ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2020

Homework 1

Due date: 23:59, September 14, 2020

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL Name to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
5. When submitting, match your solutions to the according problem numbers correctly.
6. No late submission will be accepted.
7. Violations to any of the above may result in zero grade.

1: Academic integrity (10 pts)

Please determine who violated academic plagiarism in the following situations. A stands for Alice, B stands for Bob, N stands for None, AB stands for Both.

(a) Alice and Bob are good friends, Alice asked if Bob could send her codes so that she could look at it(promising, of course, not copying). Then Bob sent his codes to Alice, Alice copied it and handled it in. AB

(b) Alice and Bob are roommates. Bob let Alice use his laptop to play PUBG, and Alice copied Bob's solution for assignment. AB

(c) Alice uses Bob's computer to complete the programming assignment and accidentally submits her own assignment using Bob's account. After that, Alice and Bob resubmitted their assignments using their own accounts. AB

(d) Bob is Alice's boyfriend. In order to promote feelings, they competed in the discussion room about who could complete the programming assignment first. There was no communication during the period. N

(e) Alice found the homework question in CSDN and copied the answer on her homework. A

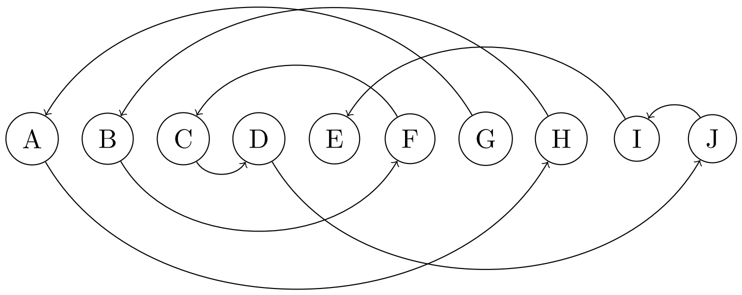
2: Array for list (10 pts)

In this question, we use two arrays: “next” and “value” to simulate a linked-list. The elements at each index in “next” and “value” are combined together to represent a node in the linked-list, where “next” represents the pointer to the next node, and “value” represents data in the nodes.

(a) **Linked list from array representation (5 pts)**

Here are the two arrays of “next” and “value”. Please draw the linked list presented by the two arrays.

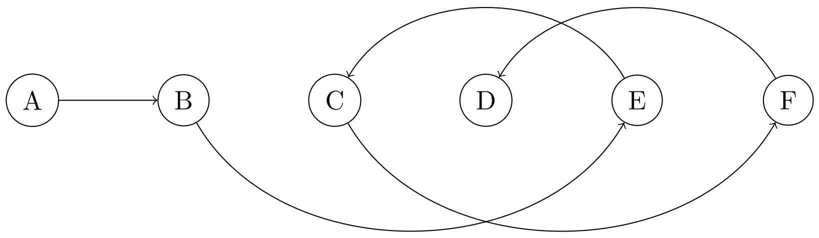
index	0	1	2	3	4	5	6	7	8	9
value	A	B	C	D	E	F	G	H	I	J
next	7	5	3	9	/	2	0	1	4	8



$$G \to A \to H \to B \to F \to C \to D \to J \to I \to E$$

(b) **Array representation for a linked list (5 pts)**

Fill in the “next” array in order to make the array and the linked-list showing below equivalent.



index	0	1	2	3	4	5
value	A	B	C	D	E	F
next	1	4	5	/	2	3

3: List operations (10 pts)

You should use the defined linked list to implement all the newly defined operations. Please pay attention to the following rules:

1. There is no dummy node as the entry node, i.e. if the linked list is not empty, `head` points to the first node, `tail` points to the last node.
2. Use `new` and `delete` to deal with memory issues.
3. Remember to maintain `head` and `tail`.
4. You may not use all of the blank lines. There is at most one statement (ended with `;`) in each blank line.
5. For simplicity, the retrieve function is not given and you can directly use the private attributes like `node->data`, `list->head` etc. (Invalid for programming but OK for this homework)
6. If there is any syntax error, this line will be counted as wrong.

```
class Node {
public:
    int data;
    Node * next;

    Node(int val, Node * nextNode): data(val), next(nextNode)
    {
    }
    ... ...
};
Node * head;
Node * tail;
```

(a) **Insert before (5 pts)**

This operation has been discussed in detail in the lecture slides. Given a new data and a target node, create a new node using the new data and insert it before the target node. You should implement this function with $O(1)$ time complexity.

- Assume the target node is neither `head` nor `tail`.

```
void InsertBefore(Node * curr, int data)
{
    Node * newNode = new Node(   curr->data,curr->next    );

    curr->next = newNode      ` or curr->data = data `      ;

    curr->data = data         ` or curr->next = newNode `   ;
}
```

(b) **Reverse (5 pts)**

Now think about how to reverse a list. Given `head` and `tail` of a non-empty list,

- Assume the list is not empty.

```
void Reverse(Node * &head, Node * &tail) {

    Node * prevNode = head;

    Node * rotateNode =     prevNode->next   ;

    head->next = NULL;

    while (rotateNode) {

        Node *next = rotateNode->next;

        rotateNode->next = prevNode          ;

        prevNode = rotateNode                ;

        rotateNode = next;

    }

    swap(head, tail);

}
```

(c) [**Bonus**] **Is there a cycle? (0 pts)**

Sometimes, a list may form a cycle, which means its tail is pointing at a node in the list instead of `NULL`.

$$\begin{tikzcd}
A \arrow[r] & B \arrow[r] & C \arrow[r] & D \arrow[r] & E \arrow[r] & F \arrow[d] \\
 & & J \arrow[u] & I \arrow[l] & H \arrow[l] & G \arrow[l]
\end{tikzcd}$$

Since the list may be extremely large, it is not allowed to record whether a node is visited – which means you could not use extra space bigger other than $O(1)$. Given the head of a list, describe how to determine if there is a cycle on the list.

(Note: tail and length are not given. If there is no cycle on the list, the last node is supposed to be pointing at `NULL`)

(Hint: Think about 3000 meters long run in the playground.)

Create two pointers pointing at `head`. Let one move one step at a time and the other move two steps at a time. If they meet in the trip, there is a cycle.