Mini-Project 5 Journal

Juejing Han
jhan446@gatech.edu

1 AGENT IMPLEMENTATION

To solve the Monster Diagnosis Problem, my agent follows the **Diagnosis as Abduction** approach and utilizes the **Iterative Deepening Depth-First Search (IDDFS)** [1] **algorithm** to arrive at the optimal solution (i.e., the smallest subset of diseases).

Mapping the inputs – the patient's vitamin profile and the disease candidates – is the core of translating the **Data Space** and **Hypothesis Space** concepts to the Monster Diagnosis Problem. In this report, the Data Space consists of 26 vitamin attributes, and the Hypothesis Space contains up to 24 disease candidates. Since each disease can be included or excluded, the search space grows exponentially with up to 2^{24} possible combinations. A Breadth-First Search approach would suffer from a memory explosion. Therefore, IDDFS is utilized for memory efficiency.

The agent performs iterative deepening on the solution size k (the maximum number of diseases allowed), where k increases from 0 to N, the total number of disease candidates. For each $k = 0, 1, \ldots, N$, the agent runs a Depth-First Search (DFS) over all N disease candidates with a **pruning strategy**, ensuring that the first solution found is guaranteed to be the minimum subset.

In each DFS iteration, my agent explores a binary decision tree over the N disease candidates by including or excluding each disease. For each disease, my agent applies the **Generate and Test** method by generating two branches (include and exclude) before recursing to the next disease candidate. The agent **prunes branches** when the selected set exceeds the current limit k, or when the remaining candidates cannot possibly satisfy the patient's symptoms (i.e., match each vitamin level) based on the current selected set.

When the optimal subset is found during search (i.e., all diseases have been considered and the patient's vitamin levels are perfectly matched), my agent

1 https://en.wikipedia.org/wiki/Iterative_deepening_depth-first_search.

immediately returns the stored path representing the minimum disease subset – no backtracking needed. If no valid subset is found after all iterations, my agent returns an empty list as a safe fallback.

2 AGENT PERFORMANCE

My agent successfully passes 20 out of 20 test cases during the performance evaluation, yielding correct results for all 20 Monster Diagnosis problems. It receives full credit and does not struggle with any particular cases.

As mentioned in Section 1, my agent utilizes IDDFS to find the optimal outcome. This approach ensures completeness and guarantees that the first solution is the smallest subset. The pruning mechanism enhances efficiency by eliminating infeasible branches, preventing exploration of combinations that cannot match the patient's vitamin profile.

3 AGENT EFFICIENCY

My agent efficiently passes 20 out of 20 test cases within the allowed runtime and memory limits, earning full credit for the performance evaluation.

My agent utilizes IDDFS with pruning to guarantee the optimal solution while maintaining low memory usage. IDDFS is the dominant process contributing to the overall time complexity.

IDDFS has a time complexity of $O(b^d)$, where b is the branching factor and d is the depth of the goal[2]. In the Monster Diagnosis Problem, $b = 2$ since each disease candidate can be either included or excluded, and the search depth corresponds to $d = N + 1$, where N is the number of disease candidates. Thus, the simplified time complexity is $O(2^N)$. Additionally, at each node, my agent performs constant-time operations across all vitamin attributes ($M = 26$). Therefore, the overall time complexity is $O(2^N \times M)$, which can be further simplified to $O(2^N)$ since M is constant.

As the number of diseases (N) grows, the total runtime increases exponentially with N. However, with the pruning mechanism, my agent stops exploring

2 https://youcademy.org/graph-iterative-deepening-dfs/.

infeasible branches early, resulting in a much faster practical runtime than the theoretical estimate.

4 AGENT OPTIMIZATION

My agent does two things to arrive at answers efficiently.

IDDFS Algorithm. My agent employs the IDDFS framework to balance optimality and memory efficiency by combining the space efficiency of DFS with the search completeness of BFS[3]. It guarantees that the first solution found is the optimal result (the smallest subset), while ensuring the minimal memory usage. IDDFS explores solutions by incrementally increasing the depth limit, resulting in optimal memory consumption and avoiding large frontier storage.

Pruning Mechanism. My agent integrates an early-stopping pruning strategy to maintain efficient performance. Before expanding a branch, it evaluates the remaining candidate diseases and only explores those with the potential to yield a valid result. Branches that cannot possibly match the target vitamin profile are terminated immediately. This pruning strategy eliminates a large portion of the search tree, reducing redundant computation and significantly improving runtime efficiency.

5 AGENT VS. HUMAN

My agent does not solve the problem the same way I do.

My agent uses IDDFS to achieve completeness and memory efficiency while ensuring the optimal outcome. It also incorporates a pruning strategy to improve efficiency by preventing redundant branch exploration.

However, the IDDFS process is too complex for me to execute manually. As a human, I would start with the most similar disease candidates to the target vitamin profile and proceed using the Generate and Test method – generating different combinations and testing the outcomes. This approach works well for small test cases (both small Data Space and Hypothesis Space), such as the two initial local examples. However, as the number of disease candidates increases, my ability to solve the Monster Diagnosis Problem degrades dramatically.

3 https://ai2-iiith.vlabs.ac.in/exp/iterative-deepening-dfs/theory.html.

My agent outperforms a human in efficiency.

My agent successfully passes two local test cases in about 0.53 ms, which is undoubtedly faster than human performance. It would take several minutes for me to analyze the 26 vitamin attributes, let alone solve two test cases within a millisecond.

People approach the problem similarly.

The Monster Diagnosis Problem is straightforward, and most people would likely reason about it by comparing vitamin attributes and evaluating different combinations. Although people might vary in their interpretative approach, the overall reasoning process remains consistent.