Set 10

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

It’s specified by interface Grid, and implemented by classes UBoundedGrid ad BoundedGrid.

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

Public ArrayList<Location> getValidAdjacentLocations(Location loc);

Because the other methods call the getValidAdjacentLocations method, so they dont’t need to call the isValid method.

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

getOccupiedAdjacentLocations method, and it’s implemented by class AbstractGrid;

Get method, and it’s implemented by classed BoundedGrid and UnboundedGrid.

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?

Because method getEmptyAdjacentLocations needs to get the adjacent locations, and to judge whether a location is empty, we have to get the object in this location, and see whether the object is null or not. So the get method must be used to get the object in the neighboring locations.

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?

If we do so, then the getValidAdjacentLocations will only return the locations in the left, right, front and behind of loc and ignore the left-front, right-front, left-bottom and right-bottom locations.

Step11

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

In the constructor BoundedGrid, it will throw IllegalArgument exception if rows <= 0 or cols <=0, which ensures that a grid has 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?

The getNumCols method just returns the length of first row of array occupantArray. It assumes that the array occupantArray has been initialized.

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

The row of the location has to be bigger than 0 and smaller than the number of rows of grid, and the col of the location has to be bigger than 0 and smaller than the number of cols of grid.

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?

The getOccupiedLocations method returns ArrayList<Location>. The time complexity for this method will be 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?

The get method returns E, and the time complexity for it will be 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 the location input from the parameters is invalid in the grid, it will throw an IllegalArgumentException exception; when the object input the parameters is null, it will throw a NullPointerException exception. The time complexity for the put method will be 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 remove method returns E. When it tries to remove an item from an empty location, it will not happen anything and change anything. The time complexity for the remove method 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.

I think this is an efficient implementation. From questions 4, 5, 6 and 7, we can get that the general time complexity of this implementation is small.

Step12

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 Location class must implement the equals method, which will determine whether two locations point to the same place;it also has to implement the hashCode method to return the hash code of an object for the HashMap. In the meantime, we can also know that the Location class implements the interface Comparable, so it has to implements the comparaTo method from Comparable. If a TreeMap were used instead, the key of every location must to be compared to determined their order.

Location class satisfies all 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?

Because before we operates on the object in the input location, we have to make sure that the location is valid, and is not null. If the location is null or invalid, then there will be no object in the location, and we can not operates on it. So the get, put and remove methods has to check for the null. In the BoundedGrid class, the get, put and remove call the isValid method to check whether a location is valid or null, so it is not necessary for the them to check 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?

The average time complexity for get, put and remove methods is O(1). If a TreeMap were used, the average time complexity will be O(log n), while n is the number of occupants 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 the location stored in the map will be different while a TreeMap orders it keys in a ascending order and a HashMap orders it keys according to hash code of the key.

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?

A map implementation could also be used for a bounded grid. The two-dimensional array implementation only needs to store the location, which will reduce the use of memory, while the map implementation has to store the key and its value.

Exercises

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	r * c	r * c	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)