Karat^ Interview Prep for Interviewer Candidates

- The interview is totally remote. You'll use our scheduler to pick a time that works for you. You'll get more information after that. I recommend using headphones, and testing your audio/visual before the interview starts -- and of course we'll help you through any problems that arise!

- Be prepared to introduce yourself (1-2 minutes) and to talk about a project you've worked on and the technical challenges you faced. This will be a discussion with your interviewer; they'll ask questions to clarify what you did and how things worked. I recommend choosing industry experience if possible, but personal projects are fine, too. Don't stress this part -- we really just want to have a technical conversation about something you're familiar with!

- The coding section is 45 minutes long. You'll get coding questions in which your goal is to write a function that takes the sample input and returns the expected output, which can then get printed to verify it's correct. Things like code quality and runtime performance will be noted, but your primary goal is to get a working solution, without bugs on other inputs and edge cases, so that you can move on to demonstrating your skills on more problems.

- I recommend practicing using https://coderpad.io/ since that is what we use during the interview. You can use whatever language you'd like for the interview. Feel free to run your code as much as you like during the interview. Use the REPL (if applicable to the language selection). Develop however is most comfortable to you. If using helper methods is helpful, do that. If using simplified input and print statements is helpful, do that. Watch out for spending too much time on the most perfect object oriented design or most optimal solution, or getting lost in cascading bug confusion, as this time takes away from demonstrating your skills on other problems. Candidate who do well are able to make steady progress towards fully working code, usually by identifying an approach up front that they are then able to implement.

- We're looking for computer science fundamentals. You'll want to be comfortable using arrays, nested arrays and maps (hashmap in java, dict in python, object in js, etc). The input may be in a string format that requires some manipulation. You may need to take an input that is a simplified abstraction of graph edges, and process it into some format that your code can then operate over. If you have any questions about this let me know. None of our questions are tricks: you don't have to worry about invalid input or knowing how to balance a red black tree. If you're comfortable writing real world code, and have a done a few HackerRank-style questions to get back in the swing of basic algorithms and data structures (loops, BFS/DFS), you'll do great!

- You'll also want to be comfortable analyzing the runtime and space complexities of your code. While the optimality of your approach will certainly be noted, it's more important that you first get a working solution and second are able to analyze the code you wrote.