*Mindbank has created a programming exercise to assist in determining what level of experience a candidate has in relation to the requirements of the custom IPaC application . We find this is a better experience than live whiteboard coding, since it more accurately reflects the life of a developer where one is required to research a problem and experiment with solutions.*

*I've included the exercise below, followed by some questions that would be discussed in an interview. We'd like you to submit the programming exercise when completed through GitHub.*

***PROGRAMMING EXERCISE***

[*Directed acyclic graphs*](https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FDirected_acyclic_graph&data=05%7C01%7CTodd.Gindlesperger%40mindbank.com%7Cd274693a7f024034b6ee08dbc3666451%7Cb8af36bd8f414a1c8236f1d4c210c39c%7C0%7C0%7C638318616793357989%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=iu%2FwUNwVBjB13hjZAcaZIjCpdxQ2slEodKYIAoc4cT8%3D&reserved=0) *(DAGs) are a class of graph in computer science. Directed meaning the structure goes one way (there are from and to vertices, conceptually); acyclic meaning there are no loops (if A leads to B, B cannot lead to A).*

*DAGs are a collection of vertices and edges, which could be represented as simple objects (these classes are just to illustrate, you can use them or define your own):*

*class Vertex {*

*long id*

*}*

*class Edge {*

*Vertex from*

*Vertex to*

*}*

*Given a DAG and a vertex, calculate the longest directed path from that vertex. To give the exercise some reference, this was an actual problem encountered in the IPaC program  - showing progress through a variable-length, graph-based questionnaire.*

***QUESTIONS***

1. ***Does the solution work for larger graphs?***

Yes, within the scope of long type in Java, the complexity of the algorithm is O(Vertices + Edges), the topological sort takes O(Vertices + Edges) and the processing for the longest path calculation also takes: O(Vertices + Edges). This linear complexity makes the algorithm suitable for larger graphs.

The recursive Deep First Searching approach in the topological sort could lead to a stack overflow for a large number of vertices if the graph has a very deep recursion depth and exceed the computer memory capacity.

1. ***Can you think of any optimizations?***

Yes, we can segment the graph, for example, there are vertices: A, B, C, D, E, F, G. and A, B, C and D are connected and E, F and G are connected. So we are segment the graph into two sub-graphs, if we want to know the longest path for A, there is no need to consider the vertices: E, F and G. If we want to know the longest path for E, there is no need to consider the vertices: A, B, C and D.

1. ***What’s the computational complexity of your solution?***

The topological sort takes O(Vertices + Edges) and the processing for the longest path calculation also takes: O(Vertices + Edges), so the overall computational time complexity is O(Vertices + Edges). The space complexity is O(Vertices), primarily to store the topological ordering and distance map.

1. ***Are there any unusual cases that aren't handled?***

Yes, the algorithm assumes that the input graph is a DAG, if the graph contains cycles, the topological sort will fail.

For very large graphs, this could lead to a stack overflow error.

*Feel free to provide a README.md with comments on the implementation.*

*Additional questions to think about:*

* ***What are some things you don’t like about Java?***

I don’t like Java’s performance overhead, because Java is operates on Java Virtual Machine, which can lead to performance overhead compared to C and C++. And the memory management, the Java garbage collector can be unpredictable and may cause performance issues.

* ***If you could choose any language/framework/technology stack, what would you choose and why?***

I choose the language according to their capability, libraries and frameworks. For example, for data science and AI, I will choose Python, because there are TensorFlow, PyTorch, Pandas, NumPy libraries to support the subject domain. If I want to develop enterprise application, I might choose Java, because Java is good at enterprise application. I like Java and TypeScript, I found the syntax of Java and TypeScript to look at and they are formatted pretty and neat.

I can use any languages without any efforts. I am multi-lingual. Those languages shares similar concepts and just little differences in syntax and language features. Whatever language is suitable for the subject domain or project, I will choose the language to use.

*If you have any questions about the exercise, please don’t hesitate to ask and we will have an IPaC team member reach out to you.*

*Thank you again for your interest in this opportunity. We look forward to seeing what you create from your research!*