

LAB-12.5: Algorithms with AI Assistance – Sorting, Searching, and Optimizing Algorithms

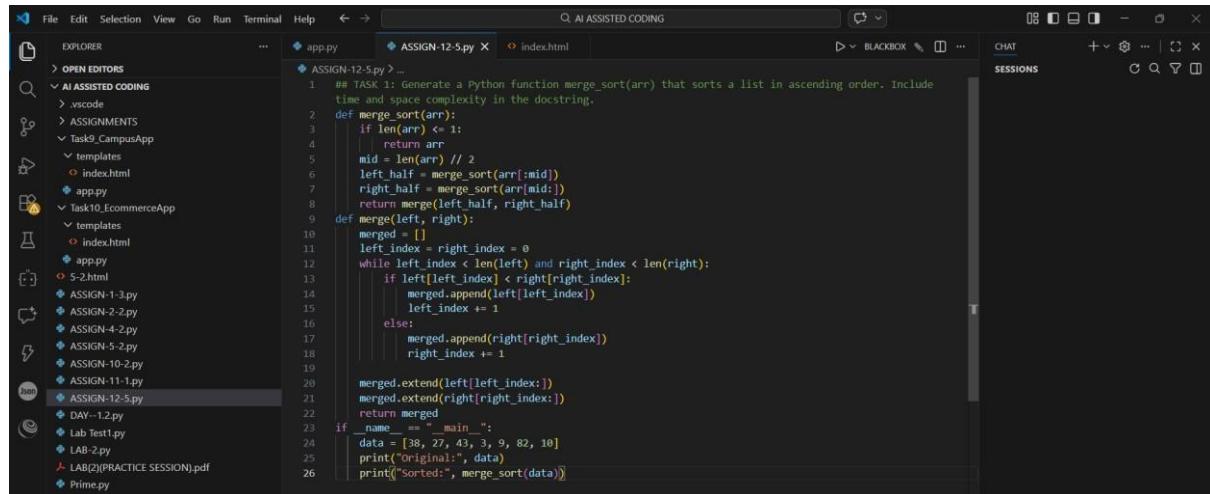
Vyshnavi Parisha

2403A51L34

B-52

Task 1 – Merge Sort:

Prompt: Generate a Python function `merge_sort(arr)` that sorts a list in ascending order. Include time and space complexity in the docstring.



The screenshot shows the VS Code interface with the code editor open. The file `ASSIGN-12-5.py` contains the following Python code for merge sort:

```
## TASK 1: Generate a Python function merge_sort(arr) that sorts a list in ascending order. Include time and space complexity in the docstring.
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
    left_half = merge_sort(arr[:mid])
    right_half = merge_sort(arr[mid:])
    return merge(left_half, right_half)
def merge(left, right):
    merged = []
    left_index = right_index = 0
    while left_index < len(left) and right_index < len(right):
        if left[left_index] < right[right_index]:
            merged.append(left[left_index])
            left_index += 1
        else:
            merged.append(right[right_index])
            right_index += 1
    merged.extend(left[left_index:])
    merged.extend(right[right_index:])
    return merged
if __name__ == "__main__":
    data = [38, 27, 43, 3, 9, 82, 10]
    print("Original:", data)
    print("Sorted:", merge_sort(data))
```

OUTPUT:



The screenshot shows the terminal window in VS Code displaying the output of the merge sort function. The command `python ASSIGN-12-5.py` was run, and the terminal shows the original list and the sorted list.

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> python ASSIGN-12-5.py
Original: [38, 27, 43, 3, 9, 82, 10]
Sorted: [3, 9, 10, 27, 38, 43, 82]
```

Explanation: Merge Sort is chosen because it guarantees $O(n \log n)$ time complexity in all cases. It is stable and efficient for large datasets.

Task 2 – Binary Search:

Prompt: Create a Python function `binary_search(arr, target)` that returns the index of the target or -1 if not found. Include best, average, and worst-case complexities in the docstring.

```

28     # TASK 2: Create a Python function binary_search(arr, target) that returns the index of the target or
29     # -1 if not found. Include best, average, and worst-case complexities in the docstring.
30     def binary_search(arr, target):
31         left, right = 0, len(arr) - 1
32         while left <= right:
33             mid = left + (right - left) // 2
34             if arr[mid] == target:
35                 return mid
36             elif arr[mid] < target:
37                 left = mid + 1
38             else:
39                 right = mid - 1
40         return -1
41     if __name__ == "__main__":
42         sorted_data = [1, 3, 5, 7, 9, 11, 13]
43         target = 7
44         result = binary_search(sorted_data, target)
45         if result != -1:
46             print(f"Element {target} found at index: {result}")
47         else:
48             print(f"Element {target} not found in the array.")

```

OUTPUT:

```

PS C:\Users\srarik\OneDrive\Desktop\AI ASSISTED CODING> & c:/users/srarik/appdata/local/python/pythoncore-3.14-64/python.exe "c:/users/srarik/onedrive/desktop/ai assisted coding/ASSIGN-12-5.py"
Element 7 found at index: 3

```

Explanation: Binary Search works on sorted lists and reduces the search space by half each step. It provides O(log n) efficiency, making it faster than linear search.

Task 3 – Healthcare Appointment System:

Prompt: Suggest efficient searching and sorting algorithms for managing appointment records and implement them in Python.

```

49     # TASK 3: Suggest efficient searching and sorting algorithms for managing appointment records and
50     # implement them in Python.
51     class Appointment:
52         def __init__(self, date, time, description):
53             self.date = date
54             self.time = time
55             self.description = description
56         def merge_sort_appointments(appointments):
57             if len(appointments) <= 1:
58                 return appointments
59             mid = len(appointments) // 2
60             left_half = merge_sort_appointments(appointments[:mid])
61             right_half = merge_sort_appointments(appointments[mid:])
62             return merge_appointments(left_half, right_half)
63         def merge_appointments(left, right):
64             merged = []
65             left_index = right_index = 0
66             while left_index < len(left) and right_index < len(right):
67                 if (left[left_index].date, left[left_index].time) < (right[right_index].date, right[right_index].time):
68                     merged.append(left[left_index])
69                     left_index += 1
70                 else:
71                     merged.append(right[right_index])
72                     right_index += 1
73             merged.extend(left[left_index:])
74             merged.extend(right[right_index:])
75             return merged
76         def binary_search_appointments(appointments, target_date):
77             left, right = 0, len(appointments) - 1
78             while left <= right:
79                 mid = left + (right - left) // 2
80                 if appointments[mid].date == target_date:
81                     return mid
82                 elif appointments[mid].date < target_date:
83                     left = mid + 1

```

```

    75 def binary_search_appointments(appointments, target_date):
    76     left, right = 0, len(appointments) - 1
    77     while left <= right:
    78         mid = left + (right - left) // 2
    79         if appointments[mid].date == target_date:
    80             return mid
    81         elif appointments[mid].date < target_date:
    82             left = mid + 1
    83         else:
    84             right = mid - 1
    85     return -1
    86 if __name__ == "__main__":
    87     appointments = [
    88         Appointment("2024-07-01", "10:00", "Doctor's appointment"),
    89         Appointment("2024-07-02", "14:00", "Meeting with client"),
    90         Appointment("2024-07-01", "12:00", "Lunch with friend"),
    91     ]
    92     sorted_appointments = merge_sort_appointments(appointments)
    93     target_date = "2024-07-01"
    94     index = binary_search_appointments(sorted_appointments, target_date)
    95     if index != -1:
    96         print(f"Appointment found: {sorted_appointments[index].description} at {sorted_appointments[index].time}")
    97     else:
    98         print("No appointment found on the target date.")

```

OUTPUT:

```

    86 if __name__ == "__main__":
    87     appointments = [
    88         Appointment("2024-07-01", "10:00", "Doctor's appointment"),
    89         Appointment("2024-07-02", "14:00", "Meeting with client"),
    90         Appointment("2024-07-01", "12:00", "Lunch with friend"),
    91     ]
    92     sorted_appointments = merge_sort_appointments(appointments)
    93     target_date = "2024-07-01"
    94     index = binary_search_appointments(sorted_appointments, target_date)
    95     if index != -1:
    96         print(f"Appointment found: {sorted_appointments[index].description} at {sorted_appointments[index].time}")
    97     else:
    98         print("No appointment found on the target date.")

PS C:\Users\sanik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sanik/AppData/Local/Python/pythoncore-3.14-64/python.exe "C:/Users/sanik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-12-5.py"
Appointment found: Lunch with friend at 12:00

```

Explanation: Binary Search is used for fast appointment ID lookup. Merge Sort is used for stable and efficient sorting by time or consultation fee.

Task 4 – Railway Reservation System:

Prompt: Recommend suitable algorithms to search tickets by ticket ID and sort bookings by travel date or seat number.

```

    100     ## TASK 4: Recommend suitable algorithms to search tickets by ticket ID and sort bookings by travel
    101     # or seat number.
    102     class Ticket:
    103         def __init__(self, ticket_id, travel_date, seat_number):
    104             self.ticket_id = ticket_id
    105             self.travel_date = travel_date
    106             self.seat_number = seat_number
    107         def merge_sort_tickets(tickets, key):
    108             if len(tickets) <= 1:
    109                 return tickets
    110             mid = len(tickets) // 2
    111             left_half = merge_sort_tickets(tickets[:mid], key)
    112             right_half = merge_sort_tickets(tickets[mid:], key)
    113             return merge_tickets(left_half, right_half, key)
    114         def merge_tickets(left, right, key):
    115             merged = []
    116             left_index = right_index = 0
    117             while left_index < len(left) and right_index < len(right):
    118                 if getattr(left[left_index], key) < getattr(right[right_index], key):
    119                     merged.append(left[left_index])
    120                     left_index += 1
    121                 else:
    122                     merged.append(right[right_index])
    123                     right_index += 1
    124             merged.extend(left[left_index:])
    125             merged.extend(right[right_index:])
    126         return merged
    127     def binary_search_tickets(tickets, target_id):
    128         left, right = 0, len(tickets) - 1
    129         while left <= right:
    130             mid = left + (right - left) // 2
    131             if tickets[mid].ticket_id == target_id:
    132                 return mid
    133             elif tickets[mid].ticket_id < target_id:
    134                 left = mid + 1
    135             else:
    136                 right = mid - 1

```

```

File Edit Selection View Go Run Terminal Help < - > Q AI ASSISTED CODING
OPEN EDITORS app.py ASSIGN-12-5.py index.html CHAT + - X
AI ASSISTED CODING
ASSIGNMENTS
Task9_CampusApp
templates
index.html
app.py
Task10_EcommerceApp
templates
index.html
app.py
5-2.html
ASSIGN-1-3.py
ASSIGN-2-2.py
ASSIGN-4-2.py
ASSIGN-5-2.py
ASSIGN-10-2.py
ASSIGN-11-1.py
ASSIGN-12-5.py
DAY-1-2.py
Lab Test1.py
126 def binary_search_tickets(tickets, target_id):
127     if tickets[mid].ticket_id == target_id:
128         return mid
129     elif tickets[mid].ticket_id < target_id:
130         left = mid + 1
131     else:
132         right = mid - 1
133     return -1
134 if __name__ == "__main__":
135     tickets = [
136         Ticket("T001", "2024-08-01", "A1"),
137         Ticket("T002", "2024-08-02", "B1"),
138         Ticket("T003", "2024-08-01", "A2"),
139     ]
140     sorted_tickets_by_date = merge_sort_tickets(tickets, key="travel_date")
141     sorted_tickets_by_seat = merge_sort_tickets(tickets, key="seat_number")
142     target_id = "T002"
143     index = binary_search_tickets(sorted_tickets_by_date, target_id)
144     if index != -1:
145         print(f"Ticket found: {sorted_tickets_by_date[index].ticket_id} for travel date {sorted_tickets_by_date[index].travel_date}")
146     else:
147         print("No ticket found with the target ID.")

```

OUTPUT:

```

File Edit Selection View Go Run Terminal Help < - > Q AI ASSISTED CODING
OPEN EDITORS app.py ASSIGN-12-5.py index.html CHAT + - X
AI ASSISTED CODING
ASSIGNMENTS
Task9_CampusApp
templates
index.html
app.py
Task10_EcommerceApp
templates
index.html
app.py
5-2.html
ASSIGN-1-3.py
ASSIGN-2-2.py
ASSIGN-4-2.py
ASSIGN-5-2.py
ASSIGN-10-2.py
138     tickets = [
139         Ticket("T001", "2024-08-01", "A1"),
140         Ticket("T002", "2024-08-02", "B1"),
141         ...
142     ]
143     PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT TRUFFLE
144     PS C:\Users\srarik\OneDrive\Desktop\AI ASSISTED CODING & C:/Users/srarik/AppData/Local/Python/pythoncore-3.14-64/python
145     .exe C:/Users/Srarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-12-5.py
146     Ticket found: T002 for travel date 2024-08-02
147     PS C:\Users\srarik\OneDrive\Desktop\AI ASSISTED CODING>

```

Explanation: Binary Search ensures fast ticket lookup in sorted records. Merge Sort guarantees consistent $O(n \log n)$ sorting performance.

Task 5 – Hostel Room Allocation:

Prompt: Suggest optimized algorithms to search student allocations by ID and sort records by room number or allocation date.

```

File Edit Selection View Go Run Terminal Help < - > Q AI ASSISTED CODING
OPEN EDITORS app.py ASSIGN-12-5.py index.html CHAT + - X
AI ASSISTED CODING
ASSIGNMENTS
Task9_CampusApp
templates
index.html
app.py
Task10_EcommerceApp
templates
index.html
app.py
5-2.html
ASSIGN-1-3.py
ASSIGN-2-2.py
ASSIGN-4-2.py
ASSIGN-5-2.py
ASSIGN-10-2.py
ASSIGN-11-1.py
ASSIGN-12-5.py
DAY-1-2.py
Lab Test1.py
LAB2(PRACTICE SESSION).pdf
Prime.py
152 ## TASK 5: Suggest optimized algorithms to search student allocations by ID and sort records by room
153     number or allocation date.
154     class StudentAllocation:
155         def __init__(self, student_id, room_number, allocation_date):
156             self.student_id = student_id
157             self.room_number = room_number
158             self.allocation_date = allocation_date
159         def merge_sort_allocations(allocations, key):
160             if len(allocations) <= 1:
161                 return allocations
162             mid = len(allocations) // 2
163             left_half = merge_sort_allocations(allocations[:mid], key)
164             right_half = merge_sort_allocations(allocations[mid:], key)
165             return merge_allocations(left_half, right_half, key)
166         def merge_allocations(left, right, key):
167             merged = []
168             left_index = right_index = 0
169             while left_index < len(left) and right_index < len(right):
170                 if getattr(left[left_index], key) < getattr(right[right_index], key):
171                     merged.append(left[left_index])
172                     left_index += 1
173                 else:
174                     merged.append(right[right_index])
175                     right_index += 1
176             merged.extend(left[left_index:])
177             merged.extend(right[right_index:])
178             return merged
179         def binary_search_allocations(allocations, target_id):
180             left, right = 0, len(allocations) - 1
181             while left <= right:
182                 mid = left + (right - left) // 2
183                 if allocations[mid].student_id == target_id:
184                     return mid
185                 elif allocations[mid].student_id < target_id:
186                     left = mid + 1
187                 else:
188                     right = mid - 1

```

```

    178 def binary_search_allocations(allocations, target_id):
    179     if allocations[0].student_id < target_id:
    180         left = 0
    181     else:
    182         right = len(allocations) - 1
    183     return -1
    184     if __name__ == "__main__":
    185         allocations = [
    186             StudentAllocation("S001", "101", "2024-09-01"),
    187             StudentAllocation("S002", "102", "2024-09-02"),
    188             StudentAllocation("S003", "101", "2024-09-01"),
    189         ]
    190         sorted_allocations_by_room = merge_sort_allocations(allocations, key="room_number")
    191         sorted_allocations_by_date = merge_sort_allocations(allocations, key="allocation_date")
    192         target_id = "S002"
    193         index = binary_search_allocations(sorted_allocations_by_room, target_id)
    194         if index != -1:
    195             print(f"Student allocation found: {sorted_allocations_by_room[index].student_id} in room {sorted_allocations_by_room[index].room_number}")
    196         else:
    197             print("No student allocation found with the target ID.")
    198
    199
    200
    201
    202

```

OUTPUT:

```

    PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "C:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-12-5.py"
    Student allocation found: S002 in room 102
    PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>

```

Explanation: Binary Search enables efficient student ID searching. Merge Sort handles large room allocation records reliably.

Task 6 – Movie Streaming Platform:

Prompt: Recommend searching and sorting algorithms for managing movie records by ID, rating, and release year.

```

    204     ## TASK 6: Recommend searching and sorting algorithms for managing movie records by ID, rating, and
    205     ## release year.
    206     class Movie:
    207         def __init__(self, movie_id, rating, release_year):
    208             self.movie_id = movie_id
    209             self.rating = rating
    210             self.release_year = release_year
    211         def merge_sort_movies(movies, key):
    212             if len(movies) <= 1:
    213                 return movies
    214             mid = len(movies) // 2
    215             left_half = merge_sort_movies(movies[:mid], key)
    216             right_half = merge_sort_movies(movies[mid:], key)
    217             return merge_movies(left_half, right_half, key)
    218         def merge_movies(left, right, key):
    219             merged = []
    220             left_index = right_index = 0
    221             while left_index < len(left) and right_index < len(right):
    222                 if getattr(left[left_index], key) < getattr(right[right_index], key):
    223                     merged.append(left[left_index])
    224                     left_index += 1
    225                 else:
    226                     merged.append(right[right_index])
    227                     right_index += 1
    228             merged.extend(left[left_index:])
    229             merged.extend(right[right_index:])
    230             return merged
    231         def binary_search_movies(movies, target_id):
    232             left, right = 0, len(movies) - 1
    233             while left <= right:
    234                 mid = left + (right - left) // 2
    235                 if movies[mid].movie_id == target_id:
    236                     return mid
    237                 elif movies[mid].movie_id < target_id:
    238                     left = mid + 1
    239                 else:
    240                     right = mid - 1

```

```

File Edit Selection View Go Run Terminal Help ← → Q AI ASSISTED CODING
OPEN EDITORS app.py ASSIGN-12-5.py index.html CHAT + ...
AI ASSISTED CODING .vscode ASSIGNMENTS Task9_CampusApp templates index.html app.py task10_EcommerceApp templates index.html app.py 5-2.html ASSIGN-1-3.py ASSIGN-2-2.py ASSIGN-4-2.py ASSIGN-5-2.py ASSIGN-11-1.py ASSIGN-12-5.py DAY-1-2.py Lab Test1.py SESSIONS
230 def binary_search_movies(movies, target_id):
231     if movies[mid].movie_id == target_id:
232         return mid
233     elif movies[mid].movie_id < target_id:
234         left = mid + 1
235     else:
236         right = mid - 1
237     return -1
238 if __name__ == "__main__":
239     movies = [
240         Movie("M001", 8.5, 2020),
241         Movie("M002", 7.2, 2019),
242         Movie("M003", 9.0, 2021),
243     ]
244     sorted_movies_by_rating = merge_sort_movies(movies, key="rating")
245     sorted_movies_by_year = merge_sort_movies(movies, key="release_year")
246     target_id = "M002"
247     index = binary_search_movies(sorted_movies_by_rating, target_id)
248     if index != -1:
249         print(f"Movie found: {sorted_movies_by_rating[index].movie_id} with rating {sorted_movies_by_rating[index].rating}")
250     else:
251         print("No movie found with the target ID.")
252
253
254

```

OUTPUT:

```

File Edit Selection View Go Run Terminal Help ← → Q AI ASSISTED CODING
OPEN EDITORS app.py ASSIGN-12-5.py index.html CHAT + ...
AI ASSISTED CODING .vscode ASSIGNMENTS Task9_CampusApp templates index.html app.py task10_EcommerceApp templates index.html app.py 5-2.html ASSIGN-1-3.py ASSIGN-2-2.py ASSIGN-4-2.py ASSIGN-5-2.py ASSIGN-11-1.py ASSIGN-12-5.py DAY-1-2.py Lab Test1.py SESSIONS
243 Movie("M001", 8.5, 2020),
244 Movie("M002", 7.2, 2019),
245 Movie("M003", 9.0, 2021)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT TRUFFLE
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python
.exe "C:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-12-5.py"
● No movie found with the target ID.
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING

```

Explanation: Binary Search quickly locates movies by ID. Merge Sort efficiently sorts movies by rating or release year.

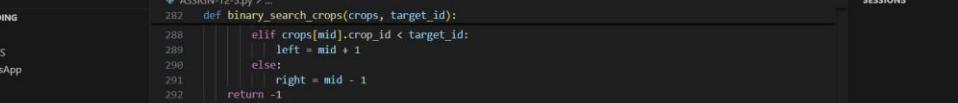
Task 7 – Smart Agriculture System:

Prompt: Suggest appropriate algorithms to search crop details by ID and sort crops by moisture level or yield estimate.

```

File Edit Selection View Go Run Terminal Help ← → Q AI ASSISTED CODING
OPEN EDITORS app.py ASSIGN-12-5.py index.html CHAT + ...
AI ASSISTED CODING .vscode ASSIGNMENTS Task9_CampusApp templates index.html app.py task10_EcommerceApp templates index.html app.py 5-2.html ASSIGN-1-3.py ASSIGN-2-2.py ASSIGN-4-2.py ASSIGN-5-2.py ASSIGN-11-1.py ASSIGN-12-5.py DAY-1-2.py Lab Test1.py LAB(2)(PRACTICE SESSION).pdf Prime.py SESSIONS
256 ## TASK 7: Suggest appropriate algorithms to search crop details by ID and sort crops by moisture level or yield estimate.
257 class Crop:
258     def __init__(self, crop_id, moisture_level, yield_estimate):
259         self.crop_id = crop_id
260         self.moisture_level = moisture_level
261         self.yield_estimate = yield_estimate
262     def merge_sort_crops(crops, key):
263         if len(crops) <= 1:
264             return crops
265         mid = len(crops) // 2
266         left_half = merge_sort_crops(crops[:mid], key)
267         right_half = merge_sort_crops(crops[mid:], key)
268         return merge_crops(left_half, right_half, key)
269     def merge_crops(left, right, key):
270         merged = []
271         left_index = right_index = 0
272         while left_index < len(left) and right_index < len(right):
273             if getattr(left[left_index], key) < getattr(right[right_index], key):
274                 merged.append(left[left_index])
275                 left_index += 1
276             else:
277                 merged.append(right[right_index])
278                 right_index += 1
279         merged.extend(left[left_index:])
280         merged.extend(right[right_index:])
281         return merged
282     def binary_search_crops(crops, target_id):
283         left, right = 0, len(crops) - 1
284         while left <= right:
285             mid = left + (right - left) // 2
286             if crops[mid].crop_id == target_id:
287                 return mid
288             elif crops[mid].crop_id < target_id:
289                 left = mid + 1
290             else:
291                 right = mid - 1

```



The screenshot shows a code editor interface with multiple tabs open. The main tab contains Python code for a binary search algorithm and crop management. The code defines a function `binary_search_crops` that takes a list of crops and a target ID. It uses a standard binary search approach to find the crop with the matching ID. If found, it prints the crop's moisture level; if not found, it prints a message indicating no crop was found with the target ID. The code also includes imports for `merge_sort_crops` from `crops` and `ASSIGN-12-5.py`.

```
def binary_search_crops(crops, target_id):
    left = 0
    right = len(crops) - 1

    while left <= right:
        mid = (left + right) // 2

        if crops[mid].crop_id == target_id:
            return crops[mid]
        elif crops[mid].crop_id < target_id:
            left = mid + 1
        else:
            right = mid - 1

    return None

if __name__ == "__main__":
    crops = [
        Crop("C001", 20.5, 1000),
        Crop("C002", 15.0, 800),
        Crop("C003", 25.0, 1200),
    ]

    sorted_crops_by_moisture = merge_sort_crops(crops, key="moisture_level")
    sorted_crops_by_yield = merge_sort_crops(crops, key="yield_estimate")
    target_id = "C002"
    index = binary_search_crops(sorted_crops_by_moisture, target_id)

    if index != -1:
        print(f"Crop found: {sorted_crops_by_moisture[index].crop_id} with moisture level {sorted_crops_by_moisture[index].moisture_level}")
    else:
        print("No crop found with the target ID.")
```

OUTPUT:

Explanation: Binary Search improves search efficiency in sorted crop data. Merge Sort ensures accurate sorting of agricultural metrics.

Task 8 – Airport Flight Management:

Prompt: Recommend algorithms to search flight records by flight ID and sort flights by departure or arrival time.

The screenshot shows a code editor interface with the following details:

- File Explorer:** On the left, it lists several projects and files. Projects include "OPEN EDITORS", "AI ASSISTED CODING", "ASSIGNMENTS", "Task9_CampusApp" (containing "templates", "index.html", "app.py"), and "Task10_EcommerceApp" (containing "templates", "index.html", "app.py"). Other files listed are "ASSIGN-1-3.py", "ASSIGN-2-2.py", "ASSIGN-4-2.py", "ASSIGN-5-2.py", "ASSIGN-10-2.py", "ASSIGN-11-1.py", "ASSIGN-12-5.py" (which is currently selected), "DAY-1-2.py", "Lab Test1.py", "LAB(2)PRACTICE SESSION.pdf", and "Prime.py".
- Code Editor:** The main area displays Python code for flight record search algorithms. The code includes functions for sorting flights by departure or arrival time, merging sorted flights, and performing a binary search on a list of flights to find a specific flight ID.
- Terminal:** At the bottom, there is a terminal window showing the command "BLKBOX Agent Open Website".
- Status Bar:** The status bar at the bottom indicates the file is "Ln 358, Col 53" with "Spaces: 2", "UTF-8", "CRLF", and "Python" as the language. It also shows the current time as "3:14.2" and a "Go Live" button. A "BLACKBOX" logo is present.

The screenshot shows a code editor interface with the following details:

- File Explorer:** On the left, it lists several projects and files:
 - OPEN EDITORS:** app.py
 - AI ASSISTED CODING:** ASSIGN-12-5.py
 - ASSIGNMENTS:** task9_CampusApp
 - Task10_ECommerceApp:** templates/index.html, app.py
 - 5-2.html**
 - ASSIGN-1-3.py**
 - ASSIGN-2-2.py**
 - ASSIGN-4-2.py**
 - ASSIGN-5-2.py**
 - ASSIGN-10-2.py**
 - ASSIGN-11-1.py**
 - ASSIGN-12-5.py**
- Code Editor:** The main area displays Python code for a binary search algorithm on a list of flights. The code includes imports for os, sys, and merge_sort_flights. It defines a function binary_search_flights that takes a list of flights and a target ID. The function uses a binary search to find the target ID. If found, it prints the flight information; if not found, it prints a message indicating no flight was found.
- Status Bar:** At the bottom, there are status icons for file operations like save, close, and refresh, as well as a progress bar.

OUTPUT:



The screenshot shows a code editor with several tabs open. The active tab contains Python code for merge sort:

```
318 left_half = merge_sort_flights(flights[:mid], key)
319 right_half = merge_sort_flights(flights[mid:], key)
```

Below the code editor is a terminal window showing the following output:

```
PS C:\Users\Sanik\OneDrive\Desktop\VAI ASSISTED CODING & C:\Users\sanik\AppData\Local\Python\pythoncore-3.14-64\python
  .\exe "C:/Users/Sanik/OneDrive/Desktop/VAI ASSISTED CODING/ASSIGN-12-5.py"
Flight found: F002 departing at 2024-10-01 09:00
PS C:\Users\Sanik\OneDrive\Desktop\VAI ASSISTED CODING
```

Explanation: Binary Search provides fast flight ID lookup. Merge Sort ensures stable sorting of time-based flight records.