1 Category Partition Method

1.1 Parameter: List

Category L1 Empty list

Input: `list = []`

Reason: One of the edge cases of the `List` interface. Given this as the input, both `Collections.sort` and `Collections.rotate` should return an empty list as the output.

Category L2 Single element in the list

Input: `list =` $[x]$, where x is an element of a class that implements the `Comparable` interface.

Reason: One of the edge cases of the `List` interface. The output returned by both `Collections.sort` and `Collections.rotate` should be the same as the input.

Category L3 More than one element in the list

Input: `list =` $[x_0, x_1, \ldots, x_n]$ for $n > 1$, where n is the number of elements in the list, and $x_0, x_1, \ldots, x_n$ are the elements of a class that implements the `Comparable` interface.

Reason: The is the minimum viable test case for testing the actual functionality of `Collections.sort` and `Collections.rotate`. This is the base form that is used to define other variants of the input (**Category L4 - L7**).

Category L4 Duplicate elements in the list

Input: `list =` $[x_0, x_1, \ldots, x_n]$ for $n > 1$. The list follows the input restrictions of **Category L3**. **N.B.** In addition, there also exists some $\{a, b\} \in n$ where the value of x_a is equal to x_b.

Reason: This is to test whether the `Collections.sort` method is able to handle duplicate elements. The duplicate elements should be arranged next to each other in the sorted list.

Category L5 List is in random order

Input: `list =` $[x_0, x_1, \ldots, x_n]$ for $n > 1$. The list follows the input restrictions of **Category L3**, but the list is could be arranged in a way where $x_k < x_{k+1}$, but $x_{k+2} < x_k$ for all $k \in n$. The list can also contain duplicate elements.

Reason: This is to test the sorting capability of `Collections.sort`. It is expected that the input will be sorted into ascending order, according to the natural ordering of its elements.

Category L6 List is in ascending natural order of its elements

Input: `list = [`$x_0, x_1, \ldots, x_n$`]` for $n > 1$. The list follows the input restrictions of **Category L3**, and the list can contain duplicate elements. However, the list must be arranged according to the natural ordering [1] of its elements, where $x_0 \leq x_1,\ x_1 \leq x_2,\ \ldots,\ x_{n-1}$

Reason: This is to test the sorting consistency of `Collections.sort`. Since the input list is already sorted, then sorting the input list should return a list then is arranged in the same order as the input list.

Category L7 List is in descending natural order of its elements

Input: `list = [`$x_0, x_1, \ldots, x_n$`]` for $n > 1$. The list follows the input restrictions of **Category L3**, and the list can contain duplicate elements. However, the list must be arranged according to the **reverse** natural ordering [1] of its elements, where $x_0 \geq x_1,\ x_1 \geq x_2,\ \ldots,\ x_{n-1} \geq x_n$.

Reason: This is the inverse test case of **Category L6**, where the input list is sorted in reverse order. The aim is to ensure that `Collections.sort` can handle the cases where certain parts of the list (i.e. sub-list) are inversely sorted.

Category L8 List size is greater than or equal to `ROTATE_THRESHOLD`

Input: `list = [`$x_0, x_1, \ldots, x_n$`]` for $n \geq$ `ROTATE_THRESHOLD`, where the value of `ROTATE_THRESHOLD` is defined as 100 in `Collections.java` [4].

Reason: According to the source code of `Collections.java` [5], Java uses two different algorithm to rotate lists that are < or ≥ than `ROTATE_THRESHOLD` respectively. The aim is to validate that both algorithms are able to correctly rotate the input list.

1.2 Parameter: distance

Category D1 Negative number

Input: `distance < 0`

Reason: To ensure that the `Collections.rotate` method will work with negative distance input by covering the negative domain of `int` data type.

Category D2 Zero

Input: `distance == 0`

Reason: 0 is the default value of `int` data type in Java, and also the starting index value of `List`. This is to ensure that the `Collections.rotate` method will work when the input distance is zero.

Category D3 Positive number

Input: `distance > 0`

Reason: To ensure that the `Collections.rotate` method will work with positive distance input by covering the positive domain of `int` data type.

Category D4 Smaller than list size

Input: `distance < list.size()`

Reason: A stricter variant of **Category D1 - D3**.

Category D5 Equal to list size

Input: `distance == list.size()`

Reason:

Category D6 Larger than list size

Input: `distance > list.size()`

Reason: A stricter variant of **Category D3**.

Category D7 Equal to minimum boundary value of `int`

Input: `distance ==` -2^{31}

Reason: The minimum value an `int` can have in Java is -2^{31} [3]. The aim is to validate the assumption that if `Collections.rotate` works for the minimum value of `int`, then it should work correctly for any value larger than the minimum value.

Category D8 Equal to maximum boundary value of `int`

Input: `distance ==` $2^{31} - 1$

Reason: The maximum value an `int` can have in Java is $2^{31} - 1$ [2]. The aim is to validate the assumption that if `Collections.rotate` works for the maximum value of `int`, then it should work correctly for any value smaller than the maximum value.

1.3 Parameter: Collection

Category C1 Single element in collection

Input:

Reason:

Category C2 More than one element in collection

Input:

Reason:

Category C3 Duplicate elements in collection

Input:

Reason:

Category C4 Duplicate minimum elements in collection

Input:

Reason:

Category C5 Collection is in random order

Input:

Reason:

Category C6 Collection is in ascending natural order of its elements

Input:

Reason:

Category C7 Collection is in descending natural order of its elements

Input:

Reason:

2 Test Cases

2.1 `Collections.sort(List<T> list)`

2.2 `Collections.rotate(List<?> list, int distance)`

2.3 `Collections.min(Collection<? extends T> coll)`

3 Metamorphic Relations

4 Remarks

References

[1] *Comparable (java platform se 8)*, Oracle Corporation, 5th Dec. 2019. [Online]. Available: `https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html`.

[2] *Integer max value*, Oracle Corporation, 5th Dec. 2019. [Online]. Available: `https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html#MAX_VALUE`.

[3] *Integer min value*, Oracle Corporation, 5th Dec. 2019. [Online]. Available: `https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html#MIN_VALUE`.

[4] *Jdk/collections.java*, Oracle Corporation, 5th Dec. 2019. [Online]. Available: `https://github.com/openjdk/jdk/blob/81ec9e3087764708/src/java.base/share/classes/java/util/Collections.java#L779`.

[5] *Jdk/collections.java*, Oracle Corporation, 5th Dec. 2019. [Online]. Available: `https://github.com/openjdk/jdk/blob/81ec9e3087764708/src/java.base/share/classes/java/util/Collections.java#L779`.