Alumno—
Luis Fernando Aguilar Ortiz
Victor Antonio González Bojórquez
ID—
- 233326
- 228419
FECHA—
29/11/2023

Proyecto final—
Análisis de complejidad

Materia—
Estructuras de datos

Profesor—
Sergio Castellanos Bustamante

Este es un análisis de la complejidad dividido por métodos y clases.

- ***LinkedList{***

T(n)=2n+9

public void addTask(Task task) {	
Node newNode = new Node(task);	**1**
if (head == null) {	**max(1**
head = newNode;	**1**
vacio = false;	**1**
size++;	**1)**
System.out.println("agregado");	
display();	
} else {	
Node current = head;	**1**
while (current.next != null) {	**n**
current = current.next;	**n**
}	
System.out.println("agregado");	
display();	
current.next = newNode;	**1**
vacio = false;	**1**
size++;	**1**
}	
}	

T(n)=3n+8

public void removeTask(Task task) {	
Node current = head;	**1**
Node prev = null;	**1**
while (current != null && current.task != task) {	**n**
prev = current;	**n**
current = current.next;	**n**
}	
if (current != null) {	**1**
if (prev == null) {	**1**
head = current.next;	**1**
size--;	**1**
} else {	
prev.next = current.next;	**1**
size--;	**1**
}	
}	

}	

T(n)=5

public boolean empty() {	
if (head == null) {	1
vacio = true;	1
return true;	1
} else {	
vacio = false;	1
return false;	1
}	
}	

T(n)=5n+3

public Task buscarPorPosicion(int posicion) {	
Node temp ;	
temp = head;	1
for (int x = 0; x <= size; x++) {	1+n+n
if(temp.task.getPosicion()!=posicion){	n
temp= temp.getNext();	n
}else{	
return temp.task;	n
}	
}	
return null;	1
}	

T(n)=21n+7

public String display() {	
Node temp ;	
String resultado ="->",tarea = "{";	1+1
temp = head;	1
if(temp!=null){	1
for (int x = 0; x <= size; x++) {	1+n+n
tarea+="titulo ";	n
tarea+=temp.task.getTitulo();	n
tarea+=". descripcion ";	n
tarea+=temp.task.getDescripcion();	n
tarea+=". fecha anadida a la lista ";	n

tarea+=temp.task.getAñadidaF().toString();
tarea+=". prioridad de la tarea ";
tarea+=temp.task.getPrioridad();
tarea+=". fecha vencimiento de la tarea ";
tarea+=temp.task.getVenceF().toString();
tarea+=". posicion en la lista ";
tarea+=temp.task.getPosicion();
 	
resultado+=("tarea "+temp.task.getPosicion()+" posicion en lista= "+(x+1)+" : "+tarea+"}, ");
System.out.println(resultado);
tarea= "{";
if(temp.getNext()!=null){
temp=temp.getNext();}
} return resultado;
}else{
System.out.println("lista vacia");
 	
return resultado;
}
}
	n
	n
	n
	n
	n
	n
	n
	
	n+n+n
	n+n
	n
	n
	n
	n
	1
	
	1

T(n)=1

public void quickSort() {
head = quickSortRec(head);
display();
}
	
	1

T(n)=7n+11

private Node quickSortRec(Node node) {
if (node == null || node.next == null) {
display();
return node;
 	
}
Task pivotTask = node.task;
Node less = null;
Node equal = null;
	
	1+1
	
	1
	
	1
	1
	1

Node greater = null;
Node current = node;
while (current != null) {
if (current.task.compareTo(pivotTask) < 0) {
less = addToEnd(less, current.task);
} else if (current.task.compareTo(pivotTask) == 0) {
equal = addToEnd(equal, current.task);
} else {
greater = addToEnd(greater, current.task);
}
current = current.next;
}
less = quickSortRec(less);
greater = quickSortRec(greater);
display();
return concatenate(less, equal, greater);
}
	1
	1
	n
	n
	n
	n
	n
	
	n
	
	n
	
	1
	1
	
	1

T(n)=2n+6

private Node addToEnd(Node list, Task data) {
Node newNode = new Node(data);
if (list == null) {
return newNode;
}
Node current = list;
while (current.next != null) {
current = current.next;
}
current.next = newNode;
return list;
}
	
	1
	1
	1
	
	1
	n
	n
	
	1
	1

T(n)=9

private Node concatenate(Node less, Node equal, Node greater) {	
Node result = less;	1
Node lessTail = getTail(less);	1
if (lessTail != null) {	1
lessTail.next = equal;	1
} else {	
result = equal;	1
}	
Node equalTail = getTail(equal);	1
if (equalTail != null) {	1
equalTail.next = greater;	1
}	
return result;	1
}	

T(n)=2n+4

private Node getTail(Node list) {	
if (list == null) {	1
return null;	1
}	
Node current = list;	1
while (current.next != null) {	n
current = current.next;	n
}	
return current;	1
}	

- ***Task{***
- **T(n)=5**

public int compareTo(Task otherTask) {	
// Comparar tareas por fecha	
int dateComp = this.venceF.compareTo(otherTask.venceF);	1+1
if (dateComp != 0){	1
return dateComp;	1
}	
// Comparar tareas por prioridad	

return Integer.compare(this.prioridad, otherTask.prioridad); }	1

T(n)=1

public String toString() {
return titulo;
}
	1

- ***TodoListGui{***
- **T(n)=5**

protected Transferable createTransferable(JComponent c) {
JList<Task> list = (JList<Task>) c;
Task selectedTask = list.getSelectedValue();
if (selectedTask != null) {
return new TaskTransferable(selectedTask);
}
return null;
}
	
	1
	1
	1
	1
	
	1

T(n)=13

public boolean importData(TransferHandler.TransferSupport support)
{
if (!canImport(support)) {
return false;
}
JList.DropLocation dl = (JList.DropLocation)
support.getDropLocation();
int index = dl.getIndex();
try {
Task dT = (Task) support.getTransferable().
getTransferData(TaskFlavor.getInstance());
if (index != -1 && index != listModel.indexOf(dT)) {
listModel.removeElement(dT);
listModel.add(index, dT);
}
return true;
	
	1
	1
	
	1
	1
	1
	
	1
	1
	1+1
	1
	1
	
	1

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        return false;                                                    1
    }
```

T(n)=29

```
public TodoListGui() {

    todoList = new TodoList();                                           1
    listModel = new DefaultListModel<>();                                1
    JList<Task> taskList = new JList<>(listModel);                       1

taskList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    taskList.setTransferHandler(new TaskTransferHandler());
    taskList.setDropMode(DropMode.INSERT);









    // Método para cambiar el nombre de la tarea haciendo doble
click
    taskList.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            if (e.getClickCount() == 2) {                                1
                int selectedIndex = taskList.getSelectedIndex();         1
                if (selectedIndex != -1) {                               1
                    Task selectedTask = listModel.get(selectedIndex);    1
                    String newTaskName =
JOptionPane.showInputDialog("Edit Task", selectedTask.titulo);           1
                    if (newTaskName != null) {                           1
                        selectedTask.titulo = newTaskName;               1
                    }
                }
            }
        }
    });
```

```
    // Método que añade las tareas a la lista TODO
    JScrollPane scrollPane = new JScrollPane(taskList);                    1
    scrollPane.setPreferredSize(new Dimension(300, 200));
    JTextField prioridadField = new JTextField("prioridad",5);             1
    JTextField descripcionField = new JTextField("descripcion",20);        1
    JTextField tituloField = new JTextField("titulo",8);                   1
    JButton addTaskButton = new JButton("Add Task");                       1

//logica boton agregar tarea a la lista
    addTaskButton.addActionListener((ActionEvent e) -> {
      String pri = prioridadField.getText().trim();                        1
      int prioridad = Integer.parseInt(pri);                               1
      Date venceF=selectorFecha.getDate();                                 1
      String titulo = tituloField.getText().trim();                        1
      String descripcion = descripcionField.getText().trim();              1
     //  Date venceF = (Date) datePicker.getModel().getValue();             1

      if (!titulo.isEmpty()) {                                             1
        Task newTask = new
Task(prioridad,todoList.linkedList.size+1,venceF,descripcion,titulo);      1
        todoList.addTask(newTask);
        listModel.addElement(newTask);
        prioridadField.setText("");
        descripcionField.setText("");
        tituloField.setText("");
      }
    });



    // Método para eliminar las tareas de la lista
    JButton removeTaskButton = new JButton("Remove Task");                 1
    removeTaskButton.addActionListener((ActionEvent e) -> {
      int selectedIndex = taskList.getSelectedIndex();                     1
      if (selectedIndex != -1) {                                           1
        Task selectedTask = listModel.get(selectedIndex);                  1
        todoList.removeTask(selectedTask);
        listModel.remove(selectedIndex);
      }
    });
```

```
    // Método para mezclar la lista con el algoritmo quicksort
    JButton sortButton = new JButton("Sort");
    sortButton.addActionListener((ActionEvent e) -> {
        todoList.quickSort();
    });
    JPanel panel = new JPanel();
    panel.add(scrollPane);
    panel.add(tituloField);
    panel.add(descripcionField);
    panel.add(prioridadField);
    panel.add(addTaskButton);
    panel.add(removeTaskButton);
    panel.add(sortButton);
    add(panel);
    setTitle("TODO List");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    pack();
    setLocationRelativeTo(null);
    setVisible(true);
      initComponents();
}
```

1

1