数据结构解题报告

曾慧敏

专业：17 电子信息工程
学号：17020022049

目录

合并非降序链表

1、实验目的

通过构造出两个非降序的链表序列 M1 和 M2 合并后的非降序链表，了解链表分类和特性，掌握链表构造及其应用方法。

关键词：链表 非降序 合并链表

2、实验内容

输入两个非降序的链表序列 M1 和 M2 作为已知链表，构造出它们合并后的非降序链表。

基本步骤:

（1）定义链表结构

（2）构造合并链表函数

（3）构建链表输入程序（输入提示，考虑非法输入的情况）

（4）测试程序

3、实验输入和输出说明

• **Input：**两个非降序的链表序列 M1 和 M2。（相邻节点元素值用“,”隔开，链表以“;”作为结束标志）

• **Output：**合并后的非降序链表（输出格式与输入链表的格式相同）

• **Example：**

Sample Input:

M1=1,2,4;

M2=1,3,4;

Sample Output:

M3=1,1,2,3,4,4;

• 测试中输入样例的考虑因素：

（1）两个正常非降序列

M1= 1,2,4;

M2= 1,3,4;

（2）两个序列长度不相等

M1= 1,3,4,10,19,20;

M2= 1,2,9,10;

（3）序列中含有重复的数字

M1= -12,0,1,1,3,4;

M2= 9,10,11,343,2090;

（4）含有降序列

M1= 0,3,5,2,1;

M2= 0,6,7,9,8,7,6;

4、解题思路（时间复杂度 O(n)）

（1）在取空间最优的时候可能比较麻烦，故可以建立一个新的单链表 M3，用来存放合并后的结果。

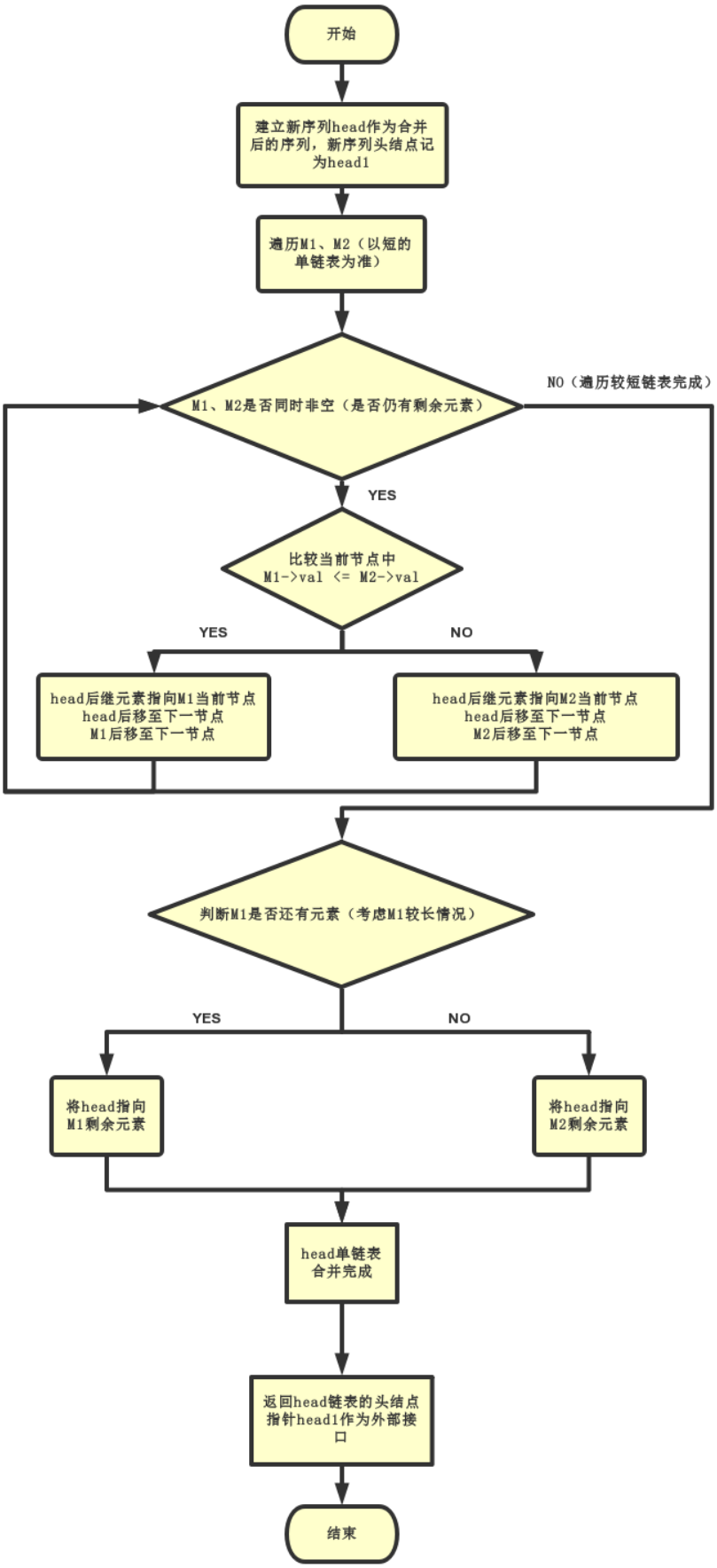
（2）遍历已知链表 M1 和 M2 的所有节点，每次都选取较小的那个元素值记录到 M3 中，同时小元素所在链表当前节点后移一位，继续比较，直到遍历完较短的链表。以保证 M3 中元素排列符合非降序的条件。

（3）由于 M1,M2,M3 都是按照非降序排列的，则当较短链表遍历完时，若 M1 或 M2 有元素剩余，则可以肯定这些元素均大于 M3 此时的所有元素，故直接将剩余部分连接到 M3 尾部即可

5、实验代码及注释

```
#include<iostream>
#include <string>
#include<stdlib.h>
using namespace std;
/**
* Definition for singly-linked list.
* struct ListNode {
* float val;
* ListNode *next;
* ListNode(int x) : val(x), next(NULL) {}
* };
**/
//########## 定义单链表结构 ########
class ListNode
{
public:
int val;
ListNode *next;
ListNode(int x) : val(x), next(NULL) {}
};
//########## 创建测试样例 ##########
ListNode * create_test_case(){
ListNode * l=new ListNode(0);
ListNode * l_head=l;
float val=0.0;
float pre=-12131;
char str=',';
while(str!=';'){
cin>>val>>str;
if(val<pre){
cout<<"invalid input!! the input has to be bigger than the previous one!!"<<endl;//考虑输入非法情况（即不按照非降序排列输入）
cout<<"but if we exchange the invalid element..."<<endl;
}
else{
pre=val;
ListNode * M1= new ListNode(val);
l->next=M1;
l=l->next;
}
}
return l_head->next;
};
//########## 合并序列 ##########
ListNode* mergeTwoLists(ListNode* M1, ListNode* M2) {
ListNode* head=new ListNode(0) ;//新建 head 作为合并后的序列
ListNode* head1=head; //令 head1为单链表 head 的头指针，作为对外接口
while(M1!=NULL && M2!=NULL){ //判断 M1 val< M2 val 是否成立
if(M1->val<=M2->val){
head->next=M1;
head=head->next;
M1=M1->next; //head 序列连接较小元素
}
else{ //M1 val>M2 val
head->next=M2;
head=head->next; //head 序列连接较小元素
M2=M2->next;
}
}
if(M1!=NULL)
head->next=M1; //判断所给两个链表是否还有剩余元素
else
head->next=M2;
return head1->next;
};
int main(){
cout<<"ATTENTION! elements of list have to be divided by ',' and end with ';'！ "<<endl;//输入提示
cout<<"please input list M1:"<<endl;
ListNode* M1=create_test_case();//输入第一个链表M1所包含元素
cout<<"please input list M2:"<<endl;
ListNode* M2=create_test_case();//输入第二个链表M2所包含元素
ListNode* M3=mergeTwoLists(M1,M2);
cout<<"the merged list M3 is:"<<endl;
while(M3->next!=NULL){
cout<<M3->val<<',';
M3=M3->next;
}
cout<<M3->val<<";"<<endl;
cout<<"\n"<<"\n"<<endl;
return 0;
}
```

6、合并链表步骤流程图


开始
建立新序列head作为合并后的序列，新序列头结点记为head1
遍历M1、M2（以短的单链表为准）
M1、M2是否同时非空（是否仍有剩余元素）
NO（遍历较短链表完成）
YES
比较当前节点中M1->val <= M2->val
YES
NO
head后继元素指向M1当前节点
head后移至下一节点
M1后移至下一节点
head后继元素指向M2当前节点
head后移至下一节点
M2后移至下一节点
判断M1是否还有元素（考虑M1较长情况）
YES
NO
将head指向M1剩余元素
将head指向M2剩余元素
head单链表合并完成
返回head链表的头结点指针head1作为外部接口
结束

7、实验运行结果分析

Input1:

M1=1,2,4;
M2=1,3,4;

Output1:

```
ATTENTION! elements of list have to be divided by ',' and end with ';'!
please input list M1:
1,2,4;
please input list M2:
1,3,4;
the merged list M3 is:
1,1,2,3,4,4;
```

Input2:

M1=1,3,4,10,19,20;
M2=1,2,9,10;

Output2:

```
ATTENTION! elements of list have to be divided by ',' and end with ';'!
please input list M1:
1,3,4,10,19,20;
please input list M2:
1,2,9,10;
the merged list M3 is:
1,1,2,3,4,9,10,10,19,20;
```

Input3:

M1=-12,0,1,1,3,4;
M2=9,10,11,343,2090;

Output3:

```
ATTENTION! elements of list have to be divided by ',' and end with ';'!
please input list M1:
-12,0,1,1,3,4;
please input list M2:
9,10,11,343,2090;
the merged list M3 is:
-12,0,1,1,3,4,9,10,11,343,2090;
```

Input4:

M1=0,3,5,2,1;
M2=0,6,7,9,8,7,6;

Output4:

```
ATTENTION! elements of list have to be divided by ',' and end with ';'!
please input list M1:
0,3,5,2,1;
invalid input!! the input has to be bigger than the previous one!!
but if we exchange the invalid element...
invalid input!! the input has to be bigger than the previous one!!
but if we exchange the invalid element...
please input list M2:
0,6,7,9,8,7,6;
invalid input!! the input has to be bigger than the previous one!!
but if we exchange the invalid element...
invalid input!! the input has to be bigger than the previous one!!
but if we exchange the invalid element...
invalid input!! the input has to be bigger than the previous one!!
but if we exchange the invalid element...
the merged list M3 is:
0,0,3,5,6,7,9;
```

8、实验收获总结

• 在合并链表时如果为了不增加空间复杂度，则可以只在 **M1,M2** 的基础上进行增删节点的操作，但这样的操作注意要考虑到增删节点之后指针位置的变化。
• 在进行增删节点操作时为简便起见，可以为链表添加一个头指针。