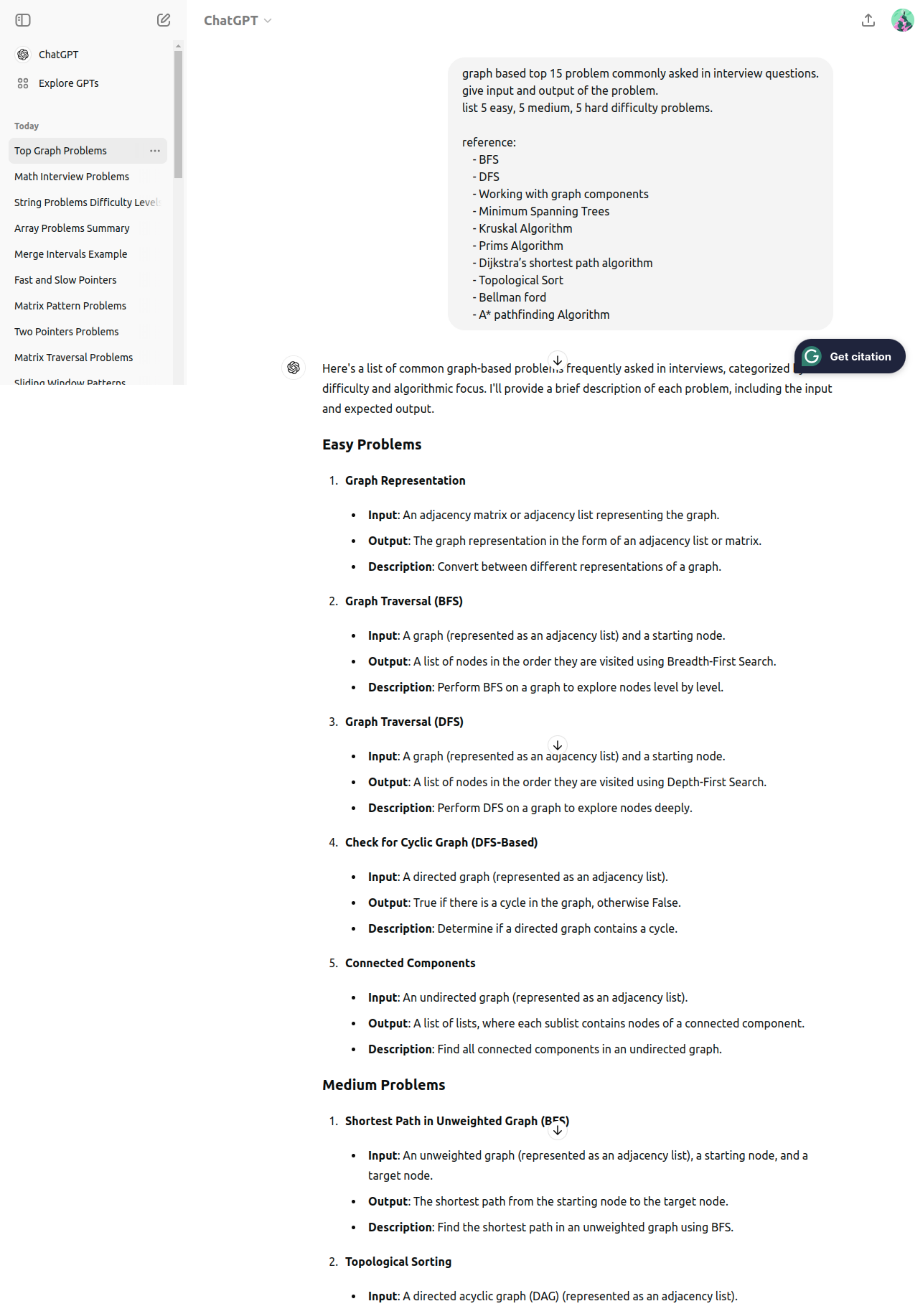

graph based top 15 problem commonly asked in interview questions.
give input and output of the problem.
list 5 easy, 5 medium, 5 hard difficulty problems.

reference:
- BFS
- DFS
- Working with graph components
- Minimum Spanning Trees
- Kruskal Algorithm
- Prims Algorithm
- Dijkstra's shortest path algorithm
- Topological Sort
- Bellman ford
- A* pathfinding Algorithm

Here's a list of common graph-based problems frequently asked in interviews, categorized difficulty and algorithmic focus. I'll provide a brief description of each problem, including the input and expected output.

Easy Problems

1. **Graph Representation**
   - **Input**: An adjacency matrix or adjacency list representing the graph.
   - **Output**: The graph representation in the form of an adjacency list or matrix.
   - **Description**: Convert between different representations of a graph.
2. **Graph Traversal (BFS)**
   - **Input**: A graph (represented as an adjacency list) and a starting node.
   - **Output**: A list of nodes in the order they are visited using Breadth-First Search.
   - **Description**: Perform BFS on a graph to explore nodes level by level.
3. **Graph Traversal (DFS)**
   - **Input**: A graph (represented as an adjacency list) and a starting node.
   - **Output**: A list of nodes in the order they are visited using Depth-First Search.
   - **Description**: Perform DFS on a graph to explore nodes deeply.
4. **Check for Cyclic Graph (DFS-Based)**
   - **Input**: A directed graph (represented as an adjacency list).
   - **Output**: True if there is a cycle in the graph, otherwise False.
   - **Description**: Determine if a directed graph contains a cycle.
5. **Connected Components**
   - **Input**: An undirected graph (represented as an adjacency list).
   - **Output**: A list of lists, where each sublist contains nodes of a connected component.
   - **Description**: Find all connected components in an undirected graph.

Medium Problems

1. **Shortest Path in Unweighted Graph (BFS)**
   - **Input**: An unweighted graph (represented as an adjacency list), a starting node, and a target node.
   - **Output**: The shortest path from the starting node to the target node.
   - **Description**: Find the shortest path in an unweighted graph using BFS.
2. **Topological Sorting**
   - **Input**: A directed acyclic graph (DAG) (represented as an adjacency list).

- **Output**: A list of nodes in topologically sorted order.
- **Description**: Perform topological sort on a DAG.

3. **Minimum Spanning Tree (Prim's Algorithm)**
    - **Input**: A weighted, undirected graph (represented as an adjacency list).
    - **Output**: The edges included in the minimum spanning tree and its total weight.
    - **Description**: Find the Minimum Spanning Tree of a graph using Prim's Algorithm.
4. **Dijkstra's Shortest Path Algorithm**
    - **Input**: A weighted graph (represented as an adjacency list), a starting node.
    - **Output**: Shortest path distances from the starting node to all other nodes.
    - **Description**: Find the shortest path from a single source to all other nodes in a weighted graph using Dijkstra's Algorithm.
5. **Bellman-Ford Algorithm**
    - **Input**: A weighted graph (represented as an adjacency list) with possible negative weights, and a starting node.
    - **Output**: Shortest path distances from the starting node to all other nodes and detect any negative weight cycles.
    - **Description**: Compute shortest paths from a single source in a graph with possible negative weights using Bellman-Ford Algorithm.

Hard Problems

1. **Kruskal's Minimum Spanning Tree Algorithm**
    - **Input**: A weighted, undirected graph (represented as a list of edges).
    - **Output**: The edges included in the minimum spanning tree and its total weight.
    - **Description**: Find the Minimum Spanning Tree of a graph using Kruskal's Algorithm.
2. _A Pathfinding Algorithm_*
    - **Input**: A grid-based graph with costs, a start node, and a target node.
    - **Output**: The shortest path from the start node to the target node considering heuristic information.
    - **Description**: Find the shortest path in a weighted grid-based graph using the A* algorithm.
3. **Network Flow (Ford-Fulkerson Method)**
    - **Input**: A directed graph with capacities on edges, a source node, and a sink node.
    - **Output**: The maximum flow from the source to the sink.
    - **Description**: Compute the maximum flow in a network using the Ford-Fulkerson method.
4. **Strongly Connected Components (Kosaraju’s Algorithm)**
    - **Input**: A directed graph (represented as an adjacency list).
    - **Output**: A list of strongly connected components.
    - **Description**: Find all strongly connected components in a directed graph using Kosaraju’s Algorithm.
5. **Traveling Salesman Problem (TSP) - Approximation**
    - **Input**: A complete weighted graph (represented as an adjacency matrix).
    - **Output**: An approximate solution to the Traveling Salesman Problem (TSP).
    - **Description**: Find an approximate solution to the TSP, which is a classic optimization problem.

Feel free to ask for more details or specific implementations for any of these problems!

Is this conversation helpful so far?

Sliding Window Patterns

Upgrade plan
Get GPT-4, DALL·E, and more

Message ChatGPT

ChatGPT can make mistakes. Check important info.

?