Week 1 Lecture Notes: Intro & Linked Lists

Slides: <https://drive.google.com/file/d/1hM_1wN94kPi25khc8THP4CkxGE52IlGt/view>

**Interviews**

* Algorithms are the core of interviews – we’ll learn about algorithms that are common in interviews in this course
* Judged on **coding score, speed, debugging** (can you figure out why a solution doesn’t work?)
* Will also discuss resume, non-technical interviews, etc. in this course

**Weekly Schedule**

* Before first lecture: read guides, watch walkthroughs, and do warm up problems
* Attend lectures
* Complete weekly timed HackerRank test
* Optional: extra reading & practice problems

**Tips for success**

* Reach out for help as needed
* Build a habit [of practicing regularly] – try doing one or two problems a day
  + Don’t be too hard on yourself; keep going even if it’s challenging
* Don’t spend too much time on one problem (think 25-30 minutes max)
  + Afterwards, review the solution and then try again with the solution in mind
* Attend all practice sessions
* Use the language you’re most comfortable with

**“Tell me about yourself” – talking points**

* Talk about:
  + Interesting focuses/projects from past companies
  + Passion projects
    - Talking a bit about projects you’re excited about is a good way to garner interest
  + Why you’re interested in the company
  + Specialties you’re interested in (mobile, machine learning, …)
* Common mistakes:
  + Only talking about things that are on your resume
  + Not mentioning why you’re interested in the role/company
  + Not showing enthusiasm
    - \*interviewers are interviewing you as a potential coworker; they’re evaluating if they want to work with you
    - Showing excitement keeps the interviewer engaged
  + Not knowing about the company/role you’re interviewing for
* Spiel should be ~3 minutes total

**Most common interview mistakes**

1. Jumping to conclusions/solving the wrong problem – avoid doing this and wasting time
2. Not communicating thought process – you should be telling the interviewer what you’re doing as you’re doing it, don’t stay silent
3. Not engaging with interviewer – interview is a collaborative process, your interviewer can help guide you to the right place and point out mistakes
4. Missing crucial edge cases – people often focus so heavily on solution that they forget to enumerate edge cases
5. Not discussing space/runtime tradeoffs – you should analyze the solution you come up with against other possible solutions

**UMPIRE method** (to address above mistakes) – can be used for coding, whiteboard interviews, etc.

* **Understand** what the interviewer is asking with clarifying questions and test cases
  + *Take a few minutes to make sure you and interviewer are on same page*
  + *Come up with your own test cases and validate with interviewer that output from your test case is what they expect*
  + *May be able to come up with solutions by walking through cases and possible solutions*
  + State any assumptions you make
    - Is the input always sorted?
    - Is the input guaranteed to satisfy [x & y] conditions?
  + Given [x] input, do we expect [y] output?
* **Match** – does this problem match any common patterns we’ve seen?
  + Which data structures/techniques can we use to simplify this problem?
  + e.g. given linked list problem, would employing dummy head, two pointer, or multi-pass techniques solve the problem?
* **Plan**
  + Use diagrams and pseudocode to visualize how the problem will be solved
    - *It’s quicker to write pseudocode than actual code*
    - *Doing this gives the interviewer a chance to help you correct your course, give you hints, and help you catch bugs*
    - *Once validated, it makes actual coding a lot easier*
  + It’s easier to modify your solution before you write all the code
  + Catch potential bugs before starting to write code
  + Run through your approach with test cases to check that it works
* **Implement** – code
  + *Interviewers judge your code cleanliness, so keep it organized*
  + *Aim for readability over conciseness – make it easy to read through; it will help your interviewer understand your code* 
    - *Also makes it easier for you to debug*
    - *To improve this, review weekly solutions and compare 🡪 see if you can apply what you notice in solutions to your own code*
* **Review** the code you’ve written
  + *Run through all the test cases to make sure they’re all caught*
  + Trace through each line of your code with an input to check for the expected output
  + Catch possible edge cases, off-by-one errors, missed steps
  + Run your code and debug your code
    - *Polish it and show that you know how to debug your code*
* **Evaluate** 
  + Analyze the runtime and space complexity of your solution
  + Discuss tradeoffs that were made, or assumptions that were taken
  + *Can also discuss what else you’d want to test if you had more time and what you’d want to improve if you had more time*

**Bonus tips:**

* medium/hard LeetCode questions are most common, you may need to implement follow-up questions from the interviewer if there’s time left over
* If you get stuck, talk to your interviewer – they can help you get unstuck and work with you; they want to see you succeed
* To make sure you’re on the same page as the interviewer, make sure the outputs of your test cases are what your interviewer expects
* Answering the “tell me about yourself”– this question should just be a brief intro of yourself so they can ask follow-up questions after

**Linked Lists**

* The linked list patterns are generally pretty simple
  + *Read up and practice dummy head, multi-pass, and two pointer patterns outlined in course portal*
* The most challenging aspects of linked list questions are:
  + Making sure you update all the pointers properly
  + Keeping track of all the pointers
  + Writing clean code to deal with all the pointers
* To get better at linked lists, go through extra practice problems in assignment tab 🡪 helps you improve pointer bookkeeping and debugging pointer issues

**Before the next session**

* Read warm up guides
* Review warm up problems
* Read and watch UMPIRE guides
  + <https://guides.codepath.org/compsci/UMPIRE-Interview-Strategy>
  + <https://www.youtube.com/watch?v=W6V7MLE_5X4&feature=youtu.be>
* Practice with post-session practice problems
  + [Copy List with Random Pointer](https://leetcode.com/problems/copy-list-with-random-pointer/)
  + [Linked List Cycle II](https://leetcode.com/problems/linked-list-cycle-ii/)

**Next session:** walking through a linked list problem with UMPIRE approach, group exercises using UMPIRE