FF LINKED LISTS

SEPT 8, 2023

ANNOUNCEMENTS

CHANGES TO INFO SHEET:

- EC POINTS

THURSDAYS ACCOMMODATIONS

- 09/14 - FF CLASSES & RECURSION
- 09/15 – FF2
- For people who cannot make it…

HOW TO PREPARE:

In class programming Slides

How FF Will Work (most of the time)

Treat it like an technical interview

1. Create Video
2. Explain DS
3. Explain Concept/Problem Solving technique
4. Live Solving & implementation
5. Success! +2 EC points

How FF is Graded

- In depth explanation about related DS concepts
- Live Implementation & Solving (or get close to solving) question

Explain in your thought process in your own words using your own resources (Drawings, slides, diagrams).

Plagiarism bad

MAX VIDEO TIME = 30 mins (large file sizes might cause submission errors)

NO MIN VIDEO TIME

Videos be finished grading by next friday.

HOW TO SUBMIT

Upload video to coogTube or any other video sharing platform (youtube)

COOGTUBE VIDEO UPLOAD LINK

Upload code to repl.it or any other code sharing platform

Provide links to code and video in spreadsheet:

SPREADSHEET LINK

SUBMIT **BEFORE** 9PM

Contact me if there are any upload/submission errors

INSTRUCTIONS P1

Explain Linked Lists

- How they interact with memory
- Benefits/Disadvantages
- How to set up the linked list class and node struct ***
- Head pointer? ***
- private / public?
- Singly/Doubly/Circular? ***
- Common LL functions *** (only explain unless required to implement for question)

Instructions P2 (LIVE IMPLEMENTATION PORTION)

Question:

A car factory quickly produces multiple different cars on one conveyor belt. One out of every n cars on the belt there will be one defected. You are given the index of the defected car x places from the end of the line.

Design a program can:

1. add cars to the end(tail) of the conveyor belt
2. Print the model of the first car in line and its price
3.
    a. remove the defected car from the conveyor belt.
    b. Print the model of the removed car.
    c. The program must remove defected cars at an O(N) time complexity to maintain the factory's quick production. (One traversal!)

- You may **not** use a global list_size variable.
- The Line of cars on the conveyor belt will be represented as a Linked List
- The cars themselves will be represented as nodes
- Don't Forget to account for Edge Cases!

Instructions P2 - EX

```
struct car{
    // ???
};
```

head

```
class factoryBelt{
    private:
    // ???
    public:
    void removeDefected(int);
    // ???
};
```

Instructions P2 - EX

start

head

```
void factoryBelt::removeDefected(int x){
///CODE GOES HERE
/// x == 2 in this scenario
}
```

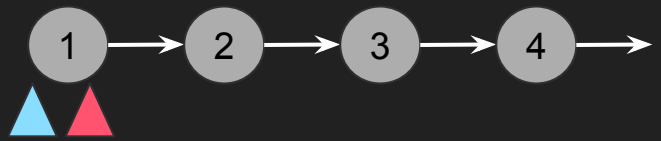
End

head

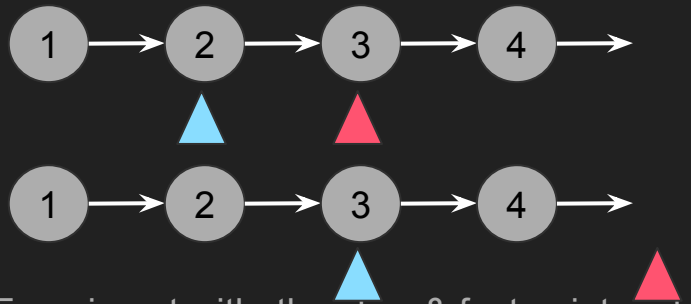
General Concept of Common Linked List Solving Techniques

Fast & Slow Pointers / Turtle & Hare Method

Given a LL, initialize Slow and Fast pointer



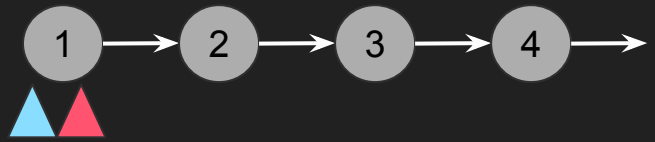
Increment Slow by 1 and Fast by 2



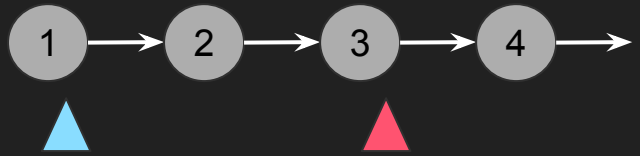
Experiment with the slow & fast pointers to complete your algorithm

Sliding Window

Given a LL, initialize a L and R pointer



Increment R as x times as needed for your algorithm



You now have a sub-array or “window” (1→2→3) that you can use to complete your algorithm.