23ed Aug, 2015

2015 年软件工程实训 Grid World - Part 5

Name：陈炜健　**Student ID**：**13331018**

Do You Know? - Set 10

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

Answer: The invalid method specified in the Grid class and implemented in the BoundedGrid class and the UnboundedGrid class.

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

Answer: The getValidAdjacentLocations method. Because the other methods call the getValidAdjacentLocations method to get valid location instead of calling the isValid method directly.

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

Answer: The getOccupiedAdjacentLocations method which is implemented in the abstract class, and the get method which is implemented in the BoundedGrid class and the UnboundedGrid class.

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: The getEmptyAdjacentLocations method needs to use the get method to judge whether the adjacent location is empty or not.

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: In this case, the method will only check the ahead, right, left, behind location of the checked location, which means that it will miss checking other location.

Do You Know? - Set 11

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

Answer: The constructor of the BoundedGrid class will throw a IllegalArgumentException error if the row or the column is less than 0, which means that the grid will always have at least one 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: The getNumCols returns the length of the occupantArray[0]. The assumption that the grid has at least one valid location makes this possible.

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

Answer: The row number must be between 0 and the grid row number and the column number must be between 0 and the grid column number.

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

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

Answer: ArrayList<Location>. The complexity(Big-Oh) 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: The E type is returned and a Location parameter is needed. The complexity(Big-Oh) 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: A IllegalArgumentException will be thrown when the location is not valid. A NullPointerException will be thrown when the object is empty. The complexity(Big-Oh) 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: A E type is returned. Null will be returned if an attempt is mae to remove an item from an empty location. The complexity(Big-Oh) 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. It uses a two-dimensional array to store the data of the object, which makes the complexity of lots of operations be a low level.

Do You Know? - Set 12

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 equal method and the hashCode method.
- If a TreeMap were used instead, a comparable operation is needed so the compareTo method of the Location is required.
- Yes. Location 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 it is no need to waste space to store a empty location in the map. In the BoundedGrid, each Location has already been allocated a space for itself, so there are no such checks.

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: O(1). If a TreeMap were used, O(lnN).

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

Answer: A TreeMap is sorted, which means that comparable operations will become convenient, but the normal operation like get, put, remove operations will have a higher complexity.

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, a map implementation can be used for a bounded grid. However, using the two-dimensional array implementation will do better than map implementation in some ways like creating, accessing and deleting the datas quickly because it use a fixed space to store the datas.

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: Because this way will save lots of space if the program requires a very large bounded grid but contains very few objects.

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:

- Using the hashmap which is used in the UnboudedGrid class.
- The getOccupiedLocations method could be used without 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

Answer:

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(c+n)	O(c+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. Consider an implementation of an unbounded grid in which all valid locations have non-negative row and column values. The constructor allocates a 16 x 16 array. When a call is made to the put method with a row or column index that is outside the current

array bounds, double both array bounds until they are large enough, construct a new square array with those bounds, and place the existing occupants into the new array.

Implement the methods specified by the Grid interface using this data structure. 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 when the row and column index values are within the current array bounds is O(1).
- The efficiency is O(newsize * newsize) when the array needs to be resized.