**The data structure:**

**SequenceWithMinimum:** We create a new class with two elements of the type “Node”. The sequence consists of a series of Nodes. The most left node is called “head”, and the most right node is called “tail”.  
Each **Node** has a reference to the left node and the right node,and an integer to present the value.

**Implementation and the complexity:**

**– public void insertRight(Integer value)**

|  |  |  |
| --- | --- | --- |
| IF tail is null | O(1) | O(1) |
| tail is a new Node(value) | O(1) |
| head is equal to tail | O(1) |
| ELSE | O(1) | O(1) |
| newNode is a new Node(value) | O(1) |
| left node of newNode is equal to tail | O(1) |
| right node of tail is equal to newNode | O(1) |
| tail is equal to newNode | O(1) |
| ENDIF |  |  |

max(O(1),O(1)= O(1)

**– public Integer removeRight()**

|  |  |  |
| --- | --- | --- |
| IF tail is null or tail is null | O(1) | O(1) |
| return null |  |
| ENDIF |  |  |
| int temp is equal to the value of tail | O(1) | O(1) |
| IF left node of tail is null | O(1) | O(1) |
| tail is null | O(1) |
| head is null | O(1) |
| ELSE |  | O(1) |
| tail is equal to left node of tail | O(1) |
| right node of tail is null | O(1) |
| ENDIF |  |  |
| return temp |  |  |

O(1)+ O(1)+max(O(1), O(1)) = O(1)

**– public void insertLeft(Integer value)**

|  |  |  |
| --- | --- | --- |
| IF head is null | O(1) | O(1) |
| head is a new Node(value) | O(1) |
| head is equal to tail | O(1) |
| ELSE |  |  |
| newNode is a new Node(value) | O(1) | O(1) |
| right node of newNode is equal to head | O(1) |
| left node of head is equal to newNode | O(1) |
| head is equal to newNode | O(1) |
| ENDIF |  |  |

max(O(1),O(1)= O(1)

**– public Integer removeLeft()**

|  |  |  |
| --- | --- | --- |
| IF tail is null or tail is null | O(1) | O(1) |
| return null |  |
| ENDIF |  |  |
| int temp is equal to the value of head | O(1) | O(1) |
| IF right node of head is null | O(1) | O(1) |
| tail is null | O(1) |
| head is null | O(1) |
| ELSE | O(1) | O(1) |
| head is equal to right node of head | O(1) |
| left node of head is null | O(1) |
| ENDIF |  |  |
| return temp |  |  |

O(1)+ O(1)+max(O(1), O(1)) = O(1)

**– public Integer findMinimum()**

|  |  |  |
| --- | --- | --- |
| Node current is equal to head | O(1) | O(1) |
| IF current is null | O(1) | O(1) |
| return null |  |  |
| ENDIF |  |  |
| int min is equal to the value of current | O(1) | O(1) |
| WHILE right node of current is not null | N TIMES | O(N) |
| current is equal to the right node of current | O(1) |
| IF the value of current is less than min | O(1) |
| min is equal to the value of current | O(1) |
| ENDIF |  |  |
| ENDWHILE |  |  |
| return min |  |  |

O(1)+ O(1)+ O(1)+ O(N) = O(N)