Prompt Engineering / Reasoning Systems

Generate multiple reasoning branches at each step, evaluate them, and prune poor branches before continuing.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Complex Problem Solving, Strategy Planning, Creative Tasks
Updated: May 2026
Why This Prompt Exists
Chain-of-thought follows one path. If that path is wrong, you never recover. Tree-of-thoughts explores multiple paths and picks the best.

You get:

- getting stuck on the first reasonable-sounding path (even if wrong)
- no way to recover from early bad decisions
- missing better solutions that require different first steps
- linear thinking when branching would help
- no systematic way to evaluate alternatives

But tree-of-thoughts solves this:

- branching: generate multiple possible next steps at each decision point
- evaluation: score each branch (promising vs. dead end)
- pruning: drop low-scoring branches
- expansion: continue from promising branches
- selection: choose best final path

Without branching, you commit too early.

This prompt implements tree-of-thoughts reasoning for complex problems.

The Prompt

```
Assume the role of a tree-of-thoughts reasoning engine that explores
multiple solution paths.

Your task is to solve a problem by exploring, evaluating, and pruning
multiple reasoning branches.

Generate:

1. PROBLEM RESTATEMENT
   - Restate the problem in your own words

2. INITIAL BRANCHING (Level 1)
   - Generate 3-5 different approaches to the problem
   - For each approach, state the first step

3. BRANCH EVALUATION
   - For each branch, score: (1 = dead end, 5 = very promising)
   - Brief rationale for each score

4. PRUNING
   - Discard branches with score ≤ 2
   - Keep branches with score ≥ 3

5. DEEPEN SELECTED BRANCHES (Level 2)
   - For each kept branch, generate next step possibilities (2-3 per
```

branch)
   - Show the growing tree structure

6. CONTINUE UNTIL SOLUTION OR MAX DEPTH
   - Repeat evaluation, pruning, and deepening

7. FINAL SOLUTION
   - The best path through the tree
   - Why this path is□□

8. ALTERNATIVE PATHS (briefly)
   - What promising paths were pruned and why

INPUTS:

Problem to solve:
[PASTE THE PROBLEM]

Problem type:
[LOGIC / PLANNING / CREATIVE / OPTIMIZATION / OTHER]

Branching factor (how many alternatives per step):
[3 / 5 / 7] (higher = more thorough but more expensive)

Max depth (how many steps):
[3 / 5 / 7 / 10]

Pruning threshold (score to keep):
[2/5, 3/5, etc.]

```
Model:
[GPT-4 / CLAUDE / GEMINI]

RULES:
- Branch widely at first (explore options), prune aggressively later (focus on promising)
- Evaluation criteria should match problem type (for logic: correctness; for creative: novelty+feasibility)
- Don't prune too early — some branches look weak but become strong after a few steps
- Track visited states to avoid loops (same reasoning step twice)
- If tree grows too large (exponential), increase pruning aggressiveness
```

How To Use It

- Use for complex problems with multiple viable approaches (strategy, planning, creative tasks).
- Start with moderate branching (3 options per step) to avoid exponential explosion.
- Prune aggressively (keep only top 1-2 branches per level) for large problems.
- Visualize the tree structure to understand trade-offs between paths.
- Don’t use for simple problems — overhead isn’t worth it.

Example Input

Problem to solve:
“How can a small e-commerce company increase customer retention by 20% within 6 months with a $10,000 budget?”

Problem type:
“PLANNING / OPTIMIZATION”

Branching factor:
“3”

Max depth:
“4”

Pruning threshold:
“3/5”

Why It Works
Chain-of-thought commits to one path early — which is fine for simple problems but disastrous for complex ones.

This framework improves outcomes by forcing:

- branching exploration (not just one path)
- explicit evaluation (score each branch)
- pruning (drop dead ends early)
- deepening (explore promising branches further)
- comparison (why the chosen path is best)

Great tree-of-thoughts exploration doesn’t find the first solution — it finds the best solution by exploring many paths.

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 Self-Consistency Ensembler