Análisis de complejidad temporal y espacial.

```java
public int convertStringToNumbers(K key) {
    int values = 0;
    String s = key.toString();
    for (int i = 0; i < s.length() ; i++)
        values += s.charAt(i) * (i + 1);
    return values;
}
```

Instrucciones	Veces que se repite (Notación Big O)
`1. int values = 0;`	1
`2. String s = key.toString();`	1
`3. for (int i = 0; i < s.length() ; i++)`	n + 1
`4. values += s.charAt(i) * (i + 1);`	n
`5. return values;`	1

Tipo	Variable	Tamaño de 1 valor atómico	Cantidad de valores atómicos
Entrada	key	64 bits	1
Auxiliar	s i	64 bits 32 bits	1 1
Salida	values	32 bits	1

Complejidad espacial total = Entrada + Auxiliar + Salida = 1 + 2 + 1 = 4 = O(4) = O(1).

Complejidad espacial auxiliar = Auxiliar = 2 = O(2) = O(1).

Complejidad espacial auxiliar + salida = Auxiliar + Salida = 2 + 1 = 3 = O(3) = O(1).

```java
public void enqueue(T data) {
    Node<T> element = new Node<>(data);
    if( isEmpty() ) {
        first = element;
        last = first;
    } else {
        Node<T> temporal = first;

        while( temporal.getNext() != null )
            temporal = temporal.getNext();

        temporal.setNext(element);
        last = element;
    }
}
```

Instrucciones	Veces que se repite (Notación Big O)
1. Node<T> element = new Node<>(data);	1
2. If(isEmpty()) {	1
3. first = element;	1
4. last = first;	1
5. } else {	1
6. Node<T> temporal = first;	1
7. while(temporal.getNext() ¡= null)	n + 1
8. temporal = temporal.getNext();	n
9. temporal.setNext(element);	1
10. last = element;	1

Tipo	Variable	Tamaño de 1 valor atómico	Cantidad de valores atómicos
Entrada	data	64 bits	1
Auxiliar	temporal	64 bits	1
Salida	Ninguno	0	0

Complejidad espacial total = Entrada + Auxiliar + Salida = 1 + 1 + 0 = 2 = O(2) = O(1).

Complejidad espacial auxiliar = 1 = O(1).

Complejidad espacial auxiliar + salida = Auxiliar + Salida = 1 + 0 = 1 = O(1).

```
private void changeIncElement(int i) {
    if(list.size()>1) {
        int parent=(i/2)-1;
        while((i-1>0) && (((Patient) list.get(parent).getPatient()).compareTo((Patient) list.get(i-1).getPatient())<0 |
                (((Patient) list.get(parent).getPatient()).compareTo((Patient) list.get(i-1).getPatient())==0 && list.get(parent).getEntery()-list.get(i-1).getEntery()>0))) {
            PriorityNode<K> temp= list.get(parent);
            list.set(parent, list.get(i-1));
            list.set(i-1, temp);

            i=parent+1;
            parent=(i/2)-1;
        }
    }

}
```

Instrucciones	**Veces que se repite (Notación Big O)**
`1. if(list.size() > 1) {`	1
`2. int parent=(i/2)-1;`	1
`3. while((i-1>0) && (((Patient) list.get(parent).getPatient()).compareTo((Patient) list.get(i-1).getPatient())<0 | (((Patient) list.get(parent).getPatient()).compareTo((Patient) list.get(i-1).getPatient()) == 0 && list.get(parent).getEntery() – list.get(i-1).getEntery() > 0))) {`	n + 2
`4. PriorityNode<K> temp = list.get(parent);`	n
`5. list.set(parent, list.get(i-1));`	n
`6. list.set(i-1, temp);`	n
`7. i = parent + 1;`	n
`8. parent = (i / 2) – 1;`	n

Tipo	**Variable**	**Tamaño de 1 valor atómico**	**Cantidad de valores atómicos**
Entrada	i	32 bits	1
Auxiliar	parent temp	32 bits 64 bits	1 1
Salida	Ninguna	0	0

Complejidad espacial total = Entrada + Auxiliar + Salida = 1 + 2 + 0 = 3 = O(3) = O(1).

Complejidad espacial auxiliar = Auxiliar = 2 = O(2) = O(1).

Complejidad espacial auxiliar + salida = Auxiliar + Salida = 2 + 0 = 2 = O(2) = O(1).