REPORT BIG ASSIGNMENT
Chương 2

Họ tên: Nguyễn Anh Tuấn
MSSV: 20200400

Requirement:

Cài đặt thuật toán MergeSort bằng danh sách liên kết kép

Bài làm:

Code:

Khởi tạo cấu trúc cho Node

```
12  #include <iostream>
13  using namespace std;
14
15  struct Node {
16      int data;
17      Node *prev;
18      Node *next;
19  };
```

Hàm thêm một node mới với data vào sau danh sách đã có:

```
21  void insertNodeAtTail(Node** head, int data) {
22      Node* newNode = new Node();
23      newNode->data = data;
24      newNode->prev = nullptr;
25      newNode->next = nullptr;
26      if (*head == nullptr) {
27          *head = newNode;
28      } else {
29          Node* current = *head;
30          while (current->next != nullptr) {
31              current = current->next;
32          }
33          current->next = newNode;
34          newNode->prev = current;
35      }
36  }
```

Hàm trả về giá trị tại vị trí cụ thể trong danh sách:

```
38  int dataAtIndex(Node** head, int idx){
39      int pos = 0;
40      Node* tmp = *head;
41      while ( pos != idx ){
42          tmp=tmp->next;
43          pos++;
44      }
45      return tmp->data;
46  }
```

Hàm thay đổi data tại vị trí cụ thể

```
48  void changeDataAtIndex(Node** head, int idx, int data){
49      int pos = 0;
50      Node* tmp = *head;
51      while ( pos != idx ){
52          tmp=tmp->next;
53          pos++;
54      }
55      tmp->data = data;
56  }
```

Hàm trả về số lượng phần tử trong danh sách

```
58  int sizeOfNode(Node** head){
59      int res = 0;
60      Node* tmp = *head;
61      while (tmp != NULL){
62          res++;
63          tmp=tmp->next;
64      }
65      return res;
66  }
```

Hàm in node ra màn hình

```
68  void log(Node* head) {
69      Node* current = head;
70      while (current != NULL) {
71          cout << current->data << " ";
72          current = current->next;
73      }
74      cout << endl;
75  }
```

Hàm mergeSort và merge sử dụng cho danh sách liên kết cùng với sự hỗ trợ của các hàm trên:

```
77  void merge(Node** head, int l, int m, int r) {
78      int i, j, k;
79      int n1 = m - l + 1;
80      int n2 = r - m;
81      Node *L = NULL, *R = NULL;
82      for (i = 0; i < n1; i++) insertNodeAtTail(&L,dataAtIndex(head, l+i));
83      for (j = 0; j < n2; j++) insertNodeAtTail(&R,dataAtIndex(head, m + 1 + j));
84      i = 0; j = 0; k = l;
85      while (i < n1 && j < n2) {
86          if (dataAtIndex(&L,i) <= dataAtIndex(&R,j)) {
87              changeDataAtIndex(head,k,dataAtIndex(&L,i));
88              i++;
89          } else {
90              changeDataAtIndex(head,k,dataAtIndex(&R,j));
91              j++;
92          }
93          k++;
94      }
95      while (i < n1) {
96          changeDataAtIndex(head,k,dataAtIndex(&L,i));
97          i++; k++;
98      }
99      while (j < n2) {
100         changeDataAtIndex(head,k,dataAtIndex(&R,j));
101         j++; k++;
102     }
103 }
104
105 void mergeSort(Node** head, int n) {
106     for (int i = 1; i <= n-1; i = 2*i) {
107         for (int j = 0; j < n-1; j += 2*i) {
108             int m = min(j + i - 1, n-1);
109             int re = min(j + 2*i - 1, n-1);
110             merge(head, j, m, re);
111         }
112     }
113 }
```

Kết quả chụp màn hình :

```
115  int main() {
116      Node* head = NULL;
117      insertNodeAtTail(&head, 7);
118      insertNodeAtTail(&head, 3);
119      insertNodeAtTail(&head, 9);
120      insertNodeAtTail(&head, 5);
121      insertNodeAtTail(&head, 1);
122      insertNodeAtTail(&head, 8);
123
124      cout << "Danh sach truoc khi sort: \n";
125      log(head);
126      mergeSort(&head,sizeOfNode(&head));
127      cout << "Danh sach sau khi sort: \n";
128      log(head);
129      return 0;
130  }
```

```
(base) macad@nganhhtuann-3 output % ./"doublyLinkedList"
Danh sach truoc khi sort:
7 3 9 5 1 8
Danh sach sau khi sort:
1 3 5 7 8 9
```

Nhận xét:

Cài đặt thuật toán phức tạp

Khử đệ quy so với cách phổ thông

Tuy nhiên việc triển khai khó khăn và tài nguyên , cụ thể:

- Độ phức tạp thuật toán về thời gian vẫn giữ ở O(nlog(n)) với n là số lượng phần tử trong node
- Trong khi đó độ phúc tạp về không gian là O(n)
- Thuật toán này áp dụng cách tiếp cận chia để trị: chia mảng thành 2 nửa rồi gom chúng lại
- Độ phúc tạp về thời gian đã được qua cắt giảm chi phí tìm kiếm data của node và thay đổi data cũng nhưng chèn node vào 2 mảng phụ đó.