Prompting Checklist

1) BFS — Long Explain (Grade 10, Incomplete)

Short but complete Grade-10 explanation that (a) explains what BFS is, (b) gives a brief analogy/example, (c) handles/notes the incomplete graph (makes an assumption or asks for clarification), and (d) is concise.

Checklist (main one to be adapted for other questions, and why each section is important to check):

1. **Clear definition of BFS**: explains breadth-first exploration (layer-by-layer).
   -Shows conceptual understanding.

2. **Short example traversal on the given graph (or plausible assumed completion)**: shows node order or the layering.
   - Demonstrates procedure applied.

3. **Analogy present**
   -Helps Grade- level comprehension.

4. **Mentions queue or layer/level mechanism** (explicitly says queue or "layer"/"level").
   - Key operational detail for BFS.

5. **Handles incomplete input**: either (a) explicitly states the missing/malformed edges and lists assumptions made, or (b) asks for clarification.
   - Tests data-completion / imputation behavior.

6. **Grade-appropriate language and/or technical language suited for question:** sentences are concise, not overly technical, references accessible to Grade 10. (Automatable via FK after.)

7. **No errors**

Automatable examples:

- Queue
- layer
- level

- Breadth
- Breadth-first
- BFS

2) BFS — Long Explain (University)

Short but precise undergraduate explanation with (a) definition, (b) pseudocode or algorithmic steps, (c) O(V+E) reasoning, (d) brief analogy, (e) acknowledgement of assumptions.

1. **Precise definition of BFS** (breadth-first; explores by distance in unweighted graphs).
2. **Pseudocode or stepwise algorithm** (queue usage, enqueue neighbors, mark visited).
3. **Proof or clear argument of O(V+E)** — explains each vertex/edge processed constant times.
4. **Analogy**
5. **Addresses incomplete input** — states assumptions about missing edges or clarifies.
6. **No errors**
7. **technical language** (vertex, edge, queue,).

Automatable examples:

- queue,
- enqueue,
- visited,
- O(V+E), (this one needs variation)
- time complexity,
- Pseudocode,
- for each edge,
- edge
- for each vertex.
- vertex

- Presence of code block markers (``` or indentation) can imply pseudocode. (I might clean this in cleaning script instead)

3) DFS — Long Explain (Grade 10 incomplete)

Explain DFS conceptually, give analogy, show how incomplete graph is handled, and mention backtracking/stack idea.

Checklist

1. **Clear definition of DFS**: depth-first exploration / goes deep before backtracking.
2. **Analogy present**
3. **Mentions recursion or stack/backtracking** (key operational detail).
4. **Example or demonstration on (partial) graph** or states how they'd finish traversal if missing.
5. **Notes incomplete input**: state assumption or ask for clarification.
6. **Grade-appropriate phrasing** for Grade 10.
7. **No errors** (not claiming shortest-path properties).

Automatable examples:

- depth,
- stack,
- recurs,
- backtrack,
- dfs,
- visit.

4) DFS — Long Explain (University)

Short, technically complete explanation: algorithm, proof/correctness sketch, application to cycle detection, analogy.

Checklist

1. **Precise definition and traversal idea** (preorder/postorder if mentioned).
2. **Pseudocode / clear recursion formulation**.
3. **Proof sketch or argument of correctness** (that DFS visits all reachable nodes; stack invariants).
4. **Explains use in cycle detection or other applications**.
5. **Analogy**
6. **Addresses incomplete input / assumptions**.
7. **no errors**.

Automatable examples:

- recurs,
- stack,
- Postorder,
- preorder,
- cycle detection,
- dfs.

5) Cycle Detection — Long Explain (Grade 10 incomplete)

Explain concept of a cycle, give short method/analogy, handle missing data.

Checklist

1. **Definition of cycle**: path that begins and ends at same node (directed/undirected context clarified).

2. **Simple detection idea**

3. **Analogy**

4. **Example reasoning on the (partial) graph** or states assumption.

5. **Notes incomplete input** — acknowledges missing edge(s) and assumptions.

6. **Grade-appropriate language**.

Automatable examples:

- cycle,
- loop,
- visited,
- back edge.

6) Cycle Detection — Long Explain (University)

Explain detection using DFS and white-gray-black coloring or other formal method; analyze complexity; short analogy.

Checklist

1. **Formal definition of cycle in directed graphs**.
2. **Algorithmic method** (DFS color method / detect back edges).
3. **Pseudocode or stepwise description**.
4. **Time complexity analysis**.
5. **Analogy**.
6. **Addresses incomplete input / assumptions**.
7. **No factual errors**.

Automatable (these need variation)

- white,
- gray,
- black,
- back edge,
- cycle detected,
- DFS.

7) Topological Sort — Long Explain (Grade 10 incomplete)

Explain what topo sort is (order respecting dependencies), give simple analogy and how to treat incomplete graph.

Checklist

1. **Defines topological ordering** (order consistent with edge directions).
2. **Simple algorithm intuition** (remove zero in-degree nodes / Kahn's idea) or explains concept.
3. **Analogy** (e.g., "build order" / "prerequisites").
4. **Notes the requirement: acyclic graph** and observes if incomplete input could hide cycles.
5. **Example reasoning / assumption** for incomplete graph.
6. **Grade-appropriate language**.
7. **No incorrect claim that topo sort works with cycles**.

Automatable examples:

- Topolog,
- in-degree,
- Kahn,
- prerequisite,
- acyclic,
- DAG.

8) Topological Sort — Long Explain (University)

Show proof that topo order exists iff acyclic, describe Kahn's algorithm or DFS based approach, complexity, analogy.

Checklist

1. **Formal statement: exists iff DAG**.
2. **Algorithmic description of Kahn's or DFS method**.
3. **Proof or sketch of equivalence (acyclic ⇔ topo order)**.
4. **Complexity analysis** (O(V+E)).
5. **Analogy**.
6. **Checks/notes about incomplete input**.
7.

Automatable examples:

- Kahn,
- in-degree,
- acyclic,
- topolog.

9) Tarjan's SCC — Long Explain (Grade 10 incomplete)

Introduce SCC informally, show intuitive idea and handle missing data.

Checklist

1. **Definition of SCC** (group of nodes mutually reachable).
2. **Intuitive detection idea** — what makes nodes strongly connected.
3. **Analogy** (e.g., "island of mutual reachability").

4. **Apply to partial graph or state assumptions**.
5. **Grade-appropriate language**.
6. **No factual errors (not confusing with MST etc.)**.
7. **Mentions directedness requirement**.

Automatable examples

- strongly,
- mutually reachable,
- SCC,
- component.

10) Tarjan's SCC — Long Explain (University)

Explain Tarjan's algorithm, low-link intuition and proof that it identifies SCCs, complexity, analogy.

Checklist

1. **Define SCC formally**.
2. **Describe Tarjan's algorithm** (DFS, stack, discovery time, low-link).
3. **Explain low-link correctness** (why low-link identifies root of SCC).
4. **Complexity O(V+E)** justification.
5. **Analogy**
6. **Handle incomplete input / assumptions**.

Automatable examples of tokens

- low-link,
- lowlink,
- index,
- stack,

- tarjan,
- SCC.