

PROJECT 1: SEARCH METHODS

PROMPT DOCUMENTATION

Joshua Castro

Professor Jesse Lowe

CS 461 Introduction to AI

7 March 2025

Documentation of Prompts & Code Snippets

Foreword:

These prompts are based on my own self-coded portions. After having coded a working framework on my own first, I then utilized ChatGPT4o to further enhance my code for:

1. certain implementations
2. finding logical errors
3. Unittesting
4. Efficiency of code expression

Legend:

- User = Me
- Assistant = ChatGPT4o

1. Prompt: Evaluate for Correct Behavior

User Prompt:

(Attached program files)

Can you evaluate the proposed code for correct behavior?

(No code snippet was returned in that direct answer; the Assistant responded with a long analysis rather than code.)

2. Prompt: “Fix ID-DFS nuances”

User Prompt:

Can you implement fixes for the proposed potential issues and fix the ID-DFS nuances?

Assistant Code (showing how to adjust ID-DFS):

```

def id_dfs(graph, start, goal, max_depth=10):
    """
    Iterative Deepening DFS up to a given depth. Returns the path found
    (if any) and the cost as the number of edges. This version avoids
    using a global 'visited' set for each depth pass, to ensure all
    paths up to 'max_depth' are systematically explored.
    """

    def dls(node, goal, depth, path):
        if node == goal:
            return path, len(path) - 1
        if depth == 0:
            return None

        for neighbor in graph.get(node, []):
            if neighbor not in path:
                result = dls(neighbor, goal, depth - 1, path + [neighbor])
                if result is not None:
                    return result
        return None

    for depth in range(max_depth):
        result = dls(start, goal, depth, [start])
        if result is not None:
            return result

    return None, float('inf')

```

3. Prompt: “What about the UI issue?”

User Prompt:

What about the issue with the user interface?

Assistant Code (code snippet excerpt) showing how to **distinguish BFS/DFS cost** from **A*** cost in the UI:

```

def display_results(start, goal, method_name, search_method, graph, coordinates):
    print(f"\nRunning {method_name} from {start} to {goal}...")

    tracemalloc.start()
    start_time = time.time()

    if search_method in [search_algorithms.best_first_search,
search_algorithms.a_star_search]:
        path, cost = search_method(graph, start, goal, coordinates)
    else:
        path, cost = search_method(graph, start, goal)

    end_time = time.time()
    memory_used = tracemalloc.get_traced_memory()[1] / 1024
    tracemalloc.stop()

    if path:
        print("Path found:", " -> ".join(path))

        if method_name in ["BFS", "DFS", "ID-DFS", "Best-First Search"]:
            print(f"Number of edges (cost): {cost}")
        else:
            # A* Search
            print(f"Total distance (cost): {cost:.2f} km")

        print(f"Time taken: {end_time - start_time:.4f} seconds")
        print(f"Memory used: {memory_used:.2f} KB")
    else:
        print("No path found.")

```

4. Prompt: “Based on assignment, what is the cost to output?”

User Prompt:

Based on these assignments qualifications, what should I be outputting in terms of cost?

(No direct code snippet was provided. The Assistant explained that every method should show real “node-to-node” distances for the final output, even BFS/DFS.)

5. Prompt: “Showcase the code for proposed fixes.”

User Prompt:

Can you showcase the code for the proposed fixes?

Assistant Code: Provided consolidated files. The crucial snippet for `search_algorithms.py` (clean version) is:

```
# search_algorithms.py

from collections import deque
import heapq
import graph_setup # Import graph_setup for distances

def bfs(graph, start, goal):
    # BFS code...

def dfs(graph, start, goal):
    # DFS code...

def id_dfs(graph, start, goal, max_depth=10):
    # ID-DFS code (with improved recursion)...

def best_first_search(graph, start, goal, coordinates):
    # Best-first code, using haversine_distance as a heuristic

def a_star_search(graph, start, goal, coordinates):
    # A* code, summing real travel cost for 'g'
```

Followed by a snippet of **user_interface.py** that sums real distances at the end:

```
# user_interface.py

def display_results(start, goal, method_name, search_method, graph, coordinates):
    # ...
    if search_method in [best_first_search, a_star_search]:
        path, cost = search_method(graph, start, goal, coordinates)
    else:
        path, cost = search_method(graph, start, goal)

    # Sum real distance:
    total_distance = 0.0
    for i in range(len(path) - 1):
        total_distance += graph_setup.haversine_distance(path[i], path[i+1],
coordinates)

    print(f"Total distance: {total_distance:.2f} km")
    # also prints time, memory, etc.
```

6. Prompt: “How is this output?”

User Prompt:

How does this output look?: (BFS, DFS, ID-DFS, etc.)

(Assistant responded with no new code snippet, just confirmation that it's correct.)

7. Prompt: “Seconds are zero, what better measurement?”

User Prompt:

For time, because seconds for all of them is zero, what time measurement can we use to evaluate the differences more effectively between the search algorithms?

Assistant Code (showing how to use `time.perf_counter()`):

```
import time

start_time = time.perf_counter()
# ... run search ...
end_time = time.perf_counter()
elapsed_time = end_time - start_time
print(f"Time taken: {elapsed_time:.6f} seconds")
```

8. Prompt: “Can we use method 1 for higher resolution timing?”

User Prompt:

Can we use method 1 for higher resolution timing?

Assistant Code (Integrating `time.perf_counter()` into `user_interface.py`):

```
def display_results(...):
    ...
    start_time = time.perf_counter()
    ...
    end_time = time.perf_counter()

    elapsed_time = end_time - start_time
    print(f"Time taken: {elapsed_time:.8f} seconds")
```

9. Prompt: “Implement these changes?”

User Prompt:

Can you implement these changes into my code?

Assistant gave the updated `user_interface.py` with `time.perf_counter()`:

```
def display_results(...):
    tracemalloc.start()
    start_time = time.perf_counter()
    ...
    end_time = time.perf_counter()
    memory_used = tracemalloc.get_traced_memory()[1] / 1024
    tracemalloc.stop()
    ...
```

10. Prompt: “Develop unit tests for each module.”

User Prompt:

Given the updated files, can you develop unit tests for each module/function in alignment with the requirements?

Assistant Code:

`test_graph_setup.py`:

```
import unittest, tempfile, os, graph_setup

class TestGraphSetup(unittest.TestCase):
    def test_haversine_distance(self):
        coords = {"Wichita": (37.6872, -97.3301), "Topeka": (39.0558, -95.6894)}
        distance = graph_setup.haversine_distance("Wichita", "Topeka", coords)
        self.assertTrue(210 < distance < 240)

    def test_load_coordinates(self):
        # create temp CSV, test reading
        ...
```



```
def test_load_adjacencies(self):
    # create temp adjacency file
    ...
```

- `test_search_algorithms.py` (testing BFS, DFS, ID-DFS, etc.)
- `test_user_interface.py` (mocking user input).

11. Prompt: “My test fails for Haversine distance 208.84 vs 210–240.”

User Prompt:

my results: FAIL: test_haversine_distance ...
AssertionError: False is not true : Expected ~220km, but got 208.84km

Assistant Code recommended:

```
self.assertAlmostEqual(distance, 220.0, delta=20.0,
    msg=f"Expected ~220km, but got {distance:.2f}km")
```

12. Prompt: “Which test function to apply this in?”

User Prompt:

Which test function can I apply this change in to keep the overall flow?

(Assistant explained it belongs in `test_haversine_distance`, no new snippet beyond that.)

13. Prompt: “Can we use option 2?” (A delta approach)

User Prompt:

Can we use option 2?

Assistant Code repeated the final snippet:

```
self.assertAlmostEqual(distance, 220.0, delta=20.0,  
                        msg=f"Expected ~220km, but got {distance:.2f}km")
```

14. Prompt: “Does the code meet requirements?”

User Prompt:

Does my code meet the proposed requirements?

(Assistant explained yes, except if you want a time-out. No snippet.)

15. Prompt: “How to implement a time-out?”

User Prompt:

how would we implement a timeout?

Assistant explained in detail: **Method 1** (in-code time checks) or **Method 2** (signal-based). Provided BFS snippet:

```
def bfs(..., max_time=5.0):  
    start_time = time.perf_counter()  
    while queue:  
        if (time.perf_counter() - start_time) > max_time:  
            return None, float('inf')  
    ...
```

(Used for DFS, ID-DFS, etc. as well.)

16. Prompt: “Can we do Method 1, will it affect tests?”

User Prompt:

Can we implement Method 1. Will this affect unit tests in any regard or overall functionality of the program?

Assistant posted code:

```
def bfs(graph, start, goal, max_time=5.0):
    start_time = time.perf_counter()
    ...
    if (time.perf_counter() - start_time) > max_time:
        return None, float('inf')
```

17. Prompt: “Show me the implemented code for changed modules.”

User Prompt:

Can you show the code (only for the modules that have changed)

Assistant Code: Provided the entire `search_algorithms.py` with the new `max_time` checks and the entire `user_interface.py` using `time.perf_counter()`. For instance:

```
# search_algorithms.py
```

```
import time
from collections import deque
import heapq
import graph_setup
```

```
def bfs(graph, start, goal, max_time=5.0):
    start_time = time.perf_counter()
    ...
```

```
# user_interface.py
```

```
import time
import tracemalloc
import graph_setup
import search_algorithms
```

```
def display_results(...):
    start_time = time.perf_counter()
    ...
```

18. Prompt: “Are there examples in the files for tests?”

User Prompt:

Are there any examples in the files that can be used for the tests?

Assistant gave an example `test_integration.py` showing BFS on real data:

```
class TestIntegrationRealData(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.coordinates = ...
        cls.graph = ...

    def test_bfs_anthony_salina_exists(self):
        path, cost = bfs(cls.graph, "Anthony", "Salina", max_time=5.0)
        ...
```

19. Prompt: “Can we implement a full test of the previous?”

User Prompt:

Can we implement a full test for this?

Assistant posted a “single-file” integration test example:

```
import unittest
import graph_setup
import search_algorithms
```

```
class TestIntegrationWithRealData(unittest.TestCase):  
    # BFS, DFS, ID-DFS, etc. on "Anthony" -> "Salina"  
    # plus a no-route test for "Anthony" -> "Fake_City"
```

20. Prompt: “Return to method selection. Right now it loops to the very start.”

User Prompt:

The requirements are: Return to the search method selection ... Right now, my code loops to the very beginning.

Assistant posted a brand new `main()` structure that loads data once, picks start/goal, and loops over method choices:

```
def main():  
    coordinates = ...  
    graph = ...  
    while True:  
        # ask for start, goal  
        while True:  
            # pick a search method  
            display_results(...)  
            # ask "Another method on the same route? y/n"  
        # ask "New route? y/n"
```

21. Prompt: “Show me the code for making changes in affected modules.”

User Prompt:

Can you showcase the code for the made changes in affected modules?

Assistant posted the final:

1. `search_algorithms.py` with time-outs.
2. `user_interface.py` with the revised main loop.

22. Prompt: “With the implemented code, error occurs: no attribute `get_user_input`.”

User Prompt:

with the implemented code, this error occurred:

AttributeError: module 'user_interface' has no attribute 'get_user_input'

Assistant explained to either reintroduce `get_user_input()` or remove the old test. (No new code snippet there, just an explanation.)

23. Prompt: “Can we revise that specific unittest?”

User Prompt:

Can we revise that specific unittest to account for the new changes?

Assistant Code: A “deep unittest” mocking the new main loop:

```
import unittest
from unittest.mock import patch
import io
import user_interface

class TestUserInterfaceDeep(unittest.TestCase):
    @patch("sys.stdout", new_callable=io.StringIO)
    @patch("builtins.input")
    def test_main_single_method_then_exit(self, mock_input, mock_stdout):
        mock_input.side_effect = [
            "Anthony", # Start
            "Salina", # Goal
            "1",      # BFS
```

```
        "n",      # No more methods
        "n"       # No new route => exit
    ]
    user_interface.main()
    output = mock_stdout.getvalue()
    self.assertIn("Running BFS from Anthony to Salina", output)
    self.assertIn("Path found:", output)
    self.assertIn("Goodbye!", output)
```