37. cout << "List is empty.\n";
38. }
39. **else** {
40. ListNode\* currentNode = head;
41. **while** (currentNode != nullptr) {
42. cout << currentNode->data << " ";
43. currentNode = currentNode->next;
44. }
45. cout << endl;
46. }
47. }
49. ListNode\* LocateNode(**int** x) {
50. **if** (head == nullptr)
51. **return** nullptr;
52. **else** {
53. ListNode\* currentNode = head;
54. **while** (currentNode != nullptr) {
55. **if** (currentNode->data == x) {
56. currentNode->freq++;
57. swapdata(currentNode);
58. **return** currentNode;
59. }
60. currentNode = currentNode->next;
61. }
62. **return** nullptr;
63. }
64. }
66. **void** swapdata(ListNode\* node) {
67. ListNode\* currentNode = node->prior;
68. **while** (currentNode != nullptr && node->freq >= currentNode->freq) {
69. swapfreq(currentNode, node);
70. currentNode = currentNode->prior;
71. node = node->prior;
72. }
73. }
75. **void** swapfreq(ListNode\* n1, ListNode\* n2) {
76. **int** tempdata = n1->data;
77. n1->data = n2->data;
78. n2->data = tempdata;
80. **int** tempfreq = n1->freq;
81. n1->freq = n2->freq;
82. n2->freq = tempfreq;
83. }
84. };
86. **int** main() {
87. DoubleLinkList list;
88. list.add(5);
89. list.add(3);
90. list.add(7);
91. list.add(13);
92. list.add(4);
93. list.add(8);
95. list.printList();
97. **int** m,n;
98. cin >> m;
99. **for** (**int** i = 0; i < m; i++) {
100. cin >> n;
101. ListNode\* node = list.LocateNode(n);

104. list.printList();
105. }
106. **return** 0;
107. }