Data Structure

Homework2: 連結串列交換首尾與反轉

Author: B11002220 CHI-CHUN, LO

Date: October 7, 2023

國立臺灣科技大學

NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

NTUST ECE

Contents

1 Problem

Consider a linked list in which each node contains a word. Give a program that receives a word sequence and two integers. A and B, and constructs and prints the new list according to the following action:

1. Exchange the initial and final elements in the linked list.
2. Reverse a sublist starting from A and ending at B.

1.1 Definition

首先，根據題目要求，我們可以定義以下單向連結串列結構且每個結構內都要能存放單詞：

```
typedef struct node *listPointer;
typedef struct node
{
  char *word;
  listPointer next;
} node;

```

接著，我們可以定義一個函數交換首尾的單詞 Exchange，它的輸入是這個結構的頭，沒有輸出：

```
void exchange(listPointer first)

```

再來，討論如何反轉子串列。我們可以定義一個函數 Reverse，它的輸入是這個結構的頭，有輸出 1 或 0 表示是否完成反轉：

```
int Reverse(listPointer *first)

```

另外要特別注意由於反轉子串列的方式可能會涉及到串列的頭修改，因此這邊是用指標的方式去訪問。

1.2 Details

考慮各個函數的細節。

1.2.1 Exchange function

由於實體的交換函數考慮的難度比較複雜，因此我們先討論交換節點的內容。exchange 函數它的輸入是串列的頭指標，因此我們需要做下面幾件事情完成這項功能:

1. 首先，考慮是否為空串列，如果是空串列，則直接返回。
2. 再來，用一個指標指向最後一個節點
3. 最後，交換首尾節點的內容

1.2.2 Reverse function

我們的 Reverse 函數的輸入是串列的頭指標。我們需要考慮多種情況，來滿足題目要求。下面是幾種可能會發生的情況；

1. 從 0 到 length-1 反轉，稱作完全反轉情況。
2. 從 0 到 length-i 反轉，這邊稱作不完全反轉情況。Note: 1 < i < length
3. 從 i 到 m 反轉，這邊稱作通用情況。Note: 這邊的 0<i<m<length。

針對從 0 到 length-1 反轉，我們可以透過討論反轉"I am a boy" 的文字進行 `reverse(&first,0,3)` 的情況來解決:

1. 先把第 i+1 的節點指向第 i 個節點直到第 i+1 的節點是 end 節點
2. 再把 first 指向的節點 end 指向的節點 (目的是為了把起頭改位置)
3. 最後把 start 節點指向 null 後一個節點

針對從 0 到 length-i 反轉，我們可以透過討論反轉"I am a boy" 的文字進行 `reverse(&first,0,2)` 的情況來解決:

1. 先把第 i+1 的節點指向第 i 個節點直到第 i+1 的節點是 end 節點
2. 再把 first 指向的節點 end 指向的節點 (目的是為了把起頭改位置)
3. 最後把 start 節點指向 end 後一個節點

最後，我們透過討論 begin 非 0 的反轉"I am a boy" 的文字進行 `reverse(&first,1,3)` 的情況來解決:

1. 先把第 i+1 的節點指向第 i 個節點直到第 i+1 的節點是 end 節點
2. 再把 start-1 節點指向的節點 end 指向的節點
3. 最後把 start 節點指向 end 後一個節點

因此把這三個情況都做出來，我們就可以完成這道題目。針對不同情況的圖片請參考下面的圖:

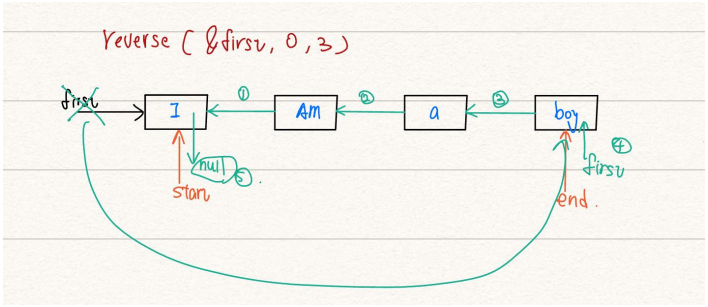

Figure 1: 0 - length-1 reverse situation

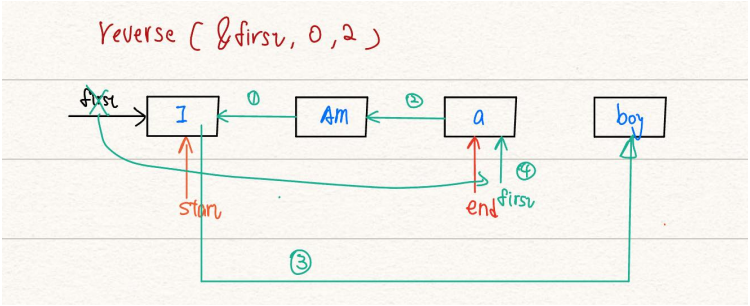

Figure 2: 0 - length-i reverse situation

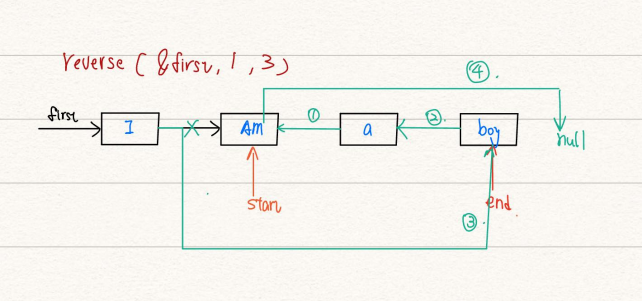

Figure 3: n - m reverse situation

2 Code

下面是針對 exchange() 函數的程式碼。最主要的功能就是先判斷是否為空的 list，再針對頭節點與尾節點的資料交換。如下:

```
void exchange(listPointer first)
{
  if (first == NULL || (first)->next == NULL)
  {
    return;
  }
  listPointer temp = first;
  for (; temp->next; temp = temp->next); // FIND THE LAST node
  // exchage the first one with last
  char* Ctemp = _strdup((first)->word);
  (first)->word = _strdup((temp)->word);
  (temp)->word = Ctemp;
}

```

第二部分是 reverse() 函數，主要功能是將 list 中的指定範圍的資料進行反轉。原則上功能大致分成四部分，首先會先將錯誤的資料排除，接者針對要反轉的範圍反轉，最後設定 start 跟 end 指向的位置。如下:

```
int reverse(listPointer* first, int begin, int end){
int count = 0;
listPointer node_before_start = NULL, node_after_end = NULL;
listPointer start_node = NULL, end_node = NULL, temp = NULL;
int length = get_length( *first);
// escape some error case
if (begin > end || length <= end)
  return 0;
else if (begin == end) /
  return 1;
// set 4 point
for (temp = *first, count = 0; temp; temp = temp->next)
{
  if (count == begin - 1)
    node_before_start = temp;
  if (count == begin)
    start_node = temp;
```

```
        if (count == end)
          end_node = temp;
        if (count == end + 1)
          node_after_end = temp;
        count++;
      }
      // start invert
      listPointer current, prev;
      current = start_node->next;
      prev = start_node;
      temp = NULL;
      // invert from start to end
      while (current != node_after_end)
      {
        temp = current->next; // null
        current->next = prev; // student ->null
        prev = current;       //
        current = temp;
      }

      if (begin == 0 && length - 1 == end) // case 0 -length
      {
        *first = end_node;
        start_node->next = NULL;
      }
      else if (begin == 0) // case 0-length-i
      { // 0 - length - i
          *first = end_node;
          start_node->next = node_after_end;
      }
      else // case other
      {
        node_before_start->next = end_node;
        start_node->next = node_after_end;
      }

      return 1;
            }

```

3 Result

在這邊我們會針對不同的例子去測試程式的穩定性、正確性。

- **example1:** 討論在正常情況下的運作從頭開始

  Input:

  ```
  0 3
  Person real value first
  ```

  Output:

  ```
  first real value Person
  Person value real first
  ```

解釋: 測試反轉 0-3 的功能是否正常，第二列的輸出 0-3 確實是第一列 0-3 的反轉

- **example2:** 討論在正常情況下的運作從第一個開始反轉
  Input:

```
1 3
Person real value first
```

  Output:

```
first real value Person
first Person value real
```

  解釋: 測試反轉 1-3 的功能是否正常，第二列的輸出 0-3 確實是第一列 0-3 的反轉

- **example3:** 超過範圍測試
  Input:

```
1 100
Person real value first
```

  Output:

```
first real value Person
-1
```

  解釋: 交換的功能不會影響，但是反轉的範圍超過我們就不動作!

- **example4:** 測試長度為 1 的句子反轉與交換是否都能輸出
  Input:

```
0 0
Person
```

  Output:

```
Person
Person
```

  解釋: 輸入單詞很短，只要範圍正確，不影響輸出結果。

4 Discussion

根據上述結果，我們可以針對每個例子的輸入、輸出和解釋進行分析和討論。

- **example1:** 針對開頭反轉
- **example2:** 針對非開頭反轉

- **example3:** 輸入超過範圍，則直接返回
- **example4:** 針對句子長度只有 1 反轉 (不會返回)

所以根據以上結果能大致囊擴大部分輸入的例子，但是我們的程式碼仍有一些小地方需要改進。 當前取得文字的函數我們目前是利用 **fgets(, length,)** 其中它需要我們先設定 input.txt 單行最多讀入的長度，但是這就造成另個問題理論上我們的 reverse 可以反轉 0 sizeof(int)-1 的長度，但是因為單行讀入的上限導致輸入的文字只能不超過上限 < 因此更不可能達到我們設定的理想範圍。唯一的解決方法就是將 fgets() 替換成 String.h 中的 getline，這個函數可以動態為輸入文字分配空間，繞過剛剛的文字上限導致的單詞上限。

5 Conclusion

最後，儘管這次的程式比上次難度略微增加，但是經過這次的練習後，我對 linklist 的認法有著更加深刻的理解。並明白資料結構是個必須搭配程式與畫圖的學科。可以先針對解決簡單的例子，在拓展到更複雜的例子查看是否可執行與遺漏。