COMP10002 Foundations of Algorithms

Workshop Week 10

Wenbin Cao

October 10, 2019

GitHub Repo: https://github.com/AlanChaw/COMP10002-FoA

Linear Data Structures

- Linked Lists
- Stacks
- Queues

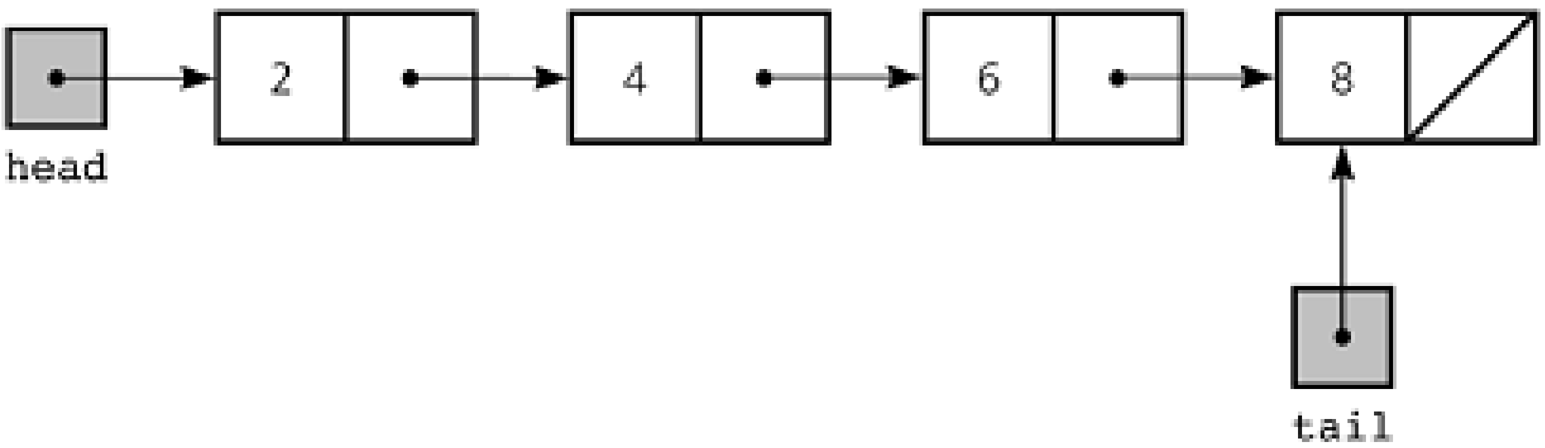
Trees

- Binary Search Trees

Dictionaries

Linked Lists

A linked list is a one-dimensional data structure in which objects are threaded together using a pointer in each node.



Linked Lists

Linked list node structure:

```
typedef struct node node_t;

struct node {
    data_t data;      /* Representational abstraction */
    node_t *next;
};
```

Linked list structure:

```
typedef struct {
    node_t *head;
    node_t *foot;
} list_t;
```

Linked Lists

Four Main Operations (mentioned in the lecture)

- Insert before head

- Append after tail

- Delete head node

- Fetch the contents of head

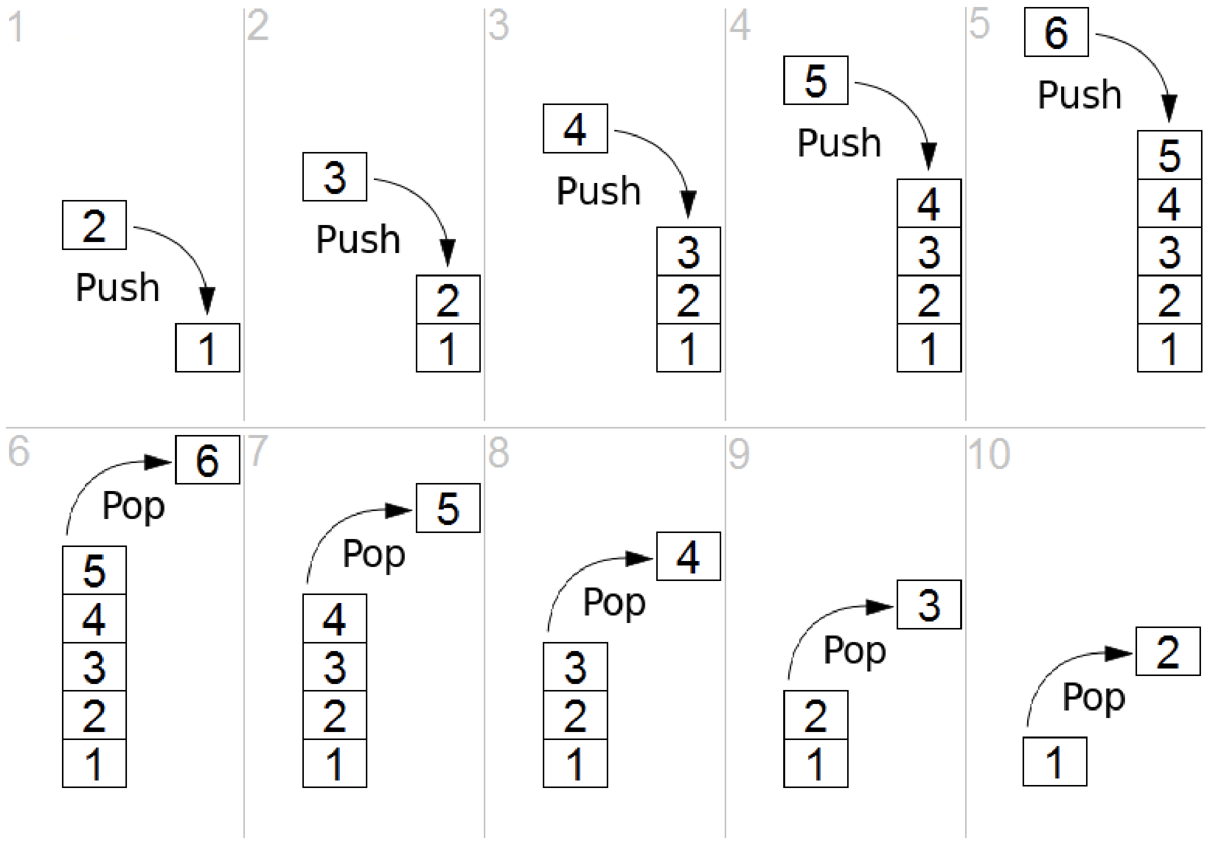
Comparing with Arrays

Stacks

- A stack is a data structure in which the most recently inserted item is returned next.

- LIFO - Last in, first out

- Two main operations: push and pop

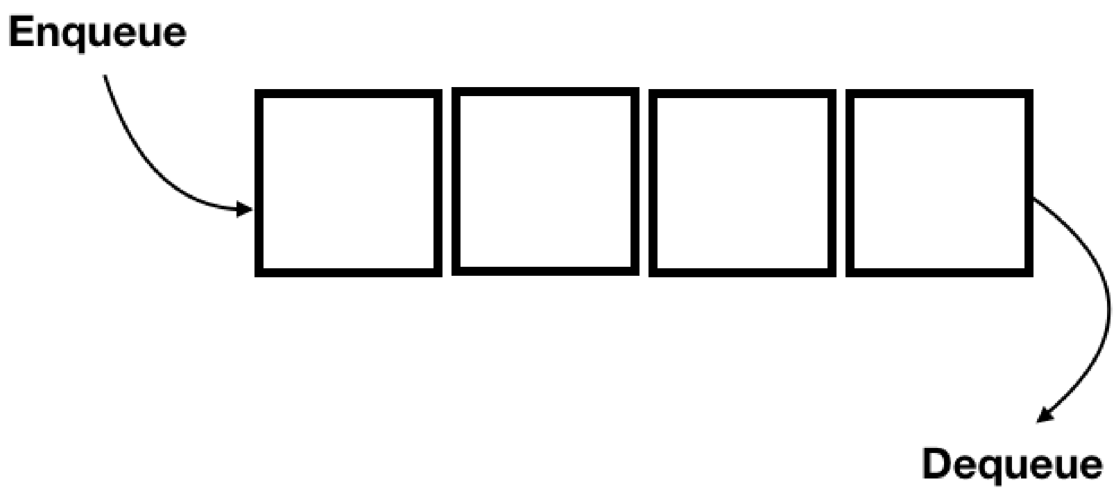


Queue

- A queue is a data structure in which the least recently inserted item is returned next.

- FIFO - First in, first out

- Two main operations: enqueue and dequeue



Discussion

Exercise 8

Stacks and queues can also be implemented using an array of type `data_t`, and static variables. Give functions for `make_empty_stack()` and `push()` and `pop()` in this representation.

Discussion

Exercise 9

Suppose that insertions and extractions are required at both head and foot. How can `delete_foot()` be implemented efficiently? (Hint, can a second pointer be added to each node?)

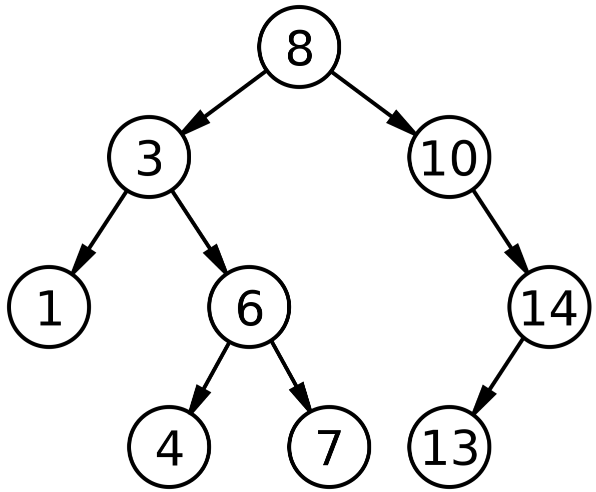
Exercise 9

```c
struct node {
    data_t data;
    node_t *next;
    node_t *prev;
};
```

```c
list_t
*delete_foot(list_t *list) {
    node_t *old_foot;
    assert(list!=NULL && list->foot!=NULL);
    old_foot = list->foot;
    list->foot = list->foot->prev;
    if (list->foot==NULL) {
        /* the only list node just got deleted */
        list->head = NULL;
    }
    free(old_foot);
    return list;
}
```

Binary Search Trees

- A binary tree is a two-dimensional data structure in which objects are threaded together using two pointers in each node.



- A binary search tree is a binary tree in which the objects are ordered from left to right across the tree.

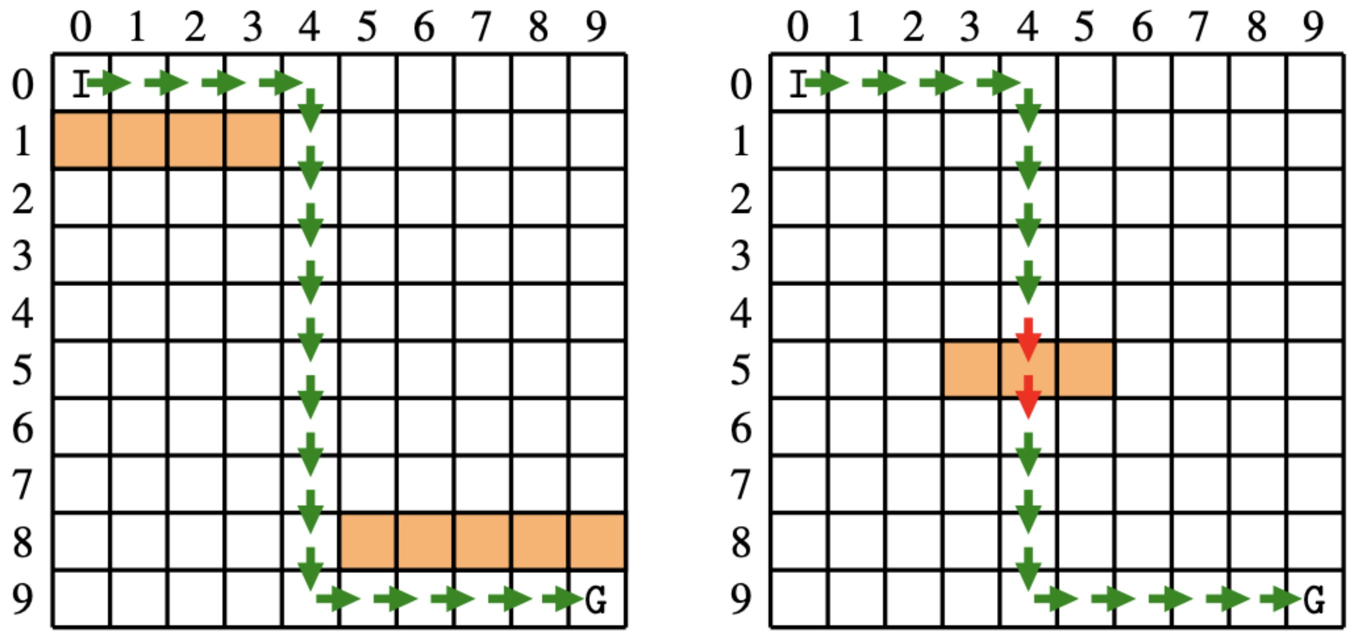
Binary Search Trees

```
typedef struct node node_t;
struct node {
void *data;
node_t *left;
node_t *rght;
};
```

```
typedef struct {
node_t *root;
int (*cmp)(void*,void*);    /* For polymorphic */
} tree_t;
```

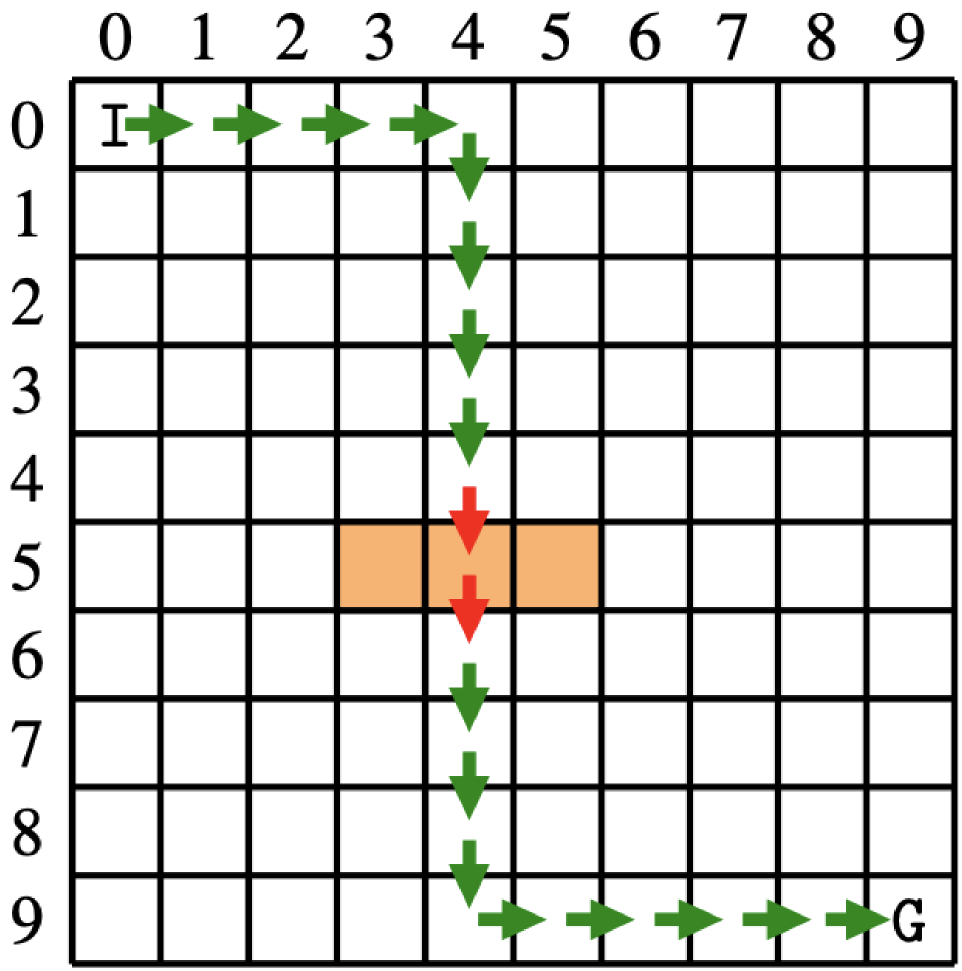
Assignment 2

- The algorithm to repair a broken route segment



Assignment 2

- The algorithm to repair a broken route segment



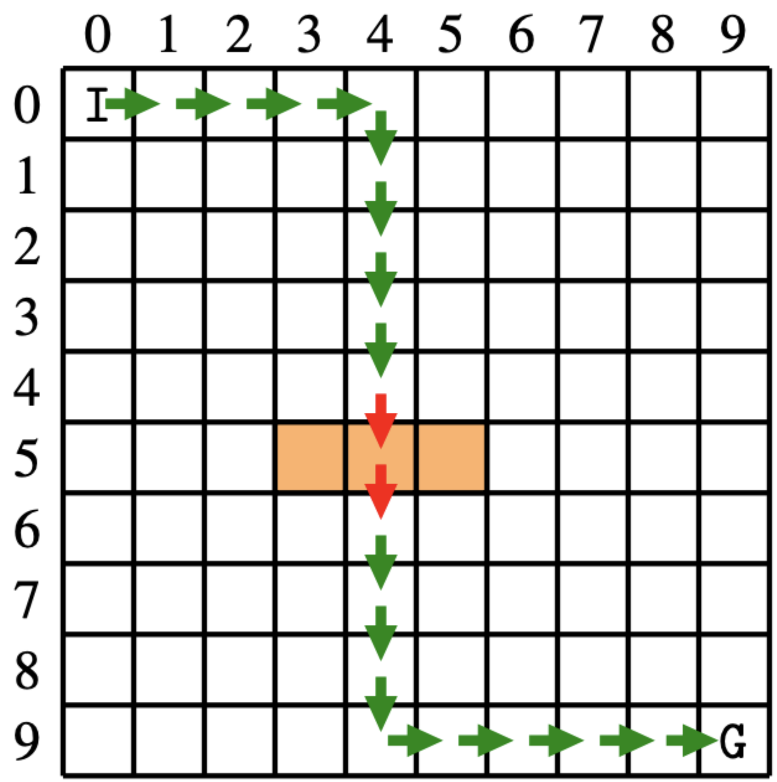
Assignment 2

- The algorithm to repair a broken route segment

- Visit order: Above, below, left, right

Queue

([4, 4], 0)



- Visit order: Above, below, left, right

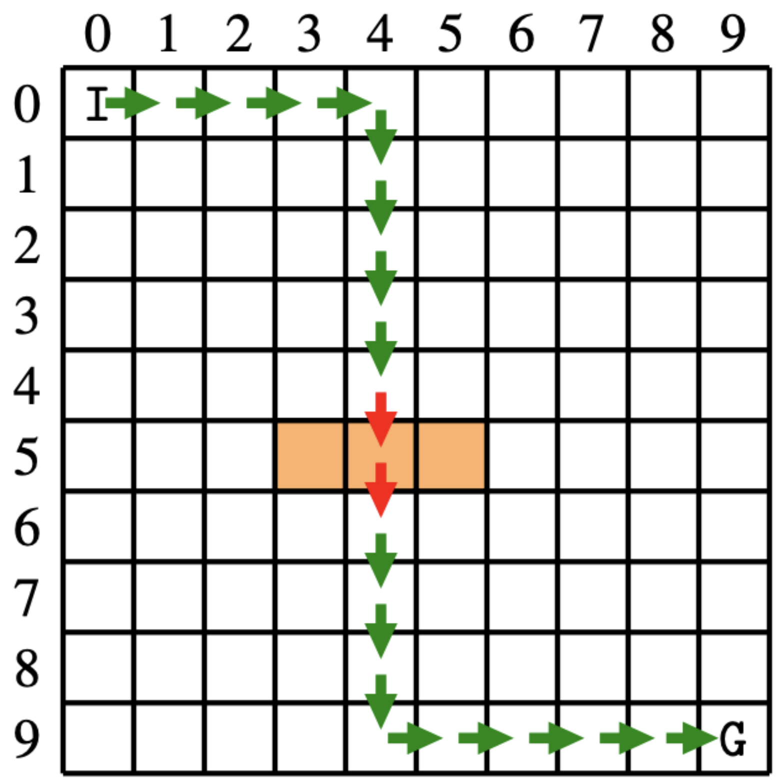
Queue

([4, 4], 0)

([3, 4], 1)

([4, 3], 1)

([4, 5], 1)



- **Visit order: Above, below, left, right**

Queue

([4, 4], 0)

([3, 4], 1)

([4, 3], 1)

([4, 5], 1)

([2, 4], 2)

([3, 3], 2)

([3, 5], 2)

([4, 2], 2)

([4, 6], 2)

	0	1	2	3	4	5	6	7	8	9
0	I		6	5	4	5	6			
1		6	5	4	3	4	5	6		
2	6	5	4	3	2	3	4	5	6	
3	5	4	3	2	1	2	3	4	5	6
4	4	3	2	1	0	1	2	3	4	5
5	5	4	3				3	4	5	
6	6	5	4	5	6	5	4	5		
7		6	5	6			5			
8			6							
9										G

...

Traverse backwards and get the repaired route

	0	1	2	3	4	5	6	7	8	9
0	I		6	5	4	5	6			
1		6	5	4	3	4	5	6		
2	6	5	4	3	2	3	4	5	6	
3	5	4	3	2	1	2	3	4	5	6
4	4	3	2	1	0	1	2	3	4	5
5	5	4	3				3	4	5	
6	6	5	4	5	6	5	4	5		
7		6	5	6			5			
8			6							
9										G

Traverse backwards and get the repaired route