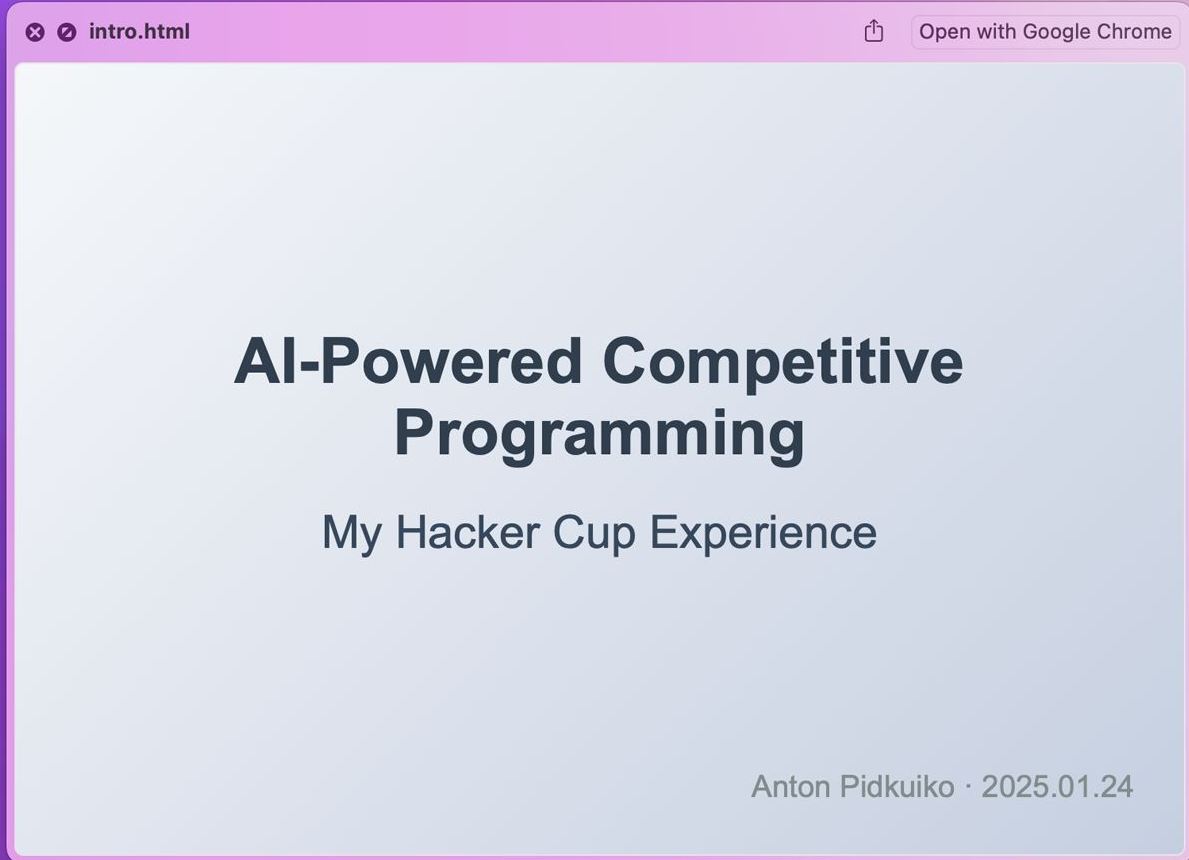




intro.html



HackerCup.zip



< > PROBLEMS

Name

- ∨ A plus B
 - full_in.txt
 - sample_in.txt
 - sample_out.txt
 - statement.txt**
- > Fall in Line
- > Line by Line
- > Line of Delivery (Part 1)
- > Line of Delivery (Part 2)
- > Sort array
- > Walk the Line

statement.txt

A plus B

Simply add two integers A and B.

Constraints

($1 \leq T \leq 100$)

($1 \leq A, B \leq 10^5$)

Input Format

Input begins with an integer T, the number of test cases.

Each case is a single line containing the integers A and B.

Output Format

For the i-th test case, print "`Case #i:` " followed by A + B.



sample_in.txt

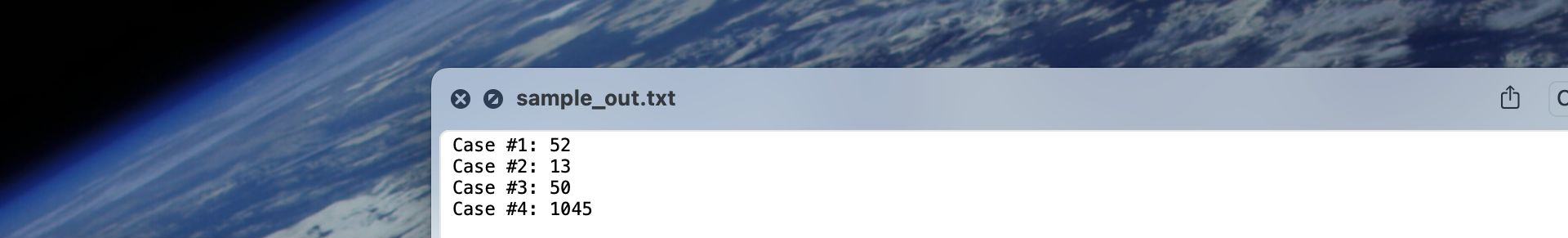


```
4
2 50
3 10
13 37
950 95
```

< > PROBLEMS

Name

- ▼ A plus B
 - full_in.txt
 - sample_in.txt**
 - sample_out.txt
 - statement.txt
- > Fall in Line
- > Line by Line
- > Line of Delivery (Part 1)
- > Line of Delivery (Part 2)
- > Sort array
- > Walk the Line



sample_out.txt



```
Case #1: 52  
Case #2: 13  
Case #3: 50  
Case #4: 1045
```

< > PROBLEMS

Name

- ∨ A plus B
 - full_in.txt
 - sample_in.txt
 - sample_out.txt**
 - statement.txt
- > Fall in Line
- > Line by Line
- > Line of Delivery (Part 1)
- > Line of Delivery (Part 2)
- > Sort array
- > Walk the Line



full_in.txt



< > PROBLEMS

Name

- ▼ A plus B
 - full_in.txt
 - sample_in.txt
 - sample_out.txt
 - statement.txt
- > Fall in Line
- > Line by Line
- > Line of Delivery (Part 1)
- > Line of Delivery (Part 2)
- > Sort array
- > Walk the Line

```
174
2 50
3 10
13 37
950 95
2 1
2 2
2 3
2 4
2 5
2 95
2 96
2 97
2 98
2 99
1000 1
1000 2
1000 3
1000 4
1000 5
1000 95
1000 96
1000 97
1000 98
1000 99
906 56
81 39
500 44
451 5
270 26
755 11
915 30
958 86
642 34
289 48
```

< > PROBLEMS

Name

✓ A plus B

full_in.txt

sample_in.txt

sample_out.txt

statement.txt

> Fall in Line

> Line by Line

✓ Line of Delivery (Part 1)

full_in.txt

sample_in.txt

sample_out.txt

statement.txt

> Line of Delivery (Part 2)

> Sort array

> Walk the Line

statement.txt

This problem shares some similarities with problem D2, with key differences in bold.

Candice is playing a solitaire game of [curling](https://en.wikipedia.org/wiki/Curling) on a $\backslash(1\backslash)$ -dimensional ice, **using stones of negligible size**. She will throw $\backslash(N\backslash)$ stones (numbered $\backslash(1\backslash)$ to $\backslash(N\backslash)$) from positions $\backslash(G\backslash)$ units to the right. In curling, though we say a stone is "thrown", it's actually sliding on the ice.

The $\backslash(i\backslash)$ th stone will be thrown with energy $\backslash(E_i\backslash)$, and will travel $\backslash(E_i\backslash)$ units to the right unless it hits another stone, in which case it will transfer its remaining energy to the stone it hits. Formally, the process until all stones are stationary:

****If the moving stone is at the same position as a stationary stone, the moving stone stops, and the now the moving stone with the remaining energy of the previous moving stone.****
*** Otherwise, the moving stone moves $\backslash(1\backslash)$ unit to the right and its energy is reduced by $\backslash(1\backslash)$. If the stone has energy $\backslash(0\backslash)$, it becomes stationary.**

After all of the stones are thrown, which stone is closest to the goal position $\backslash(G\backslash)$, and how far away is it?

Constraints

$\backslash(1 \leq T \leq 85\backslash)$

$\backslash(1 \leq N \leq 300\backslash)$

$\backslash(1 \leq E_i, G \leq 10^9\backslash)$

****In each test case, no two stones are thrown with the same energy.****

The sum of $\backslash(N\backslash)$ across all test cases is at most $\backslash(2 \cdot 10^5\backslash)$.

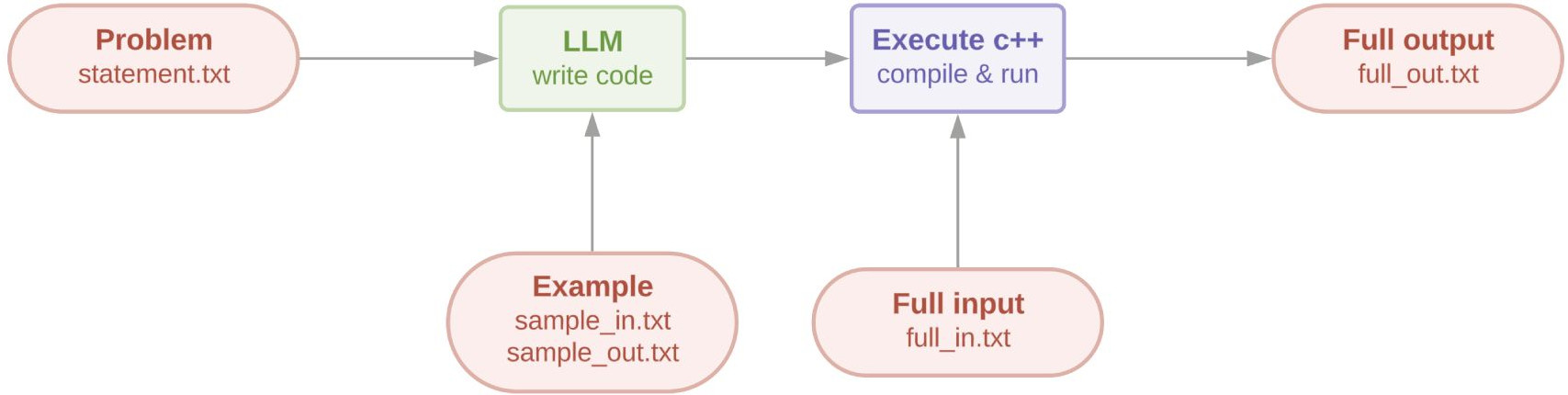
Input Format

Input begins with an integer $\backslash(T\backslash)$, the number of test cases. Each case starts with a line that contains $\backslash(N\backslash)$ and $\backslash(G\backslash)$. Then $\backslash(N\backslash)$ lines follow, the $\backslash(i\backslash)$ th of which contains $\backslash(E_i\backslash)$.

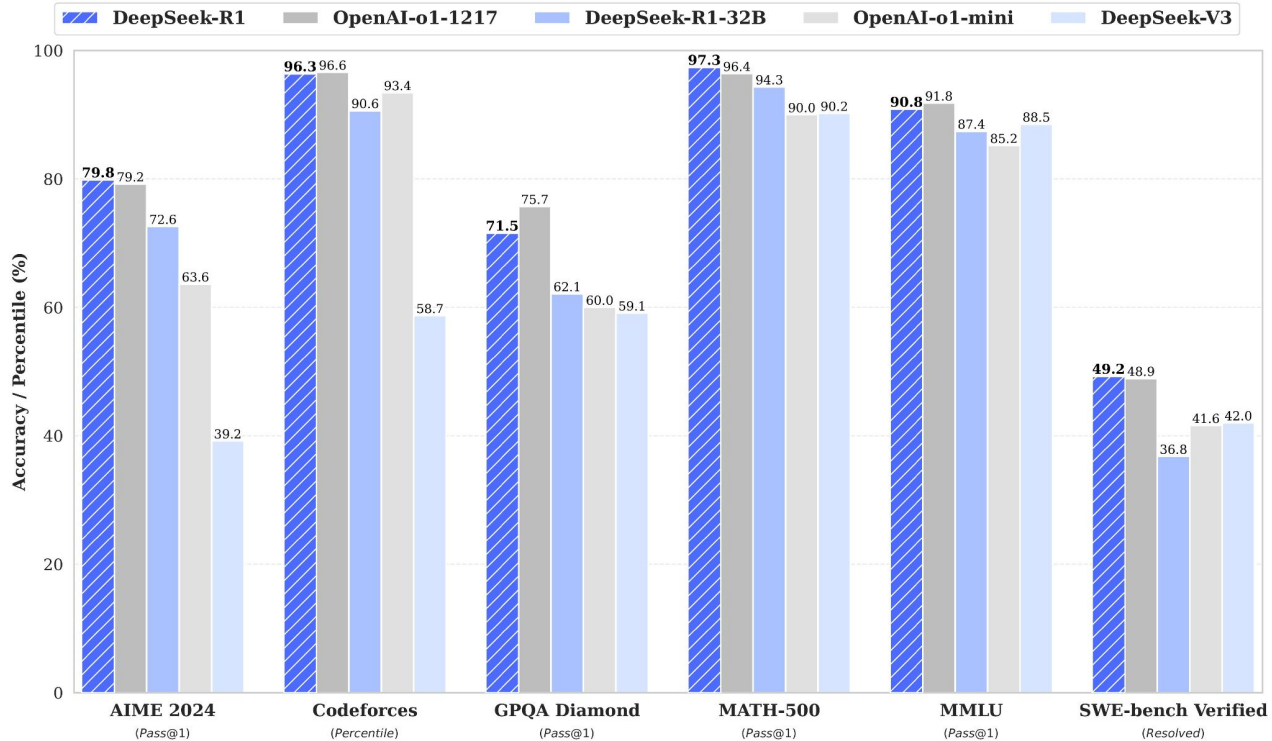
Output Format

For the $\backslash(i\backslash)$ th test case, print "Case #i: " followed by the index of the stone that ends up closest to the goal position $\backslash(G\backslash)$, and how far away it is from $\backslash(G\backslash)$. If there's a tie, output the stone with the lowest index.

MVP



deepseek-ai/ DeepSeek-R1



New chat



diagram.png

PNG 95.54KB

write a nodejs script for my mac that solves a competitive programming problem using Deepseek LLM API

I want to call the script like this:

```
$ node ./solve.js ./problem_dir/
```



Thinking... ^

Okay, the user wants a Node.js script for their Mac that solves a competitive programming problem using the Deepseek LLM API. The script should be called with a problem directory, which contains several files like the problem statement, input examples, and full inputs/outputs. Let me break down the requirements.

First, the script needs to read the problem statement from `problem_dir/problem_statement.txt`. Then, it should send this to the Deepseek LLM API to generate the C++ code. Once the code is generated, the script has to compile and run it against the sample inputs (`sample_in.txt`) to check if the output matches `sample_out.txt`. If it does, proceed to run the code with the full input (`full_in.txt`) and write the result to `full_out.txt`.

Hmm, I need to consider how to structure the script. Let's start by requiring necessary modules like `fs` for file handling, `child_process` for executing commands, and maybe `axios` for API calls. Wait, the user mentioned Deepseek LLM API. I need to check if they have an API key. Oh right, the script should include the API key, so the user needs to set it as an environment variable.

JS solve.js > main > requiredFiles

```
11  async function main() {
12      // Read problem files
13      const [statement, sampleIn, sampleOut] = await Promise.all([
14          fs.readFile(path.join(problemDir, 'statement.txt'), 'utf8'),
15          fs.readFile(path.join(problemDir, 'sample_in.txt'), 'utf8'),
16          fs.readFile(path.join(problemDir, 'sample_out.txt'), 'utf8')
17      ]);
18
19      // Generate solution code
20      const prompt = `Write a C++ solution for this programming problem.
21      The code must read from stdin and write to stdout. Ensure it passes the sample test case.
22
23      Problem Statement:
24      ${statement}
25
26      Sample Input:
27      ${sampleIn}
28
29      Sample Output:
30      ${sampleOut}
31
32      Provide only the C++ code in a code block.`;
33
34      try {
```

< > PROBLEMS

Name

▾ A plus B

full_in.txt

full_out.txt

sample_in.txt

sample_out.txt

solution

solution.cpp

statement.txt

> Fall in Line

> Line by Line

> Line of Delivery (Part 1)

> Line of Delivery (Part 2)

> Sort array

> Walk the Line

✕ solution.cpp

```
#include <iostream>
using namespace std;

int main() {
    int T;
    cin >> T;
    for (int i = 1; i <= T; ++i) {
        int A, B;
        cin >> A >> B;
        cout << "Case #" << i << ": " << A + B << endl;
    }
    return 0;
}
```



< > PROBLEMS

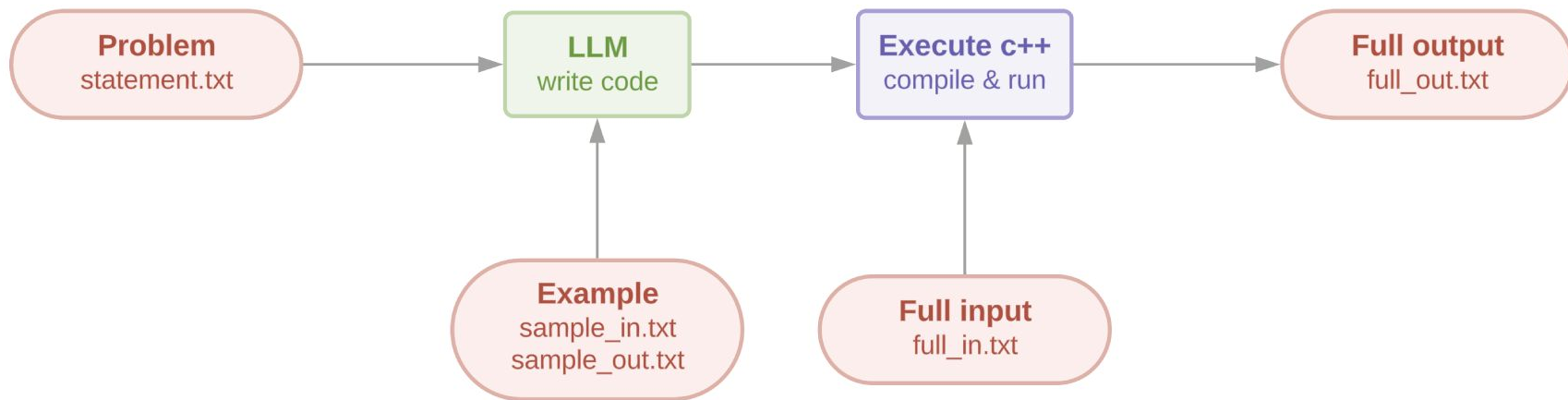
Name

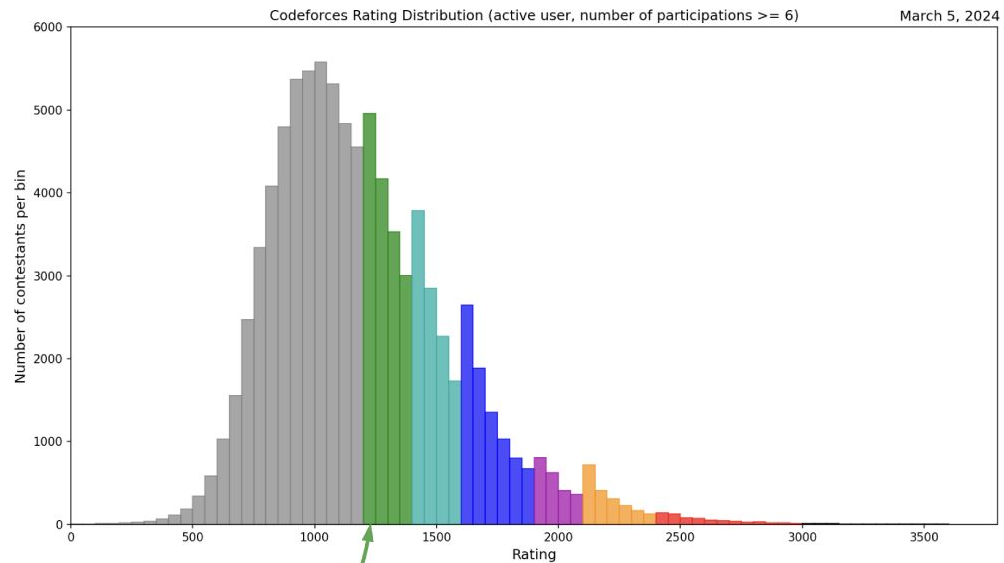
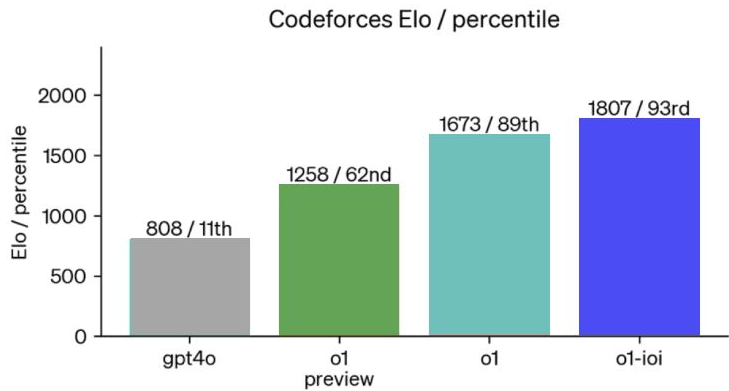
- ▼ A plus B
 - full_in.txt
 - full_out.txt**
 - sample_in.txt
 - sample_out.txt
 - solution
 - solution.cpp
 - statement.txt
- > Fall in Line
- > Line by Line
- > Line of Delivery (Part 1)
- > Line of Delivery (Part 2)
- > Sort array
- > Walk the Line

full_out.txt

```
Case #1: 52
Case #2: 13
Case #3: 50
Case #4: 1045
Case #5: 3
Case #6: 4
Case #7: 5
Case #8: 6
Case #9: 7
Case #10: 97
Case #11: 98
Case #12: 99
Case #13: 100
Case #14: 101
Case #15: 1001
Case #16: 1002
Case #17: 1003
Case #18: 1004
Case #19: 1005
Case #20: 1095
Case #21: 1096
Case #22: 1097
Case #23: 1098
Case #24: 1099
Case #25: 962
Case #26: 120
Case #27: 544
Case #28: 456
Case #29: 296
Case #30: 766
Case #31: 945
Case #32: 1044
Case #33: 676
Case #34: 337
Case #35: 862
Case #36: 450
```

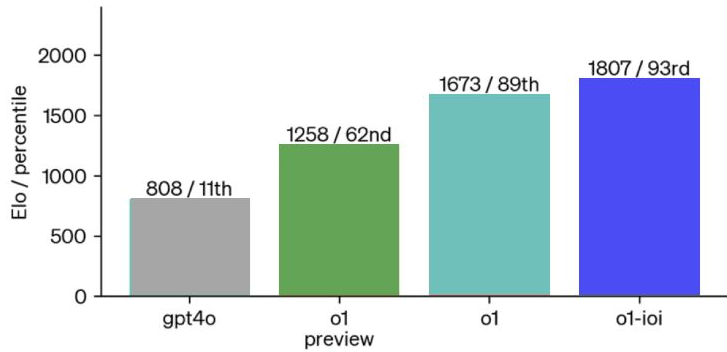
o1-preview





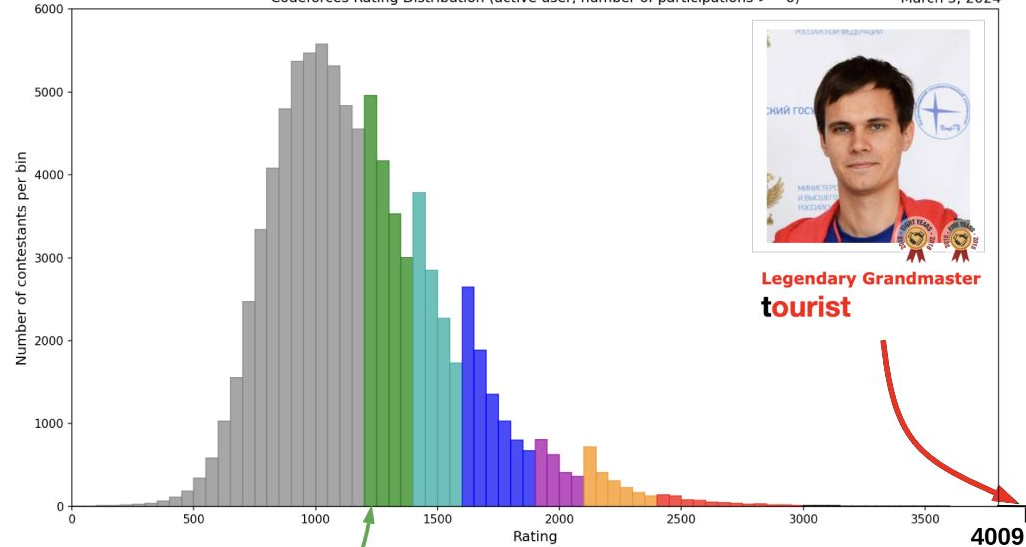
o1-preview
1258 Codeforces rating

Codeforces Elo / percentile



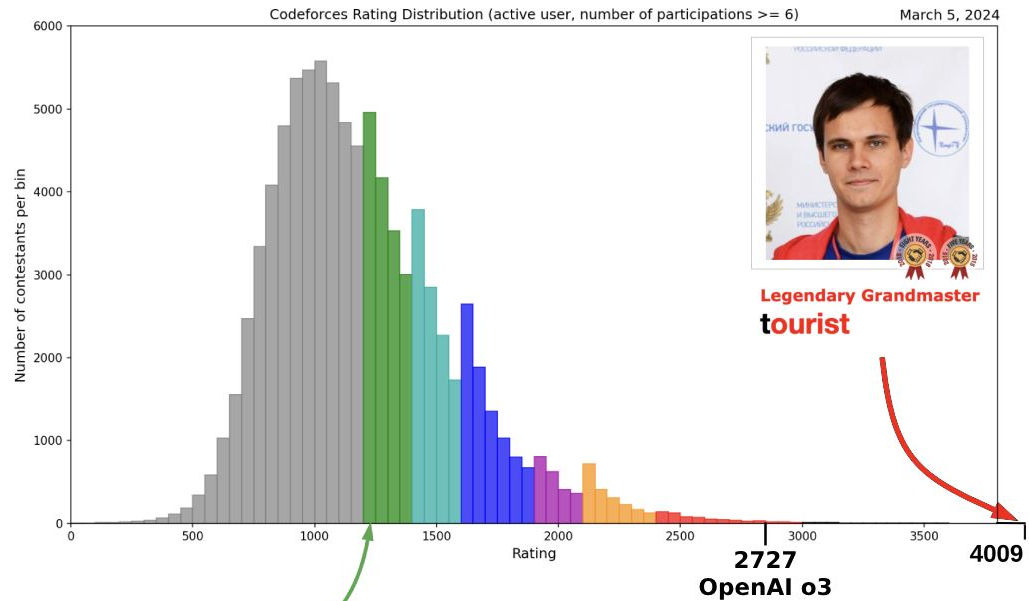
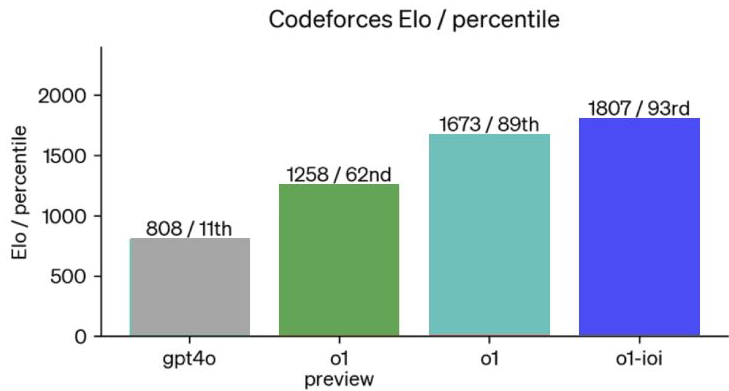
Codeforces Rating Distribution (active user, number of participations >= 6)

March 5, 2024



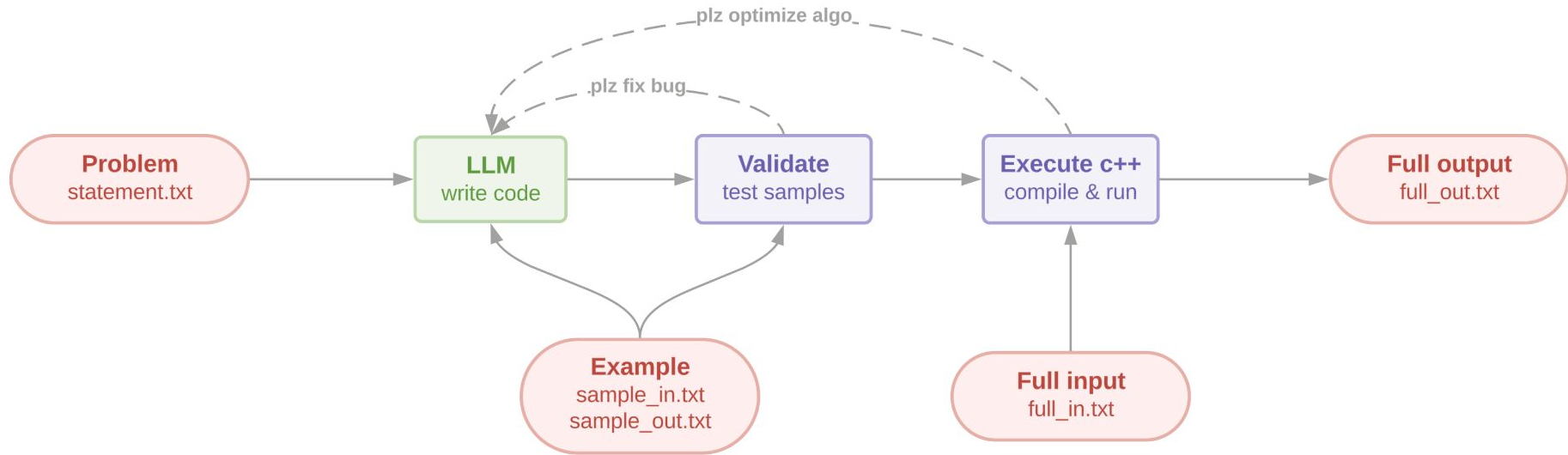
o1-preview

1258 Codeforces rating



o1-preview

1258 Codeforces rating





Hi, I'm DeepSeek.

How can I help you today?

below is the code that solves a competitive programming problem using LLM.

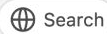
it works fine but I need you to make a more advanced version of it:

- 1) before computing the answer, validate the solution using the sample file (simply compare the expected result with output)
- 2) if something goes wrong (wrong answer or compilation error or execution takes longer than 10 sec) - send error back to the LLM thread, asking model to fix their code.

think hard how to refactor it nicely with OOP. I just want to launch the cli script and get my full_out.txt and solution.cpp ready.

THE CODE:

```
import path from 'path';
```



```

88     }
89   }
90   }
91 }
92
93 try {
94   const solver = new ProblemSolver(problemDir);
95   await solver.solve();
96   console.log('Successfully generated solution.cpp and full_out.txt');
97 } catch (error) {
98   console.error('Error:', error.message);
99   process.exit(1);
100 }
101
102 main().catch(err => {
103   console.error('Unexpected error:', err);
104   process.exit(1);
105 });

```

Key improvements and structure:

- OOP Design:** Encapsulated all problem-solving logic in the `ProblemSolver` class
- Error Handling:** Added custom error types and detailed error reporting
- Validation Pipeline:**
 - Compilation check
 - Sample input/output validation
 - Timeout handling (10 seconds)
- Retry Mechanism:** Up to 3 attempts with error feedback to LLM
- Modular Methods:** Each step (code generation, compilation, validation) has dedicated methods
- Improved Error Messages:** Clear error context sent back to LLM for better fixes
- File Management:** Centralized file paths and validation

Usage remains the same:

```
bash
```

Copy






























































```
node solve.js ./problem_dir/
```

```
































88   }
89   }
90   }
91 }
92
93 while (this.attempts < this.maxAttempts) {
94   try {
95     await this.generateCode();
96     this.compileCode();
97     await this.validateSample();
98     await this.processFullInput();
99     return;
100   } catch (error) {
101     this.handleError(error);
102     this.attempts++;
103   }
104 }
105
106 throw new ProblemSolverError(`Failed after ${this.maxAttempts} attempts`);
107 }
108
109 async generateCode() {
110   console.log(this.messages)
111   const completion = await this.openai.chat.completions.create({
112     model: "deepseek-reasoner",
113     messages: this.messages,
114   });
115
116   const response = completion.choices[0].message.content;
117   const codeBlock = response.match(/```cpp\n([\s\S]+?)\n```/);
118
119   if (!codeBlock) {
120     throw new ProblemSolverError('No code block found in LLM response');
121   }
122
123   await fs.writeFile(
124     path.join(this.problemDir, 'solution.cpp'),
125     codeBlock[1]
126   );
127
128   this.messages.push({
129     role: 'assistant',
130     content: response
131   });
132 }
133
134 compileCode() {
135   const result = spawnSync('g++', [
136     '-std=c++17',
137     '-o', 'solution',

```

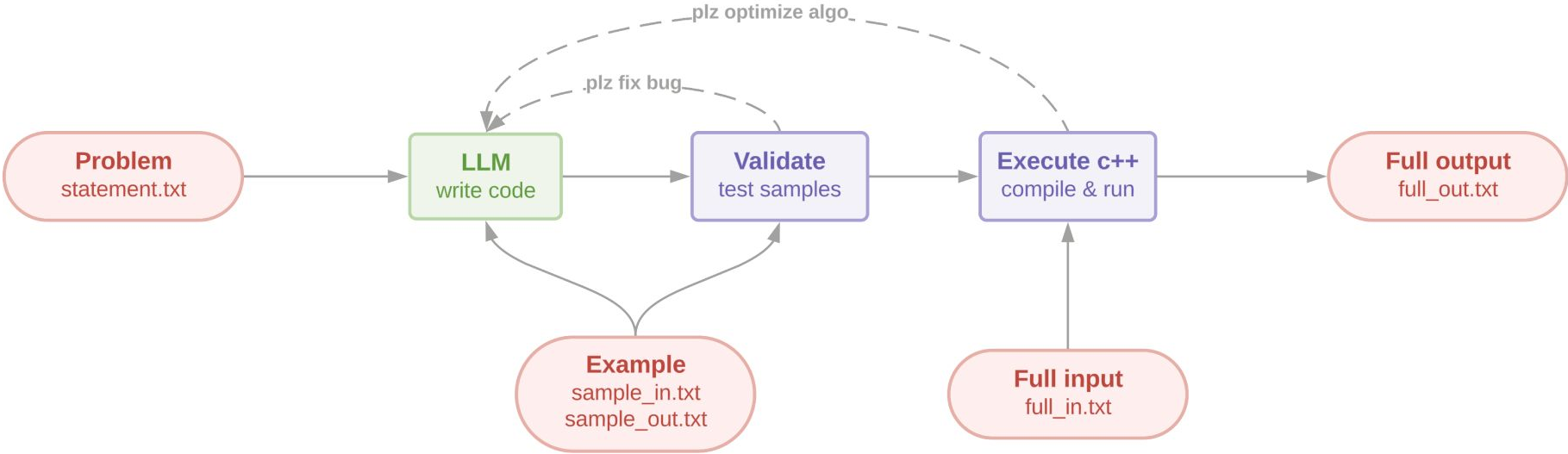
Scoreboard - Round 3 - AI Track

Rank	Contestant	Score	Total Time	A: Set, Cover	B: Least Commo...	C: Coin Change	D: Min-flow Max-...	E1: All Triplets S...	E2: All Triplets S...
1	  Artigenz Coders	22	0:10:19						
2	  Anton Pidkuiko (anton10xr)	0	0:00:00						
2	  Artigen Coders	0	0:00:00						
2	  David Tittsworth	0	0:00:00						
2	  Elizabeth Lijia Hong (MakingItHappen)	0	0:00:00						
2	  Geremie Yeo Yun Siang (Geremie Yeo + Sa...	0	0:00:00						
2	  Kateryna Lytvynets	0	0:00:00						
2	  Maheswari Rajendran	0	0:00:00						
2	  Maksym Bidnyi	0	0:00:00						
2	  Mridul Kumar	0	0:00:00						
2	  Nitin Gandem	0	0:00:00						
2	  Qi Tang	0	0:00:00						
2	  Ryoga Tanaka (evimalab)	0	0:00:00						
2	  Siva Kishore SK (Project Ramanujan)	0	0:00:00						
2	  Dongqi Huang	0	0:00:00						
2	  Tong Hui Kang (huikang)	0	0:00:00						
2	  Yi Han	0	0:00:00						

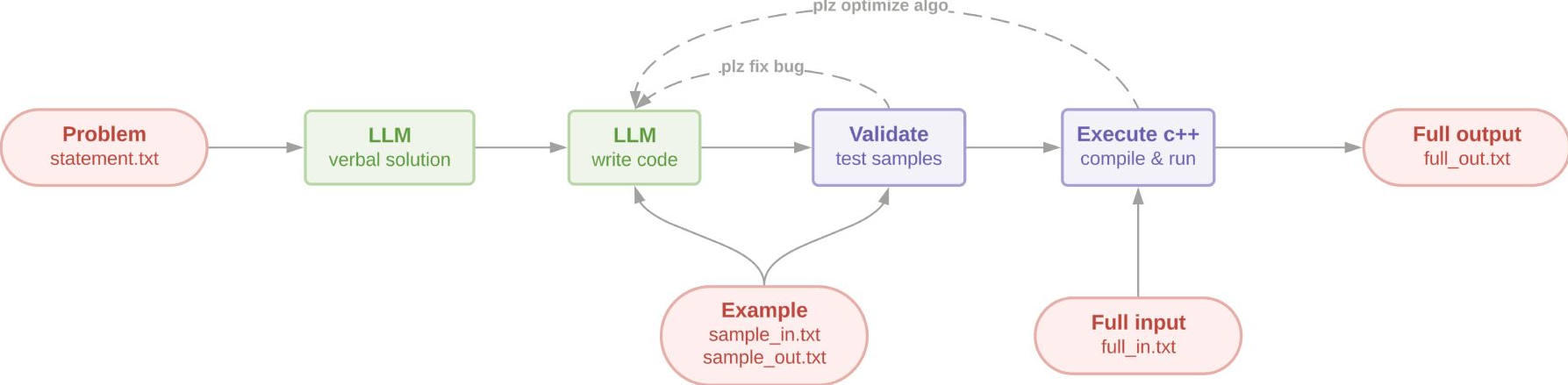
Scoreboard - Round 3 - Human Track

Rank	Contestant	Score	Total Time	A: Set, Cover	B: Least Commo...	C: Coin Change	D: Min-flow Max...	E1: All Triplets S...	E2: All Triplets S...
1	  Benjamin Qi (Benq)	96	8:07:22	✓	✓	✓	✓	✓	
2	  Максим Гороховский (maksim1744)	78	7:33:35	✓	✓	✓		✓	✓
3	  Ziqian Zhong	78	7:54:09	✓	✓	✓		✓	✓
4	  Yuhao Du (apiad)	78	9:06:51	✓	✓	✓		✓	✓
5	  邓明扬 (Slime)	78	11:37:37	✓	✓	✓		✓	✓
6	  Mateusz Radecki (Radewoosh)	74	4:55:42	✓	✓	✓		✓	
7	  Neal Wu	74	5:04:54	✓	✓	✓		✓	
8	  조승현 (ainta)	74	5:11:18	✓	✓	✓		✓	✗
9	  Nishimoto Masaki (maspy)	74	5:32:25	✓	✓	✓		✓	✗
10	  박상수 (molamola)	74	6:10:15	✓	✓	✓		✓	✗
11	  Ivan Safonov (isaf27)	74	6:28:06	✓	✓	✓		✓	
12	  Wojtek Nadara (Swistakk)	74	7:35:55	✓	✓	✓		✓	
13	  Yui Hosaka (hos.lyric)	74	7:37:12	✓	✓	✓		✓	
14	  Harris Leung (Xopowo)	74	8:39:46	✓	✓	✓		✓	✗
15	  Andrew He (ecnerwala)	61	5:04:30	✓		✓		✓	✓
16	  Hirotaaka Isa (HIR180)	57	4:30:59	✗	✓	✓		✓	✗
17	  Ilya Kornakov (ilyakor)	57	5:19:00	✗	✓	✓		✓	✗

Break down the problem



Break down the problem





Hi, I'm DeepSeek.

How can I help you today?

Below is the code that solves a competitive programming problem using LLM.
it works fine but I need you to make a more advanced version of it:

instead of asking to write the solution c++ code directly, use two LLM prompts:

1) writes VERBAL description of the solution, including big O notation

2) uses the description to write the c++ solution

CURRENT CODE:

...

```
import path from 'path';
```

```
import fs from 'fs/promises';
```

```
import { spawnSync } from 'child_process';
```



To solve this problem efficiently, we'll use a probabilistic approach leveraging randomization and hashing. The goal is to estimate the largest number of ants that are already aligned (colinear), and then calculate the minimum number of ants that need to move (which is $(N - K)$, where (K) is the maximum number of colinear ants).

Here's the strategy:

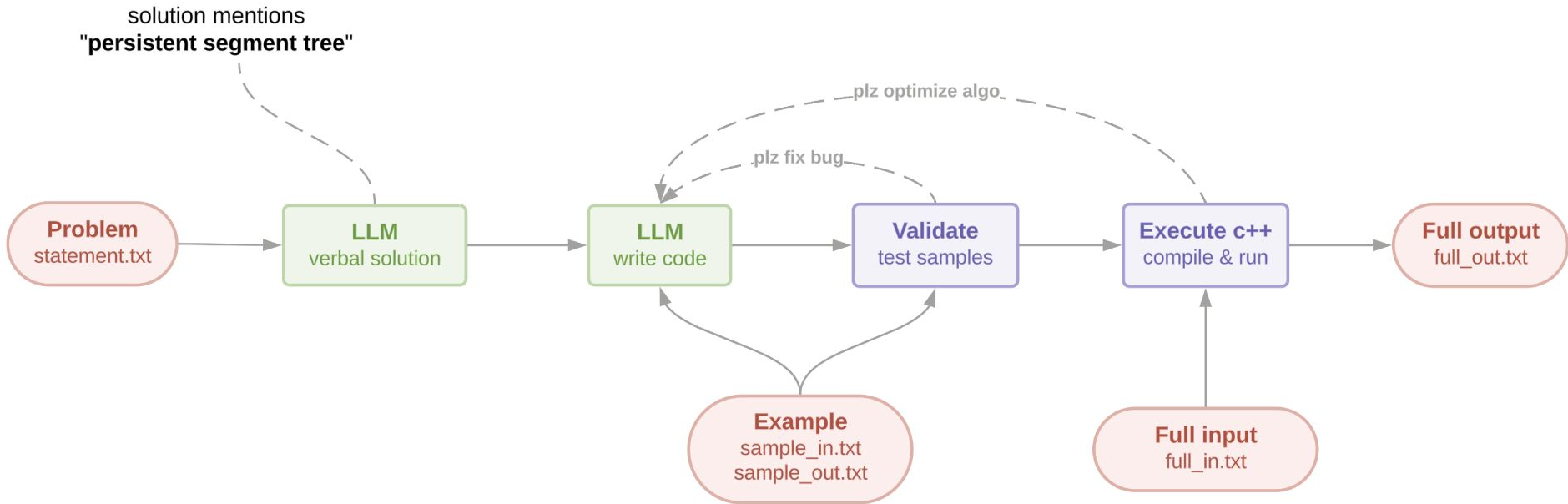
- Random Sampling**: For each test case, perform several iterations (e.g., 10 iterations). In each iteration:
 - Randomly select an ant as the reference point (P_i) .
 - For all other ants (P_j) , compute the slope between (P_i) and (P_j) .
 - Store the slopes in a hash map, reducing fractions to their lowest terms to handle sloped lines accurately.
 - The frequency of each slope gives us the number of ants aligned with (P_i) on a particular line.
- Estimating Maximum Colinear Ants**: Keep track of the maximum count of colinear ants found in all iterations. Due to the random selection and multiple iterations, there's a high probability that we'll find a line with a large number of ants if such a line exists.
- Calculating the Answer**: Subtract the maximum number of colinear ants (K) from the total number (N) to get $(N - K)$, the estimated minimum number of ants that need to move. Since any answer between (M) and $(2M)$ is acceptable, this estimation suffices.

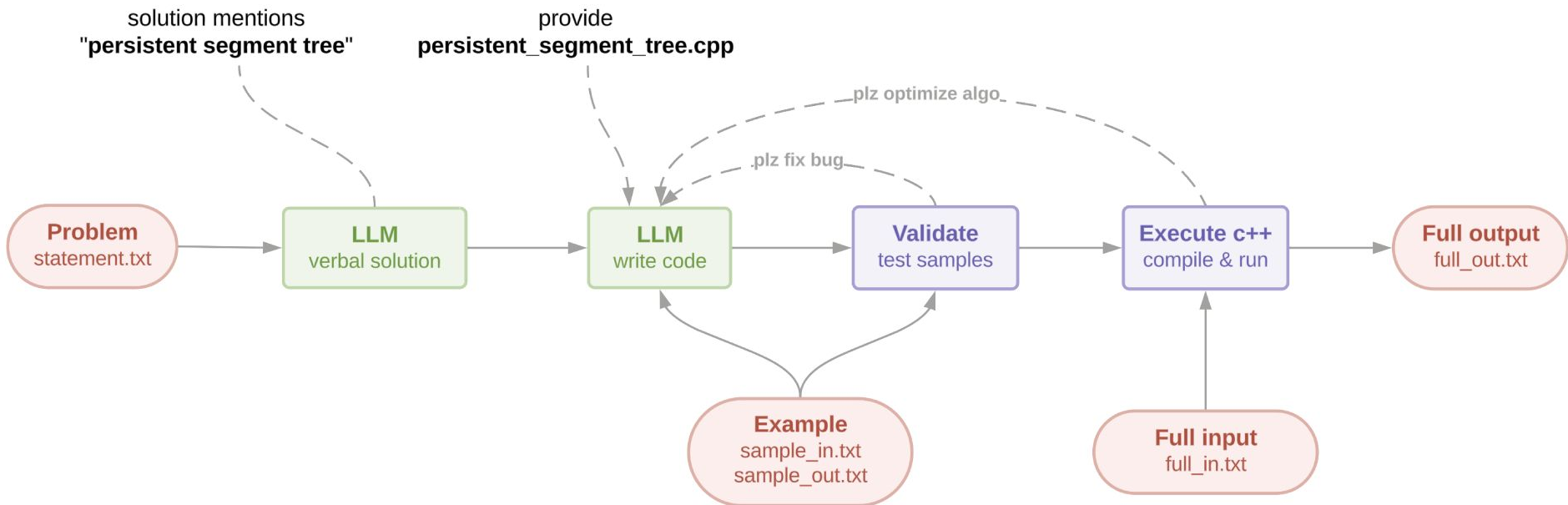
This method operates in $(O(s \cdot N))$ time per test case, where (s) is the number of iterations (a small constant), making it efficient for large (N) . By using random sampling and accepting approximate solutions within a tolerated range, we can provide a feasible answer within the time constraints.

PROBLEMS

Name

- > A plus B
- ▼ Fall in Line
 - actual_sample_out.txt
 - attack_vector.txt
 - full_in.txt
 - full_out.txt
 - full_output.txt
 - image_1528828545185163_ur
 - prompts_log.txt
 - sample_in.txt
 - sample_out.txt
 - solution
 - solution.cpp
 - statement.txt
- > Line by Line
- > Line of Delivery (Part 1)
- > Line of Delivery (Part 2)





master

3 Branches 0 Tags

Go to file

t

Add file

Code

Inishan Merge pull request #148 from mlarocca/patch-1 2026f37 · last month 362 Commits

.travis.yml	Remove dated items, update information and fix link redir...	7 years ago
CONTRIBUTING.md	Update CONTRIBUTING.md	8 years ago
LICENSE.md	Added OLYMPIADS IN INFORMATICS (an international jou...	9 years ago
README.md	Merge pull request #148 from mlarocca/patch-1	last month
README.pdf	Add pdf version of the awesome list (courtesy of markdo...	8 years ago
_config.yml	Set theme jekyll-theme-cayman and migrate Page Gener...	5 years ago
index.md	Set theme jekyll-theme-cayman and migrate Page Gener...	5 years ago

README CC-BY-4.0 license

edit menu

Awesome Competitive Programming

A curated list of awesome Competitive Programming, Algorithm and Data Structure resources.

Created with a view to connecting people to information, this list below contains a complete collection of all the fantastic resources I've collected over the course of my 11-year competitive programming career. I hope you'll find

About

A curated list of awesome Competitive Programming, Algorithm and Data Structure resources

codeforces.com/blog/entry/23054

- list
- awesome
- algorithm
- practice
- data-structure
- contest
- competitive-programming
- quora
- awesome-list
- programming-contests
- learning-materials
- reference-materials

Readme

CC-BY-4.0 license

Activity

12.9k stars

421 watching

2.6k forks

Report repository

Releases

No releases published

- ALGO_RAG_DATA
 - combinatorics
 - data_structure
 - fenwick_tree_2d.cpp
 - fenwick_tree_2d.cpp
 - fenwick_tree_interval.cpp
 - fenwick_tree_on_map.cpp
 - fenwick_tree.cpp
 - fenwick.cpp
 - hash_map.cpp
 - heavy_light_decomposition.cpp
 - heavy_light.cpp
 - kd_tree.cpp
 - KDtree.cpp
 - lazy_segment_tree.cpp
 - LCA_RMQ.cpp
 - LCA.cpp

```

fenwick_tree_2d.cpp x
data_structure > fenwick_tree_2d.cpp
1 // Efficient implementation of Fenwick 2D
2 /**
3  * Description: Computes sums  $a[i,j]$  for all  $i < I, j < J$ , and increases single elements  $a[i,j]$ .
4  * Requires that the elements to be updated are known in advance (call fakeUpdate() before init(
5  * Time:  $\mathcal{O}(\log^2 N)$ . (Use persistent segment trees for  $\mathcal{O}(\log N)$ .)
6  * Status: stress-tested
7  */
8
9 struct FT2 {
10     vector<vi> ys; vector<FT> ft;
11     FT2(int limx) : ys(limx) {}
12     void fakeUpdate(int x, int y) {
13         for (; x < sz(ys); x |= x + 1) ys[x].push_back(y);
14     }
15     void init() {
16         for (vi& v : ys) sort(all(v)), ft.emplace_back(sz(v));
17     }
18     int ind(int x, int y) {
19         return (int)(lower_bound(all(ys[x]), y) - ys[x].begin());
20     }
21     void update(int x, int y, ll dif) {
22         for (; x < sz(ys); x |= x + 1)
23             ft[x].update(ind(x, y), dif);
24     }
25     ll query(int x, int y) {
26         ll sum = 0;
27         for (; x; x &= x - 1)
28             sum += ft[x-1].query(ind(x-1, y));
29         return sum;
30     }
31 }

```

```

FILE: algo_rag_data/graph/MaxFlowHLPP.h | TITLE: Classic MaxFlow Algorithm |
FILE: algo_rag_data/graph/MaxFlowPR.h | TITLE: Max flow: Push relabel in  $O(V^2 E^{0.5})$  with gap heuristic |
FILE: algo_rag_data/graph/MinCostMaxFlowPR.h | TITLE: Min Cost Max Flow PR |
FILE: algo_rag_data/graph/MinCostMaxFlowSPFA.h | TITLE: Min Cost Max Flow - SPFA |
FILE: algo_rag_data/graph/MinimumCut.h | TITLE: Minimum cut between every pair of vertices (Stoer Wagner) |
FILE: algo_rag_data/graph/StableMarriage.h | TITLE: Matching algorithm. COMPLEXITY:  $M^2$  |
FILE: algo_rag_data/graph/StronglyConnected.h | TITLE: build DAG of strongly connected components. Returns: adjacency list of DAG |
FILE: algo_rag_data/graph/articulation_points.cc | TITLE: Block-Cut Tree (Articulation points / Biconnected components) |
FILE: algo_rag_data/graph/betweenness_centrality.cc | TITLE: Betweenness centrality of undirected unweighted graph (Brandes) |
FILE: algo_rag_data/graph/bfs_using_map.h | TITLE: BFS using map, so extra log factor |
FILE: algo_rag_data/graph/bipartite_coloring.h | TITLE: Bipartite coloring (color graph with 2 colors) |
FILE: algo_rag_data/graph/bipartite_edge_coloring.h | TITLE: returns vector of {vertex, id of edge to vertex} |
FILE: algo_rag_data/graph/bipartite_matching.cc | TITLE: Ford-Fulkerson' maximum bipartite matching |
FILE: algo_rag_data/graph/bridge.cc | TITLE: Bridge-Block Tree (Bridge / Two-edge connected component) |
FILE: algo_rag_data/graph/chromatic_number.cc | TITLE: Exact Algorithm for Chromatic Number |
FILE: algo_rag_data/graph/cycle Enumeration.cc | TITLE: Hawick and James' cycle enumeration |
FILE: algo_rag_data/graph/dijkstra.h | TITLE: Dijkstra |
FILE: algo_rag_data/graph/dominator_tree.cc | TITLE: Dominator Tree (Lengauer-Tarjan) |
FILE: algo_rag_data/graph/dynamic_reachability_dag.cc | TITLE: Italiano's dynamic reachability data structure for DAG |
FILE: algo_rag_data/graph/euler_tour_tree.cc | TITLE: Euler Tour Tree |
FILE: algo_rag_data/graph/eulerian_path_undirected.cc | TITLE: Undirected Eulerian Path (Hierholzer's algorithm) |
FILE: algo_rag_data/graph/find_triangles.h | TITLE: Find all cycles of length 3 (a.k.a. triangles) |
FILE: algo_rag_data/graph/floyd.h | TITLE: Floyd algorithm |
FILE: algo_rag_data/graph/ford_bellman.h | TITLE: Ford Bellman,  $O(N*M)$  |
FILE: algo_rag_data/graph/gabow_edmonds.cc | TITLE: General Graph Matching (Gabow-Edmonds) |
FILE: algo_rag_data/graph/hamilton_cycle_ore.cc | TITLE: Hamilton Cycle for graphs with Ore condition |

```

Look at the list and give {files: [..]} to include for code generation



Hi, I'm DeepSeek.

How can I help you today?

Below is the code that solves a competitive programming problem using LLM.

it works fine but I need you to make a more advanced version of it:

I want to help the system solving problems by including implementation of relevant algorithms & data structures. I put 200 correct c++ snippets into the dir `./algo_rag_data/[category]/[algo name].cpp`

The first line always has comment `//` with the title of the algo.

Please add one more JSON LLM call to determine which knowledge items to use. Give it list of all items like this:

| FILE: `./data_structure/fenwick_tree_2d.cpp` | TITLE: Efficient implementation of Fenwick 2D |

| FILE: `./graphs/dijkstra.cpp` | TITLE: Dijkstra algorithm |

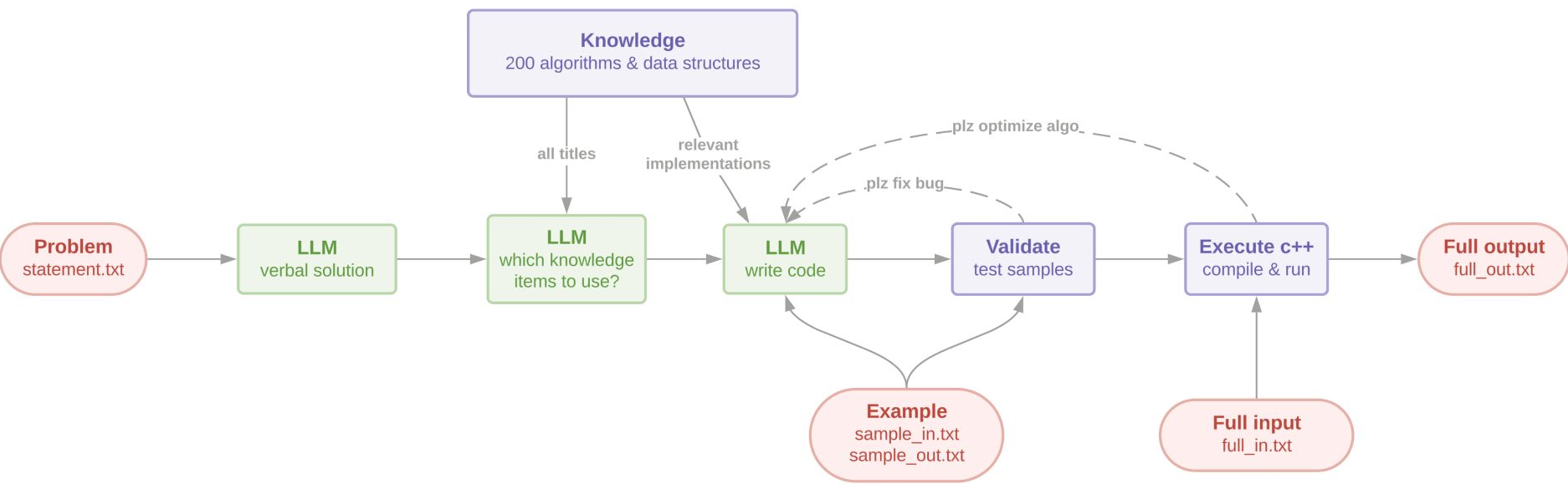
If based on initial description any of these items are needed, include such files into the code gen prompt.

Additionally, think how to refactor the code a bit, to make it more straightforward and avoid duplication.

CURRENT CODE:

```
import path from 'path';
```







Hi, I'm DeepSeek.

How can I help you today?

Below is the code that solves a competitive programming problem using LLM.

it works fine but I need you to make a more advanced version of it:

There are sample_in.txt and sample_out.txt used for validation. Use LLM to manually generate synthetic_sample_in.txt and synthetic_sample_out.txt in similar format, focusing on the problem edge cases. The test should be generated IN PARALLEL to the solution.

Additionally, think how to refactor the code a bit, to make it more straightforward and avoid duplication.

CURRENT CODE:

...

```
import path from 'path';
```

```
import fs from 'fs/promises';
```

```
import { spawnSync } from 'child_process';
```



DeepThink



Search

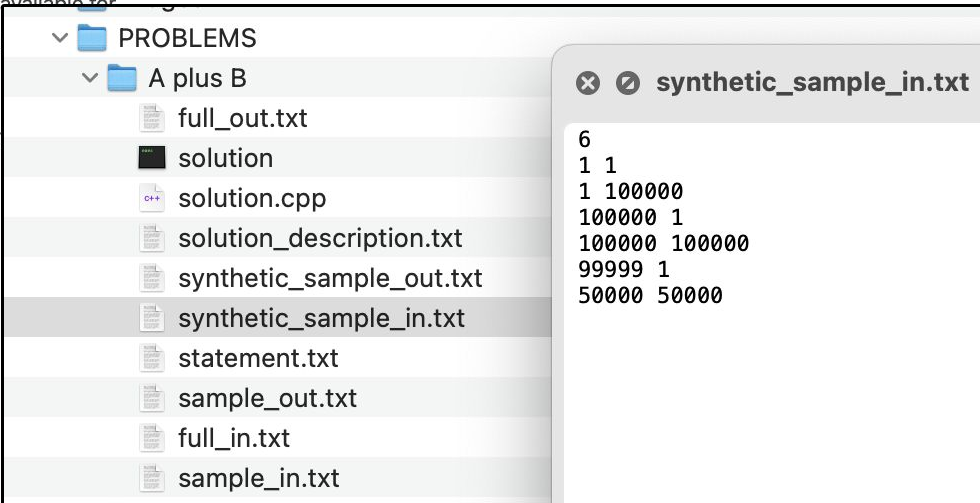


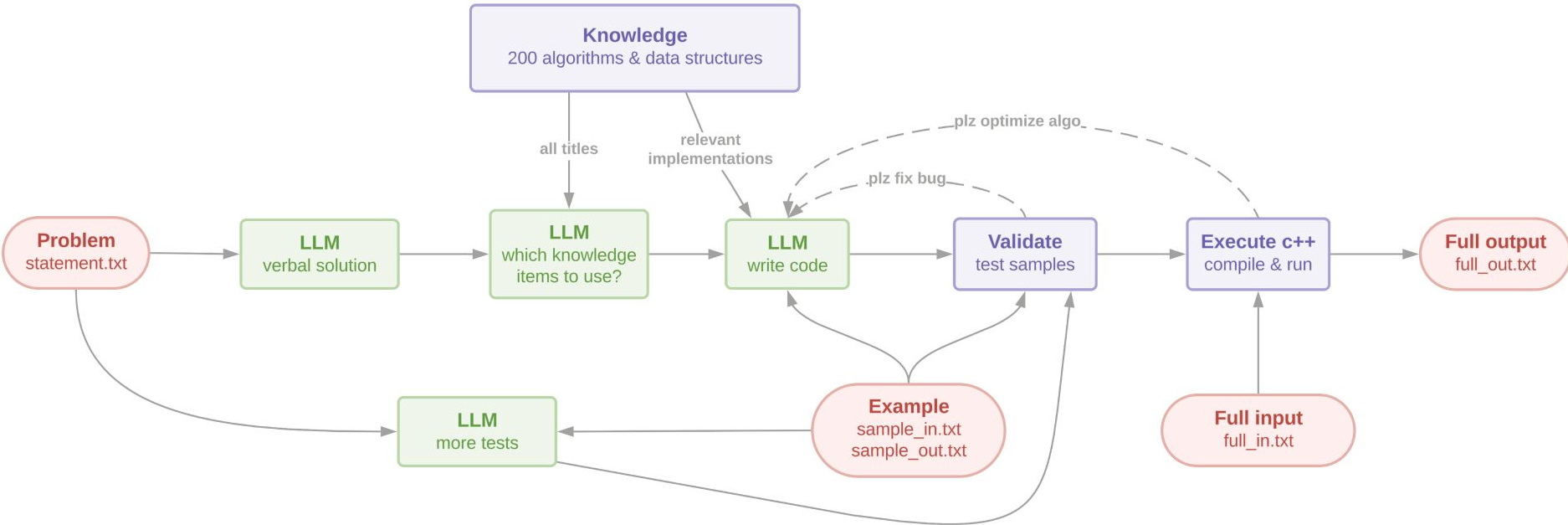
```
}  
  
main().catch(err => {  
  console.error('Unexpected error:', err);  
  process.exit(1);  
});
```

Explanation

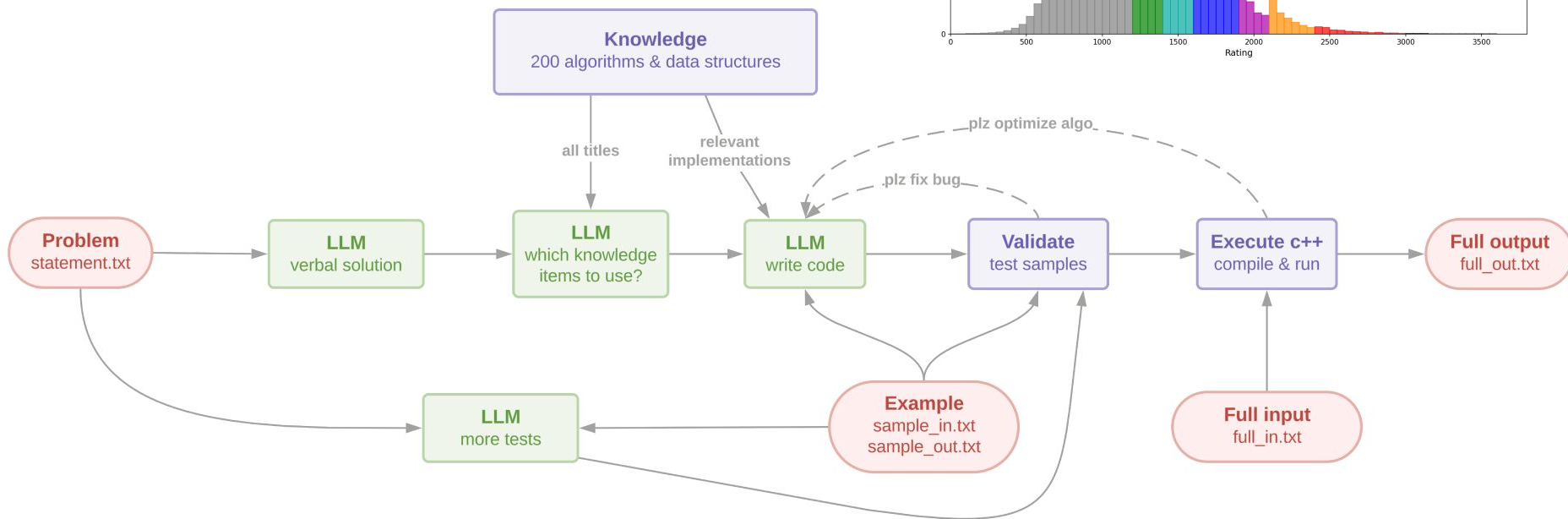
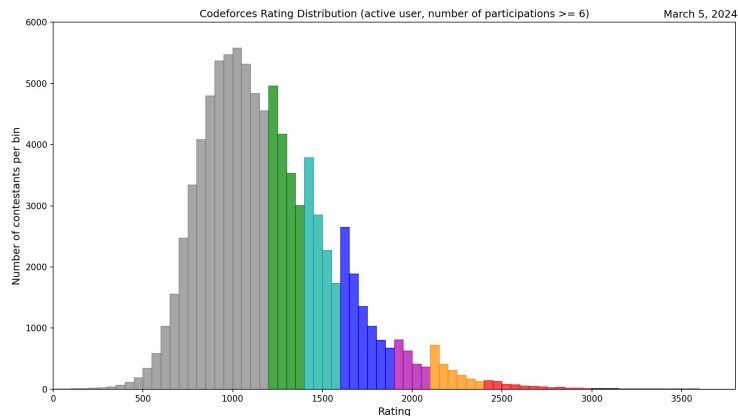
- Synthetic Test Generation:** The `generateSyntheticTests` method uses the LLM to create edge case test cases. These are saved to `synthetic_sample_in.txt` and `synthetic_sample_out.txt`.
- Code Refactoring:** The `runSolution` method centralizes the execution of the solution code, reducing redundancy. The `validateAllSamples` method runs both original and synthetic tests, enhancing validation coverage.
- Error Handling:** Errors during validation are augmented with test case context, improving debugging feedback. This helps the LLM understand which test case failed, facilitating iterative code correction.
- Parallel Processing:** Synthetic tests are generated during initialization, ensuring they are available for validation as soon as the solution code is generated. This parallel approach ensures completion from the first iteration.

This approach ensures that the solution is robust, efficient, and handles edge cases effectively with clean and maintainable code structure.



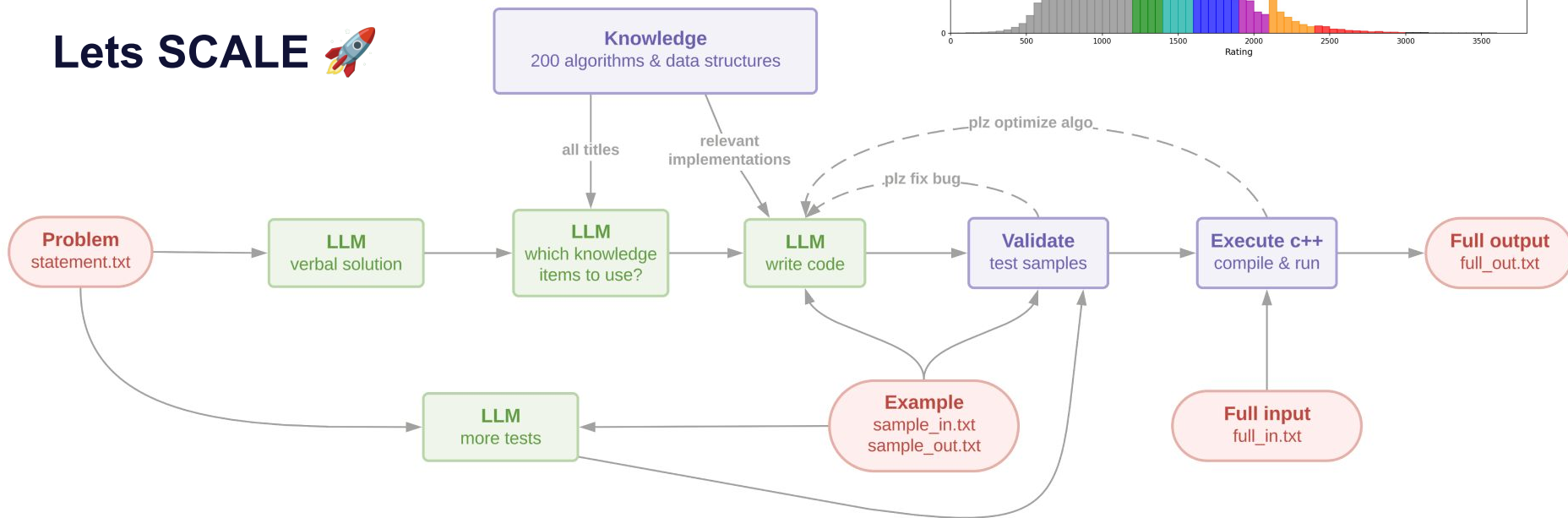
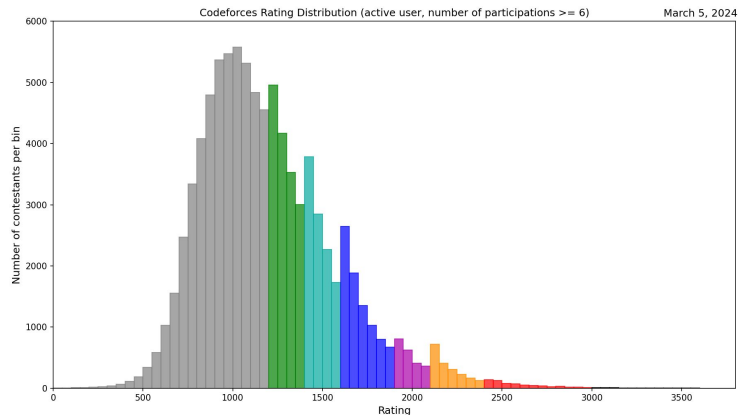


For hard problems will still fail
most of the time



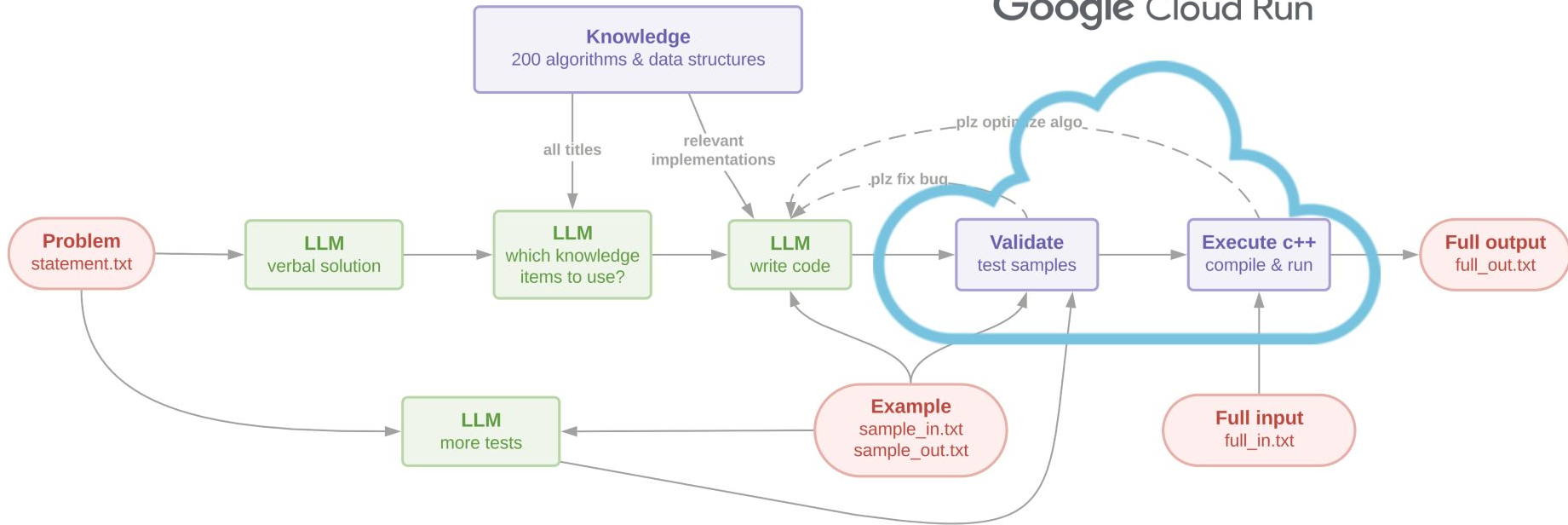
For hard problems will still fail
most of the time

Lets **SCALE** 





Google Cloud Run





Hi, I'm DeepSeek.

How can I help you today?

Below is the code that solves a competitive programming problem using LLM.

it works fine but I need you to make a more advanced version of it:

I don't like executing unreviewed C++ code on my machine and want to move it to Google Cloud Run lambdas. Please create a separate gcloud run that takes C++ code, input data - it compiles it, executes and returns the output data. Teach me how to deploy this code and use the service url in my file

Additionally, think how to refactor the code a bit, to make it more straightforward and avoid duplication.

CURRENT CODE

...

```
import path from 'path';
```

```
import fs from 'fs/promises';
```

```
import { spawnSync } from 'child_process';
```



DeepThink



Search



I'll help you create the Cloud Run service and refactor the code. Here's the step-by-step solution:

1. First, let's create the Cloud Run service:

server.js (Cloud Run Service)

```
javascript Copy

import express from 'express';
import { spawnSync } from 'child_process';
import { tmpdir } from 'os';
import { join } from 'path';
import { writeFile, rm } from 'fs/promises';

const app = express();
app.use(express.json());

app.post('/execute', async (req, res) => {
  const { code, input, timeout = 10000 } = req.body;
  const tempDir = tmpdir();
  const cppFile = join(tempDir, 'solution.cpp');
  const exeFile = join(tempDir, 'solution');

  try {
    // Write C++ code to temporary file
    await writeFile(cppFile, code);

    // Compile the code
    const compile = spawnSync('g++', [
      '-std=c++17',
      '-O2',
      '-o', exeFile,
      cppFile
    ], { encoding: 'utf-8' });

    if (compile.status !== 0) {
      return res.status(400).json({
        error: 'CompilationError',
        message: compile.stderr
      });
    }

    // Execute the compiled binary
    const execution = spawnSync(exeFile, {
      input,
      encoding: 'utf-8',
      timeout: parseInt(timeout)
    });

    if (execution.error?.code === 'ETIMEDOUT') {
      return res.status(400).json({
        error: 'TimeoutError',
        message: `Execution timed out after ${timeout}ms`
      });
    }

    if (execution.status !== 0) {

```

```
});

const port = process.env.PORT || 8080;
app.listen(port, () => {
  console.log(`Service running on port ${port}`);
});
```

Dockerfile

```
dockerfile Copy

FROM node:20-alpine

RUN apk add --no-cache build-base

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .

CMD ["node", "server.js"]
```

2. Deploy to Cloud Run:

```
bash Copy

# Build and tag the image
gcloud builds submit --tag gcr.io/YOUR_PROJECT_ID/cpp-executor

# Deploy the service
gcloud run deploy cpp-executor \
  --image gcr.io/YOUR_PROJECT_ID/cpp-executor \
  --platform managed \
  --allow-unauthenticated \
  --memory 2Gi \
  --timeout 900
```

3. Here's the refactored ProblemSolver class with Cloud Run integration:

refactored ProblemSolver.js

```
javascript Copy

import path from 'path';
import fs from 'fs/promises';
import { spawnSync } from 'child_process';
import OpenAI from 'openai';

class ProblemSolver {
  constructor(problemDir, algoRagDir = './algo_rag_data') {

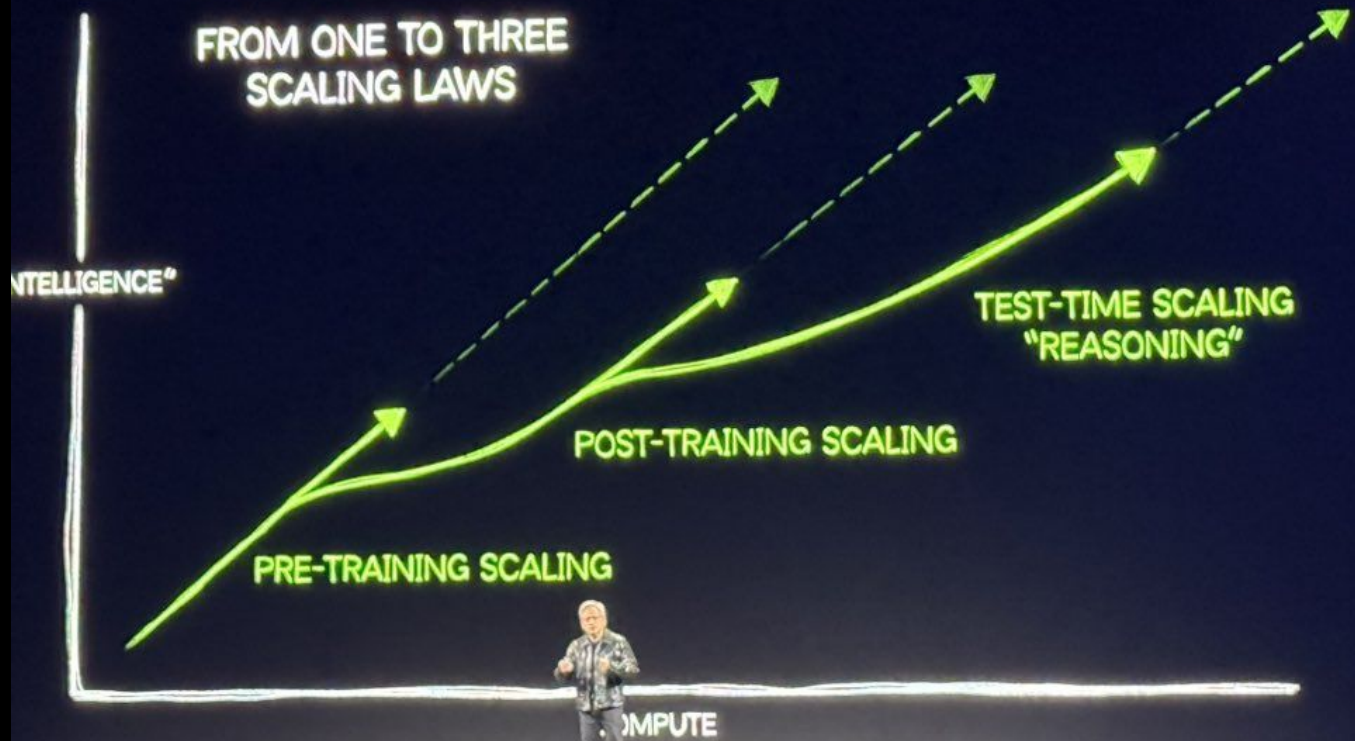
```




OpenAI Monthly Spend

\$457.50



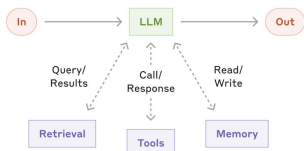


Stackfish - Competitive Programming Solver



Building block: The augmented LLM

The basic building block of agentic systems is an LLM enhanced with augmentations such as retrieval, tools, and memory. Our current models can actively use these capabilities—generating their own search queries, selecting appropriate tools, and determining what information to retain.



The augmented LLM

Workflow: Orchestrator-workers

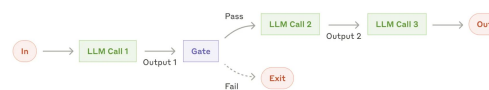
In the orchestrator-workers workflow, a central LLM dynamically breaks down tasks, delegates them to worker LLMs, and synthesizes their results.



The orchestrator-workers workflow

Workflow: Prompt chaining

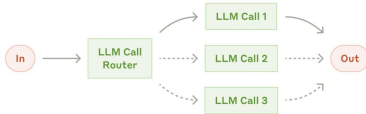
Prompt chaining decomposes a task into a sequence of steps, where each LLM call processes the output of the previous one. You can add programmatic checks (see "gate" in the diagram below) on any intermediate steps to ensure that the process is still on track.



The prompt chaining workflow

Workflow: Routing

Routing classifies an input and directs it to a specialized followup task. This workflow allows for separation of concerns, and building more specialized prompts. Without this workflow, optimizing for one kind of input can hurt performance on other inputs.



The routing workflow

Workflow: Evaluator-optimizer

In the evaluator-optimizer workflow, one LLM call generates a response while another provides evaluation and feedback in a loop.

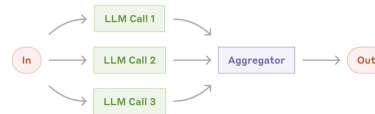


The evaluator-optimizer workflow

Workflow: Parallelization

LLMs can sometimes work simultaneously on a task and have their outputs aggregated programmatically. This workflow, parallelization, manifests in two key variations:

- **Sectioning:** Breaking a task into independent subtasks run in parallel.
- **Voting:** Running the same task multiple times to get diverse outputs.



The parallelization workflow

Learn more about agents

<https://anthropic.com/research/building-effective-agents>

