Do You Know?

Set 10

The source code for the AbstractGrid class is in Appendix D.

1. Where is the isValid method specified? Which classes provide an implementation of this method?

Answer: The method is specified in class Grid and implemented in class BoundedeGrid and class UnboundedGrid.

2. Which AbstractGrid methods call the isValid method? Why don't the other methods need to call it?

Answer: The method getValidAdjacentLocations. The other methods do not need to call isValid method directly but call indirectly. They all call method getValidAdjacentLocations which calls method isValid to ensure the location is valid.

3. Which methods of the Grid interface are called in the getNeighbors method? Which classes provide implementations of these methods?

Answer: Method getOccupiedAdjacentLocations and method get. The class AbstractGrid implementes the method getOccupiedAdjacentLocations and the method get is implemented by class BoundGrid and class UnboundGrid.

4. Why must the get method, which returns an object of type E, be used in the getEmptyAdjacentLocations method when this method returns locations, not objects of type E?

Answer: We use the get method and return the object in neighborLoc to ensure whether the location is empty or not. If the returned object is null, the location can be added to the list. Else the location cannot be added to the list.

5. What would be the effect of replacing the constant Location.HALF_RIGHT with Location.RIGHT in the two places where it occurs in the getValidAdjacentLocations method?

Answer: It will only test locations in North, South, East and West. However, other location will not be returned even they are valid. The number of possible valid adjacent locations would become four from eight.

Do You Know?

Set 11

The source code for the BoundedGrid class is in Appendix D.

1. What ensures that a grid has at least one valid location?

Answer: The constructor of class BoundedGrid which will throw IllegalArgumentException if the grid has none valid location.

2. How is the number of columns in the grid determined by the getNumCols method? What assumption about the grid makes this possible?

Answer: return occupantArray[0].length;

According to the constructor precondition, numRows() > 0, so the occupantArray[0] is non-null.

3. What are the requirements for a Location to be valid in a BoundedGrid?

In the next four questions, let r = number of rows, c = number of columns, and n = number of occupied locations.

Answer: The value of location's row greater than or equal to 0 and less than the total number of BoundedGrid's rows. The value of location's column greater than or equal to 0 and less than the total number of BoundedGrid's columns.

4. What type is returned by the getOccupiedLocations method? What is the time complexity (Big-Oh) for this method?

Answer: ArrayList<Location>. The time complexity is O(r*c).

5. What type is returned by the get method? What parameter is needed? What is the time complexity (Big-Oh) for this method?

Answer: Type E. It needs parameter Location. The time complexity is O(1).

6. What conditions may cause an exception to be thrown by the put method? What is the time complexity (Big-Oh) for this method?

Answer: The invalid conditions or the object is null will cause an exception. The time complexity is O(1).

7. What type is returned by the remove method? What happens when an attempt is made to remove an item from an empty location? What is the time complexity (Big-Oh) for this method?

Answer: Type E. Null is return and it will not cause error or exception. The time complexity is O(1).

8. Based on the answers to questions 4, 5, 6, and 7, would you consider this an efficient implementation? Justify your answer.

Answer: Yes. Because only the method getOccupiedLocations is inefficient with its time complexity is O(r*c), and other method like put, get and remove is efficient with their time complexity all are O(1).

Do You Know?

Set 12

The source code for the UnboundedGrid class is in Appendix D.

1. Which method must the Location class implement so that an instance of HashMap can be used for the map? What would be required of the Location class if a TreeMap were used instead? Does Location satisfy these requirements?

Answer: The method hashCode and equals. If a TreeMap were used, the Location class should also implement Comparable and the method compareTo must be implemented.

Yes, class Location does satisfy these requirements.

2. Why are the checks for null included in the get, put, and remove methods? Why are no such checks included in the corresponding methods for the BoundedGrid?

Answer: Because the key of HashMap is null if not allowed but the class UnboundedGrid uses HashMap which acquiesce that all value of HashMap is valid and the method isValid always return true which do not consider null.

The class BoundedeGrid does not need because its method isValid ensure the location is not null.

3. What is the average time complexity (Big-Oh) for the three methods: get, put, and remove? What would it be if a TreeMap were used instead of a HashMap?

Answer: All time complexity of method get, put and remove is O(1). If a TreeMap were used, all the time complexity would be O(log N).

4. How would the behavior of this class differ, aside from time complexity, if a TreeMap were used instead of a HashMap?

Answer: Because TreeMap is inorder traversal with storing its key in a balanced binary search tree, we may get the occupants list of different order with calling method getOccupiedLocations.

5. Could a map implementation be used for a bounded grid? What advantage, if any, would the two-dimensional array implementation that is used by the BoundedGrid class have over a map implementation?

Answer: Yes, it could.

Two-dimensional is better than map if the bounded grid are almost full. One of the reasons is that may stores both the item and location but two-dimensional just stores items so that map use more memory. What's more, two-dimensional is easier to implement and preserve.

Exercises

1. Implement the methods specified by the Grid interface using this data structure. Why is this a more time-efficient implementation than BoundedGrid?

Answer: Obviously, it will save a lot of space if the bounded grid is large and

the number of items is small. The time complexity of method getOccupiedLocations for this implementation is O(r+n), where the r is the number of rows and n is the number of items in the grid rather than O(r*c) where r is the same as O(r*n) and c is the number of columns in this grid.

2. Consider using a HashMap or TreeMap to implement the SparseBoundedGrid. How could you use the UnboundedGrid class to accomplish this task? Which methods of UnboundedGrid could be used without change?

Answer: Some methods of class UnboundedGrid can be used directly.The method getOccupiedLocations, get, put and remove in UnboundedGrid do not need to change.

Fill in the following chart to compare the expected Big-Oh efficiencies for each implementation of the SparseBoundedGrid.

Let r = number of rows, c = number of columns, and n = number of occupied locations

Methods	SparseGridNode version	LinkedList<OccupantInCol> version	HashMap version	TreeMap version
getNeighbors	O(c)	O(c)	O(1)	O(log N)
getEmptyAdjacentLocations	O(c)	O(c)	O(1)	O(log N)
getOccupiedAdjacentLocations	O(c)	O(c)	O(1)	O(log N)
getOccupiedLocations	O(r+n)	O(r+n)	O(n)	O(n)
get	O(c)	O(c)	O(1)	O(log N)
put	O(c)	O(c)	O(1)	O(log N)
remove	O(c)	O(c)	O(1)	O(log N)

3.What is the Big-Oh efficiency of the get method? What is the efficiency of the put method when the row and column index values are within the current array bounds? What is the efficiency when the array needs to be resized?

Answer: The Big-Oh efficiency of the get method is O(1).

The efficiency of the put method is O(1) within current array bounds. The efficiency when the array needs to be resized is O(dim * dim).