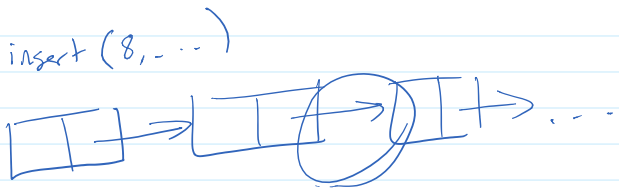
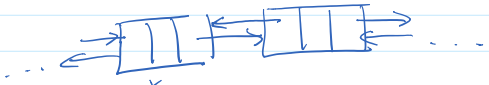
Cycle Detection & Stacks/Queues

Tuesday, September 22, 2015 11:39 AM



Null Pointer Exceptions



1) End of list

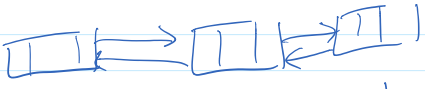
next.prev = thisNode

null.prev

Beginning of list

2)

3)

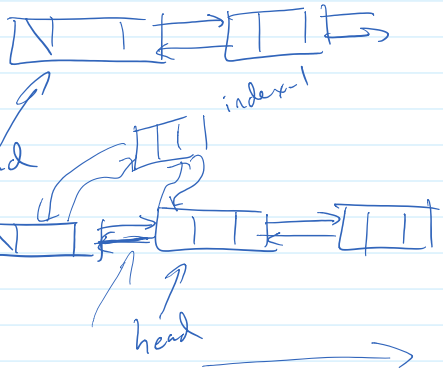


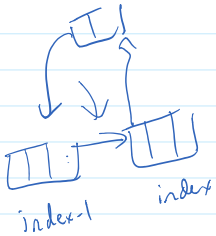
insert problem on insert

4) if (head == null) {...}

for (...)

insert at 0





newNode.p = node_index.prev

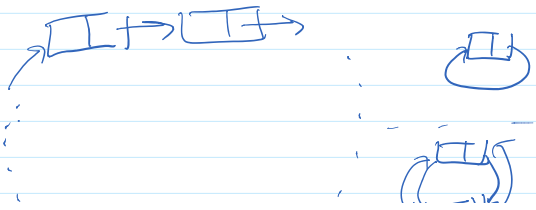
node_index.prev = newNode

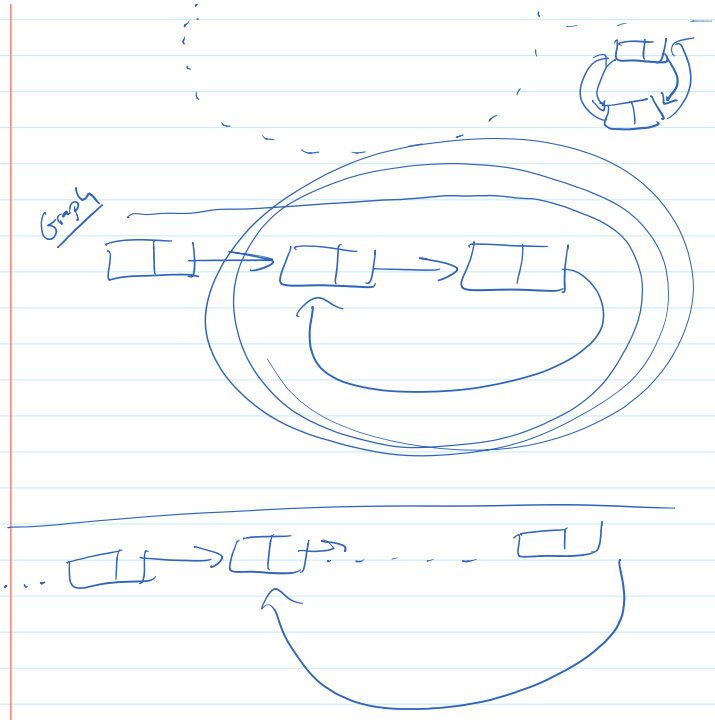
Node(E) .iter

Cycle Detection

Singly

Singly (circular)

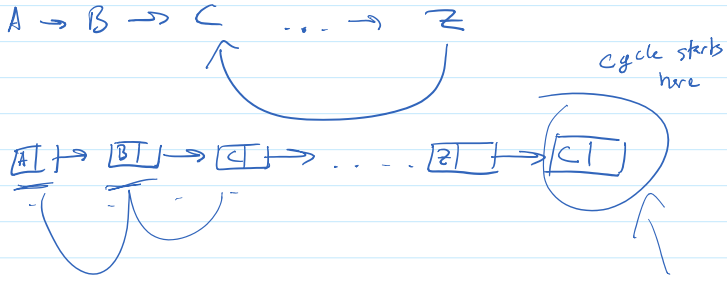




How would you detect a cycle?

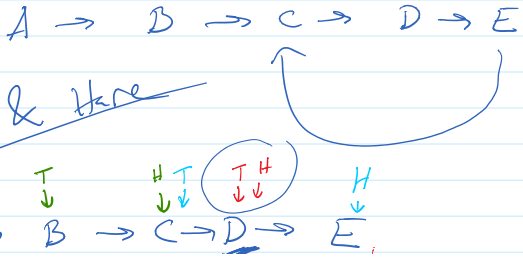
- Try to see if an element is "doubled" along the path (Floyd)
  Copy this into a list as you traverse it, and check for duplicate elements

n linked elements w/ or w/o cycle
space: n (worst case)
time: n^2 (check if element is in the list, then add it to list)



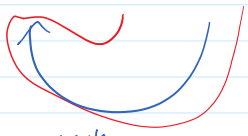
n^2 steps
for element in potential cycle: n steps
  for element2 in list: n steps
    Does element = element2

Aesop's Fables?



Tortoise & Hare

1st T H
A → B → C → D → E



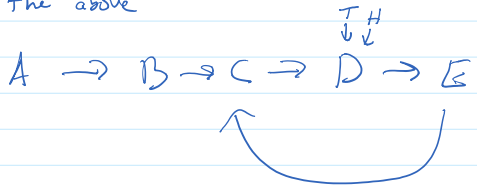
If Hare reaches null, no cycle

If Hare reaches tortoise, cycle

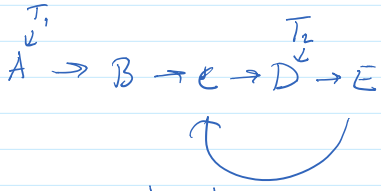
~ linear time

Check for the start cycle:

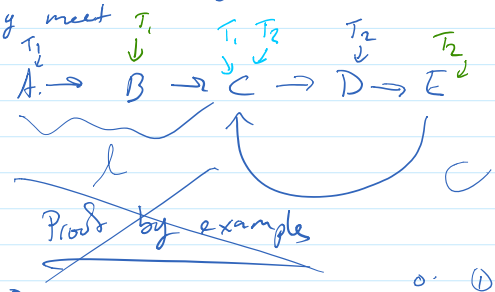
1) Do the above



2) Create another tortoise at the start



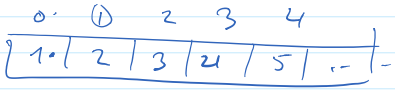
3) Increment both by one each step, until they meet



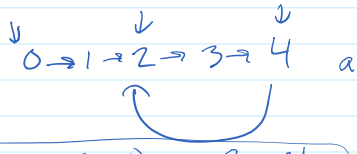
~~Proof by example~~

```
hare = 0
tort = 0
while (true) {
    tort = arr[tort];
    hare = arr[hare];
    hare = arr[hare];
    if (hare left) {
        no cycle;
        break;
    }
    if (hare == tort) {
        cycle;
        find the start;
    }
}
```

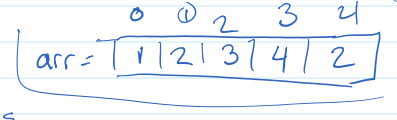
exception



0 → 1 → 2 → ...



as a chain



as an array

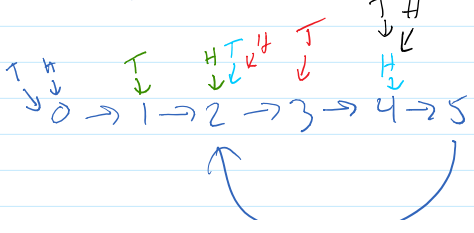
hare = 0 // start

1-step → hare = arr[0] // 1 } 1 iter

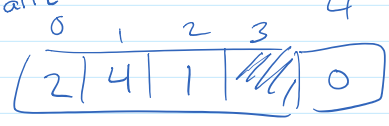
2-step hare = arr[1] // 2

1-step hare = arr[2] // 3 } 2 iter

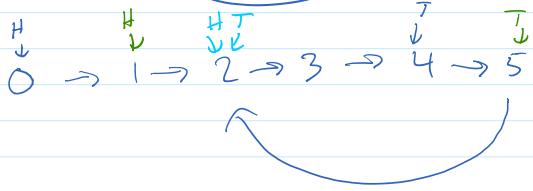
hare = arr[3] // 4

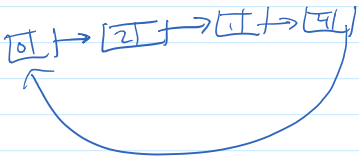


arr[arr[...]]



hare.next.next





Speed: n $O(n) \approx cn$

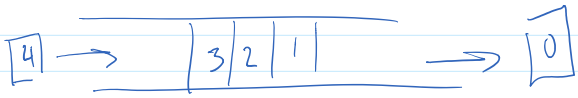
Space: constant (or 1) $O(1) \approx c1$

Stacks & Queues

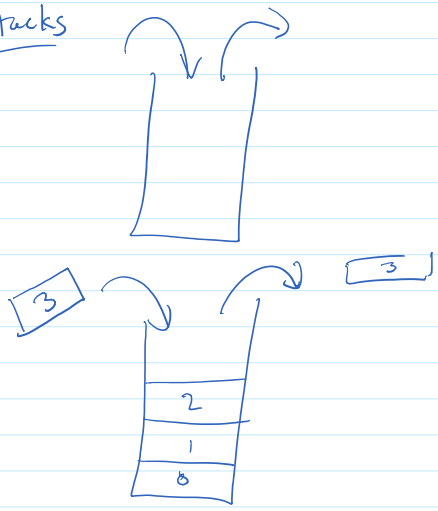
FIFO – First in first out Queue

FILO – First in last out Stack

Queue



Stacks



Queue

size()

enqueue (add)

dequeue (remove)

class Queue<E> {

```
LinkedList<E> theList;

int size() {
    return theList.size();
}

void enqueue(E elem) {
    theList.add(0, elem);
}

E dequeue() {
    return theList.remove(size()-1);
}

E peek() {
    return theList.get(size()-1);
}
}
```