13331183
卢建业

Set 10

1. Where is the isValid method specified? Which classes provide an implementation of this method?
   The isValid methoid is specified in the Grid interface, The BoundedCrid and UnboundedGrid class provide an implementation of this method.
2. Which AbstractGrid methods call the isValid method? Why don't the other methods need to call it?

The getValidAdjacentLocations call the isValid method directly , the getEmptyAdjacentLocations and getOccpidAdjacentLocations call getValidAdjacentLocations to call isValid indirectly, and the getNeighbors calls getOccupidAdjacentLocations which calls isValid indirectly to use isValid method.

3. Which methods of the Grid interface are called in the getNeighbors method? Which classes provide implementations of these methods?
   The get and getOccupiedAdjacentLocations of the Grid interface are called in get Neighbors method.The AbstractGrid class implements the getOccupiedAdjacentLocations method while the BoundedGrid and UnboundedGrid classes implemented the get method.
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?
   The getEmptyAdjacentLocations must test whether the adjacent location exit and the get method return null if no object exist the location.E is indicate the generic object type,so Location can be return;

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?

The number of possible valid adjacent loctions would decrease to four whick are north, south, east and west nor eight;

Set 11

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

   The constructor of BoundedGrid limit that if rows <= 0 or colors <= 0 , it will throw an IllegalArgumentException.

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

   By the occupantArray[0].length statement.

   It assumption that the BoundedGrid object must has at least one row and one column.

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

The location's row value must greater than or equal to 0 and less than the number of the rows in the BounedGrid, the location's column value must greater than or equal to 0 and less than the number of the columns in the BounedGrid.

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

Oh) for this method?

ArrayList<Location> was returned by this method,O(r*c) to check the occupied locations, O(1) to add to the ArrayList.

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

   E was returned by the get method, which is a generic type indicate any type stored in the occupantArray, and it requires a Location object.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?

   When to object is null or the location is invalid. 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?

   The generic type E is returned by the remove method, it indicate any type stored in the BoundedGrid, null is return.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.

Yes, the time complexitys with these all method are O(1) except the getOccupiedLocations method, it is efficent.

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?

   The hashCode and ,the equals and the compareTo methoes must be implemented.The treeMap requires keys of the map tobe comparable, and the location satisfy these requirement.

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?

   The isValid method for the UnboundedGrid always return true so it need to checks for null in the get, put, and remove method instead.It it doesn't check, when call occupantMap.get(loc), it may encounter error.

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?

   The average time complexity for the three methods: get, put, and remove is O(1), if a TreeMap were used instead of HashMap, the average complexity would be O(log n), n is the number of occupied locations in the grid.

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

   The order of occupants return by getOccupiedLocations may be different.In the TreeMap, use different seach method such as inorder or preorder could return different order occupants and

the HashMap's order also determime by the hashCode and the size of the table.

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?

   Yes, a map could be used for a bouned grid, if a HashMap weree used, the average time complexity of the getOccupiedLocations would be O(n), n is the number of item in the grid.the map implementation may cost more memory, which stores not only item but also its location, while the two-dimensional array only store the items, the location are indicate by the index combining by the row and column.

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?

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)