

Project:Cultural Heritage Site Management System

Course: Data Structure Curriculum Design

Student Name: Sheng Jie Zhang

Shuang Wu

Jia Yi Yang

Wan Qing Cheng

Zan Hui Long

Student Number: 2022310913

2022310909

2022310900

2022311305

2022311472

Faculty Adviser： Lilia SFAXI Wei Wei

**Abstract**

This project aims to develop a system to manage and preserve cultural heritage sites, focusing on the efficient organization of artifacts, structures within the site, and managing visitor tours. The system will use trees for hierarchical organization of artifacts and queues for for managing visitor tours.

First we define the data structure. The representation of nodes in the workpiece tree and the representation of the hierarchical structure of the workpiece using the tree structure are completed. We also developed the ability to add, remove or modify nodes in the artifact tree. Implements the search and retrieval of all artifacts of a specific type and importance in the tree. And passed all the tests. Finally, a function to load the data was added to the manager code.

The second part. We designed a Python class to manage the queue order of tour groups. Support for adding, moving out, viewing the next and rescheduling tour groups. It can sort queues based on arrival time and reservation priority. We also wrote a program to invoke the data set. This allows us to re-prioritize queues based on visitor arrival times. Make sure we're ready for any change.

After all that. We further optimized the program. We base this on more complex data sets. Group visitors more carefully according to their intention to visit. Put them in a more human order. In order to optimize the tourist experience and the role of heritage management. Finally, we developed a graphical interface to display the object tree and visitor queue. Administrators can use this interface to add or remove artifacts and visitors.

**Catalogue**

**[I Description Of Project 5](#_Toc31319)**

**[II Requirement Analysis of Project 5](#_Toc16367)**

**[2.1Project context 5](#_Toc645)**

**[2.2Project purpose 5](#_Toc10838)**

**[III Design of Project 5](#_Toc10008)**

**[3.1. Artifact\_tree 5](#_Toc19300)**

**[3.2 Manager 6](#_Toc26932)**

**[3.3 Queue 7](#_Toc8281)**

**[3.4. Visitors 7](#_Toc30497)**

**[IV Implementation of Project 8](#_Toc13393)**

**[4.1 Defining the Data Structures 8](#_Toc21911)**

**[4.2 Implementing Tree Utilities 8](#_Toc20210)**

**[4.3 Loading the Dataset 8](#_Toc21576)**

**[4.4 Tour Queue Management 8](#_Toc5440)**

**[4.5 Implementing Queue Utilities 8](#_Toc12252)**

**[4.6 Loading the Dataset 9](#_Toc5676)**

**[4.7 Prioritized Tour Scheduling Based on Visitor 9](#_Toc5314)**

**[4.8 Graphical Interface 9](#_Toc18597)**

**[V Running and Debugging of Project 9](#_Toc4472)**

**[VI Summary 10](#_Toc28472)**

**[6.1 Daily Work Schedul 10](#_Toc15427)**

**[6.2 Division of Labor within the Group 10](#_Toc13681)**

**[6.3 Tips 10](#_Toc12136)**

**[6.4 Difficulties and Solution of Problems 12](#_Toc15785)**

**I Description Of Project**

This project aims to develop a system to manage and preserve cultural heritage sites, focusing on the efficient organization of artifacts, structures within the site, and managing visitor tours.This system can prioritize visitors according to their time and time, and then prioritize the museum exhibits. By helping the museum to better manage it.

**II Requirement Analysis of Project**

**2.1Project context**

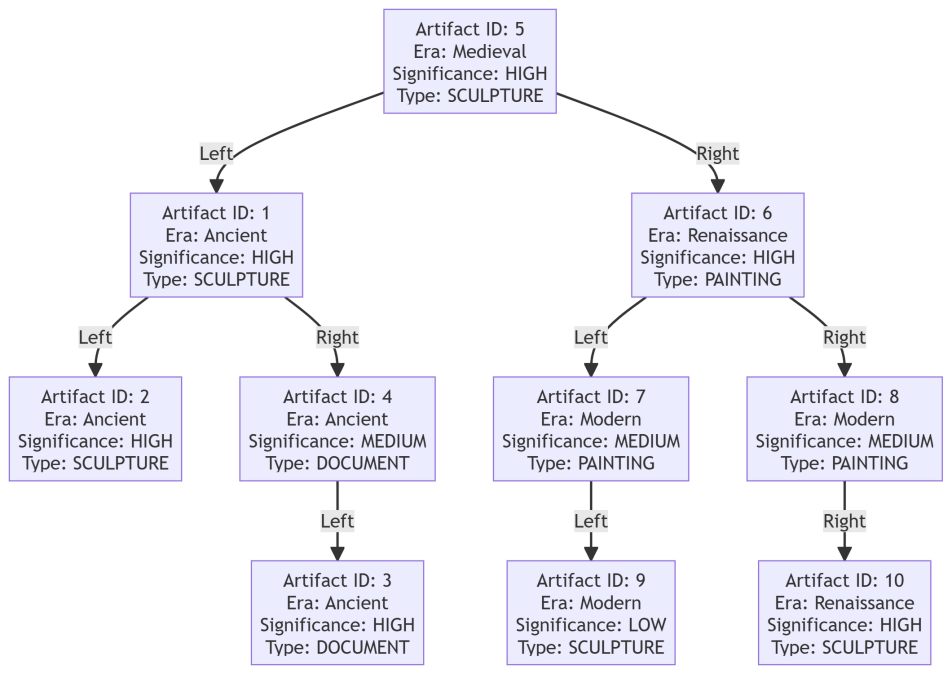
This project is used with the museum to better manage the visitors and protect the exhibits. In the case of tourists making reservations, the museum application system conducts data analysis and management.so that it can arrange the visiting order of tourists in an orderly manner, avoid or reduce congestion and reduce visiting time. It can also improve the quality of the visits.

**2.2Project purpose**

The purpose of this project is to conduct a series of analysis of the exhibits, and then allow the visitors to make an appointment with questions before the visit, to determine the information of the visitors, which is the data we need. First, let's prioritize visitors according to their time and ticket levels.Then, with the previous priority situation, we prioritize the era pavilions that they want to visit or are interested in. This can avoid crowding.Finally, because tourists can't make an appointment at the same time, we need to upgrade the system. Once there is a new appointment or an emergency, the system will be reordered to achieve the most effect.

**III Design of Project**

**3.1. Artifact\_tree**



Artifact\_tree

1. Binary Search Tree, BST:

The 'ArtifactTree' class implements a binary search tree for storing and managing artifacts of type Artifact.

The 'ArtifactTreeNode' class represents nodes in the tree, each containing a heritage object and left and right child node Pointers.

2. Insertion algorithm:

The '\_insert\_node' method is used to insert a new node into the binary search tree. According to the 'artifact\_id' of the node, it is decided to insert the node into the left or right subtree of the current node to ensure the order of the tree.

3. Delete algorithm:

The '\_remove\_node' method is used to remove nodes that have the specified 'artifact\_id'. According to the value of the 'artifact\_id' comparison node, it is divided into three cases: the node has no child nodes, only one child node, and two child nodes. When there are two child nodes, find the smallest node in the right subtree and replace it.

4. Minimum value finding algorithm:

The '\_find\_min' method is used to find the smallest node in the given stator tree. Start at the current node and work your way down the left subtree until you find the smallest node.

5. Search algorithm:

The '\_search\_helper' method is used to search artifacts by type and importance. Iterate through the tree recursively, adding eligible artifacts to the 'found\_artifacts' list.

6. Display algorithm:

The 'display' method is used to display all artifacts in the form of a tree. By recursively traversing the tree from right to left, the artifacts information of each node is printed in the way of hierarchical indentation.

**3.2 Manager**

1. Data loading and processing:

Use the pandas library to load and process heritage information and visitor information in CSV files.

The 'load\_artifacts' method and the' load\_visitors' method load the Artifact and visitor data, respectively, and convert them into 'artifact' and 'VisitorGroup' objects. And store them in self.artifacts and self.visitor\_queue.

2. Data storage and management:

Use lists (' self.artifacts' and 'self.visitor\_queue') to store the Artifact and the VisitorGroup objects.

Implements methods for adding, removing, and rearranging groups of visitors that use the basic manipulation and sorting capabilities of lists.

3. Data presentation and sorting:

The 'display\_artifacts' method and the' display\_visitor\_queue 'method print the loaded artifacts and visitor information, respectively. They use the 'sorted' function and the 'lambda' expression to sort groups of artifacts and visitors to ensure that they are displayed in the specified attribute order.

4. Data query:

The 'search\_artifacts\_by\_era' method queries artifacts according to the specified era. It uses a list deduction to select eligible artifacts from self.artifacts.

5. Data storage:

The 'save\_to\_csv' method saves the data to a CSV file. It uses' pandas' DataFrame 'and' to\_csv 'methods to write data to the specified file.

6. Exception Handling:

In the 'VisitorGroup' class, the 'try-except' statement is used to handle the parsing error of the arrival time, ensuring that the arrival time strings in different formats can be handled.

**3.3 Queue**

1. 'add\_group' :

Add visitor group objects to the end of the queue and sort them according to arrival time and booking priority.

2. 'remove\_group' :

Removes the first visitor group object from the queue.

3. 'peek\_next\_group' :

Returns the first visitor group object in the queue, but does not remove it.

4. 'reschedule\_group' :

Based on the given group ID, update the arrival time and booking priority for the corresponding tourist group and reorder the queue.

**3.4. Visitors**

1. 'Significance' :

Three importance levels are defined using 'enum.Enum' : 'HIGH', 'MEDIUM', 'LOW', corresponding to integer values 1, 2, and 3 respectively.

2. VisitorGroup:

Constructor: initializes the 'VisitorGroup' object, taking the four parameters' group\_id ', 'arrival\_time', 'reservation\_priority' and 'preference'.

The reservation\_priority parameter can be of the Significance enumerated type, orit can be the corresponding character, The string is mapped to enumeration values by 'Significance[reservation\_priority.strip().upper()]'.

String representation: Returns a string representation of the VisitorGroup object, including group\_id, arrival\_time, reservation\_priority, and preference.

Equality comparison: Implements equality comparison of objects, returns' True 'when the group\_id of two' VisitorGroup 'objects is the same, returns' False' otherwise.

**IV Implementation of Project**

**4.1 Defining the Data Structures**

An Artifact class is defined to represent an artifact, each with a unique artifact id, name, type, and significance.

Define a Visitor class to represent a visitor, each with a unique visitor id, name, and priority.

Use the ArtifactTreeNode class to represent nodes in a binary search tree, each containing an Artifact instance and left and right child node Pointers.

The queue is represented by the VisitorQueue class, in which the Visitor instance is stored, implemented using the deque structure.

**4.2 Implementing Tree Utilities**

Insert algorithm: In the ArtifactTree class, implement the add artifact method for inserting a new artifact node into a binary search tree. Compare the artifact id of the new artifact with the artifact id of the current node to find the insertion location recursively.

Delete algorithm: In the ArtifactTree class, implement the remove artifact method to remove nodes from the tree based on the artifact id. Handle three situations: a node has no child nodes, has one child node, and has two child nodes.

Minimum search algorithm: In the ArtifactTree class, the find min method is implemented to find the smallest node in a subtree and traverse along the left subtree until the leftmost node is found.

Traversal algorithm: In the ArtifactTree class, implement the search helper method to search artifacts by type and importance, recursively traverse all nodes of the tree, and collect artifacts that match the criteria.

**4.3 Loading the Dataset**

Write data loading functions to read artifacts and visitor data from external data sources such as files or databases.

Parse the data and create the appropriate Artifact and Visitor instances.

Add the Artifact instance to the ArtifactTree.

Add the Visitor instance to the VisitorQueue.

**4.4 Tour Queue Management**

Implement the add visitor and remove visitor methods in the VisitorQueue class to add and remove visitors, respectively.

The append method of the deque data structure is used to add visitors to the end of the queue and the popleft method is used to remove visitors from the head of the queue.

Implement the display queue method to display all visitors in the current queue.

**4.5 Implementing Queue Utilities**

Implement a priority queue algorithm and add a way to manage visitors based on priority to the VisitorQueue class.

The heapq module is used to implement the priority queue, the heappush method is used to add tourists to the queue by priority, and the heappop method is used to remove tourists by priority.

**4.6 Loading the Dataset**

Similar to Task 3, write the data load function again to read the visitor data from an external data source.

Parses the data and creates the corresponding Visitor instance, adding it to the priority queue.

**4.7 Prioritized Tour Scheduling Based on Visitor**

To implement the scheduling algorithm based on the priority of tourists, add the method of scheduling tourists by priority to the VisitorQueue class.

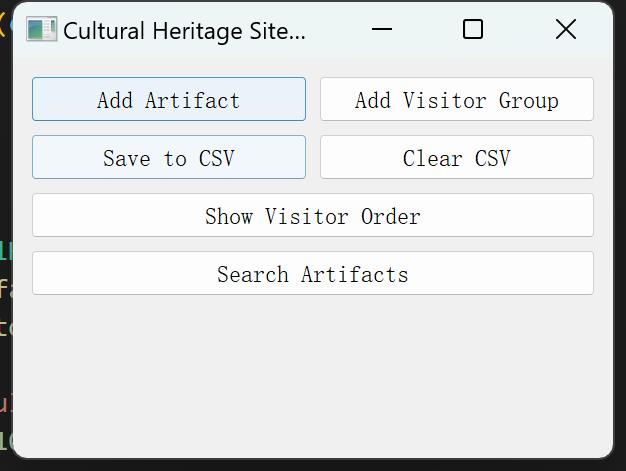
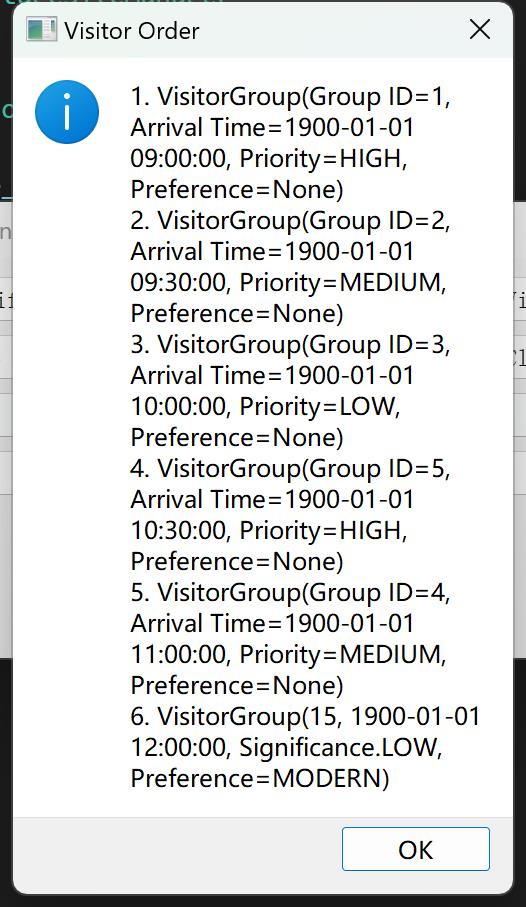
According to the priority of visitors, arrange the order of visits to ensure that high priority visitors get priority services.

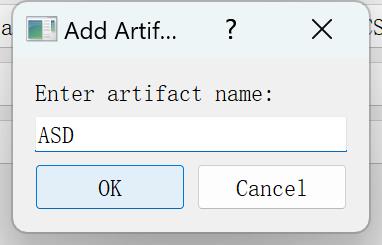
**4.8 Graphical Interface**

Create user interfaces using Graphical User interface (GUI) libraries.

Design the interface layout, including the function module of heritage management and visitor management.

The implementation of event processing algorithm, processing user operations display cultural relic tree structure and tourist queue.

 GUI



Add Artifact Visitor Order

**V Running and Debugging of Project**

1. Ensure that the Artifact and Visitor classes are defined to properly store and represent the artifact and visitor information.

2. Ensure that the ArtifactTreeNode and VisitorQueue classes properly manage the tree structure and queue operations for artifacts and visitors.

3. We need to check for logical errors in the code, especially in insert, delete, and search operations.

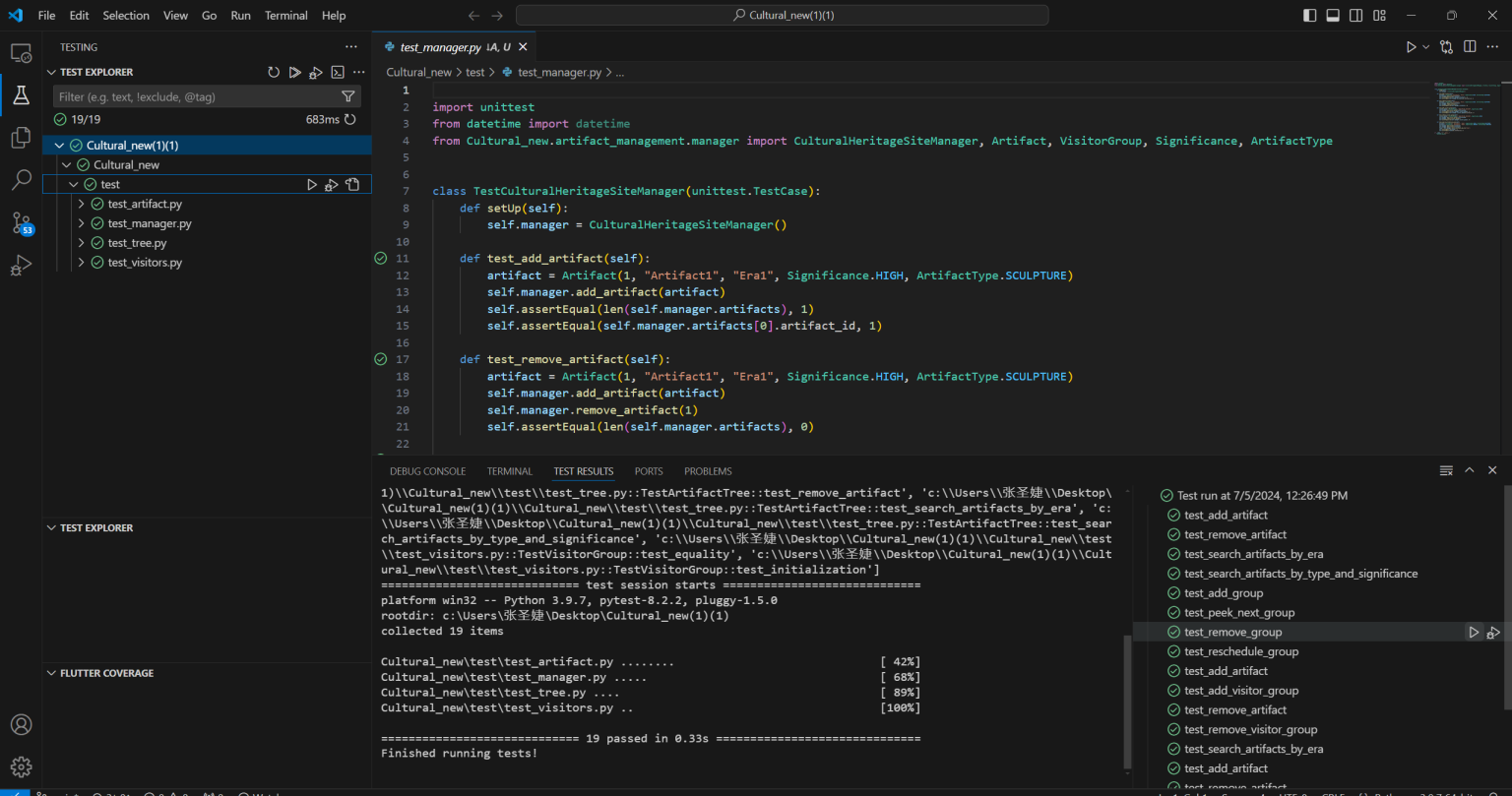
4. When inserting new nodes, deleting nodes, and searching nodes, ensure that the binary search tree property is maintained, that is, the left subtree is smaller than the root node, and the right subtree is larger than the root node.

5. Ensure that queue addition and deletion are performed in the order of FIFO.

6. When using the deque, make sure to add and remove visitors correctly using the append and popleft methods.

7. When implementing priority queues, use heapq to ensure proper management and scheduling according to visitor priorities.

8. When the GUI is involved, ensure that the layout of the interface is displayed correctly and that user actions trigger event handling correctly.



Test

**VI Summary**

**6.1 Daily Work Schedul**

6.24~6.26We finished the artifact-tree,the artifact,the visitors.

6.27~6.28We finished queue,manager.

7.1~7,3Improve the management part and Gui.

7.4We finished the test ,the coverage ,the githug and the report.

**6.2 Division of Labor within the Group**

Jiayi yang:finished the Gui ，integrate code ,upload githug，finished the artifact tree.

Shengjie zhang:finished the artifact ,the manager and the test coverage，Word typesetting.

Shuang wu:finsined the visitors and the report.

Wanqing chen:finished the test and the ppt.

Zhanhui long:finished the repot ,the queue and the test.

**6.3 Tips**

Shuang Wu : Learning binary trees and queues is not only to master the two basic data structures, but also to have a deep understanding of the core idea of "data structure + algorithm" in computer science. These two data structures have their own characteristics and are correlation, which together constitute a rich and colorful data organization and processing methods in computer science. By learning them, I not only improved my programming ability, but also learned how to choose the right data structure and algorithm to solve the problem according to the actual needs of the problem. The cultivation of this ability will have a profound impact on the future study and work.Another I know how to use the VScode,I learned to use a new programming tool.I think it is cool.

Jiayi yang: After completing the project of the algorithm analysis and design course, I felt very fulfilled. Through this project, I not only deepened my understanding of the data structure and algorithms, but also improved my programming ability and the ability to solve practical problems. In the project, we implemented a heritage management system, which uses a binary search tree to store and manage heritage information. The ArtifactTreeNode and ArtifactTree classes I designed not only efficiently add and delete artifacts, but also search by type, importance, or age. Especially the deletion operations, I took an optimized approach to preserve the binary search tree properties by finding the smallest node in the right subtree. In addition, I have implemented a graphical user interface (GUI), using the PyQt 5 library, to enable users to easily interact with the system. Through the GUI, users can add artifacts, visitor groups, save data to the CSV files, clear specific records in the CSV files, display the access order of the visitors, and search for artifacts according to the era. Throughout the development of the project, I learned how to apply theoretical knowledge into practice, how to design and implement complex systems, and how to collaborate with team members. I also realized the importance of readability and maintainability of the code, so when writing the code, I focused on the clarity and structure of the code. Finally, I am very grateful to my team members and teachers for their help and support as the key to our completion of this project.

Shengjie zhang：I learned a lot from this experiment. Write the Artifact module, where the Significance enumeration defines the importance level of the artifact, and the ArtifactType enumeration defines the type of artifact. We are able to easily create, manage, and compare artifact objects. The Manager module is written to manage the queue of artifacts and visitors. In this module, I realized the functions of loading, displaying, adding, and deleting cultural relics and tourists.I have gained a deeper understanding of module management in Python and learned how to deal with module imports in complex projects. During development and testing, I encountered an import error between modules. It is not uncommon for a package or module to be found. To solve this problem, I used the following code to dynamically add the project root to Python's search path, ensuring that when called, the Python interpreter correctly finds the required modules and packages. This approach, while simple, was very effective and solved a lot of the import problems I had during development.There were also some deficiencies in the course of the experiment. The project structure was not planned in detail in the early stages of development, resulting in frequent adjustments to the dependencies between modules in the later stages. In the future, when carrying out similar projects, I will pay more attention to the early planning and design to avoid unnecessary trouble.Overall, this experiment not only helped me improve my programming skills, but also taught me how to solve problems in real projects. These experiences and lessons will have a profound impact on my future studies and work. Finally, I would like to thank my teachers and teammates for their help.

Zanhui Long：After completing this course project, I felt very enriched and learned useful knowledge. Developing a graphical user interface involves knowledge of front-end development. User interaction and interface design need to be considered. After completing the improvement of the user's visiting experience. I felt a great sense of accomplishment. These are the feelings that need to experience the writing and debugging of the program themselves. In integrating the system. We need to consider the interaction and data flow between the various modules. It's a complicated process. Details such as data consistency and interface design are involved. Completing these will be of great help to my ability improvement. It also has a profound influence on the future study. At last. The group format allows us to learn from each other's strengths. Through discussion and sharing. It allows us to find and solve problems efficiently. Improve the quality of the overall project. Thank you very much for the opportunity provided by the school and the help from Lilia.

Wanqing Cheng：In the field of data structures, binary trees and queues are fundamental and important parts. Through my studies, I learned that binary trees are a very efficient way to store and retrieve data, and can quickly find, insert, and delete data. At the same time, as a special linear structure, the queue follows the first-in, first-out principle, which plays a crucial role in the implementation of algorithms, such as storing nodes to be accessed in breadth-first searches. Combining the two, a variety of algorithms can be efficiently implemented to solve practical problems. Mastering their use can go a long way in improving your programming skills.

**6.4 Difficulties and Solution of Problems**

No query before gui, cannot save The whole problem is that the cross-folder call reporting error cannot be found