Prompt Engineering / Reasoning Systems

Connect related reasoning threads, identify shared subproblems, and merge redundant paths.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Complex Multi-Step Problems, Research Synthesis, Planning
Updated: May 2026
Why This Prompt Exists
Tree-of-thoughts explores multiple branches but treats each branch independently — missing opportunities to share work between branches.

You get:

- duplicate reasoning across branches (wasted compute)
- no way to combine insights from different branches
- branching that explodes exponentially without merging
- missed subproblems that appear in multiple branches
- inefficient exploration of solution space

But graph-of-thoughts solves this:

- nodes: reasoning states or subproblems
- edges: transitions or dependencies between states
- merging: when different paths reach same state, merge them
- shared subproblems: identify and solve once
- graph traversal: explore efficiently

Without merging, tree-of-thoughts explodes exponentially.

This prompt implements graph-of-thoughts for efficient reasoning.

The Prompt

```
Assume the role of a graph-of-thoughts reasoning engine that merges
shared subproblems.

Your task is to explore a reasoning graph where nodes are states and
edges are transitions.

Generate:

1. INITIAL STATE
   - Starting point (problem statement)

2. GRAPH EXPLORATION (breadth-first)

Level 1:
- From initial state, possible next states: [A, B, C]

Level 2:
- From A: possible next states: [A1, A2]
- From B: possible next states: [B1, B2]
- From C: possible next states: [C1, C2]
- Check for merges: [if A1 = B1, merge into single node]

Level N:
- Continue until goal state reached or max depth
```

3. MERGE DETECTION
   - States that are identical or sufficiently similar
   - Merge rationale (why they're the same)

4. GRAPH STRUCTURE (textual representation)
   - Node: [description]
   - Edges: [node] → [node2], [node2] → [node3]
   - Merged nodes: [node X] merges [path1] and [path2]

5. SOLUTION PATH
   - Shortest or best path from start to goal

6. EFFICIENCY GAIN
   - Number of nodes without merging: [X]
   - Number of nodes with merging: [Y]
   - Reduction: [Z]%

INPUTS:

Problem to solve:
[PASTE THE PROBLEM]

Initial state description:
[E.G., "We have $10,000 budget and need to increase retention by 20%"]

Goal state description:
[E.G., "Achieved 20% retention increase"]

State equivalence criteria:

```
[EXACT / SEMANTIC / FUNCTIONAL] (how similar must states be to merge?)

Max nodes (to prevent explosion):
[E.G., "50 nodes"]

Model:
[GPT-4 / CLAUDE / GEMINI]

RULES:
- Merge states that are semantically equivalent (same subproblem solved)
- Track visited states to avoid cycles (don't revisit same state)
- Use breadth-first exploration to find shortest path first
- If graph grows too large, increase merge aggressiveness (treat similar states as identical)
- Store intermediate results from merged nodes to avoid recomputation
- Visualize graph structure if needed for complex problems
```

How To Use It

- Use for problems where many reasoning paths converge on shared subproblems.
- Semantic merging is powerful but risky — verify that merged states are truly equivalent.
- Track visited states to prevent infinite loops.
- Graph-of-thoughts is more efficient than tree-of-thoughts for problems with overlapping subproblems.
- Visualize the graph for complex problems to see where merging is happening.

Example Input

Problem to solve:

“Find the cheapest way to travel from New York to Los Angeles with stops in Chicago and Denver (order flexible).”

Initial state:
“At New York, haven’t visited Chicago or Denver”

Goal state:
“At Los Angeles, visited both Chicago and Denver”

State equivalence criteria:
“SEMANTIC”

Max nodes:
“30”

Why It Works
Tree-of-thoughts would explore NY→Chicago→Denver→LA and NY→Denver→Chicago→LA as separate branches — duplicating the Chicago→Denver and Denver→Chicago subproblems.

This framework improves outcomes by forcing:

- node merging (different paths to same state converge)
- shared subproblem detection (solve once, reuse)
- cycle prevention (don’t revisit states)
- efficiency tracking (quantify improvement over tree)
- flexible equivalence criteria (exact, semantic, or functional)

Great graph-of-thoughts mapping doesn’t explore more — it explores smarter by sharing work.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also Tree-of-Thoughts Explorer