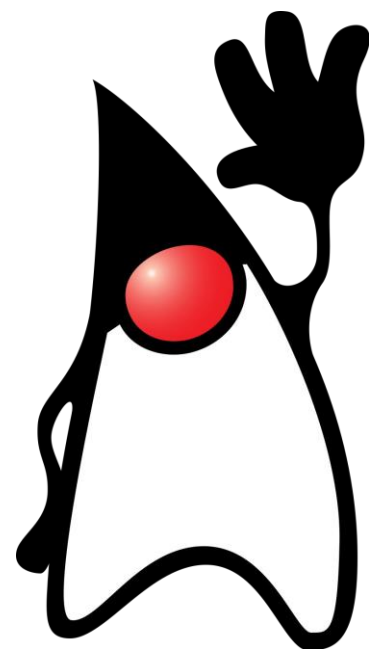




# Búsqueda binaria

Kevin Campos Venegas  
Taller de programación II  
[kcampos@ing.ucsc.cl](mailto:kcampos@ing.ucsc.cl)



# Imports

```
import java.util.Scanner;  
import java.util.List;  
import java.util.ArrayList;  
Import java.util.Collections;  
import java.util.Arrays;
```


# Collections

- Sort
- Reverse
- BinarySearch
- Shuffle

API Java Collections: <https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html>

# BinarySearch

1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



$$\begin{aligned}\text{Promedio} &= (\text{min} + \text{max}) / 2 \\ &= (0 + 15) / 2 \\ &= 7.5 \approx 7\end{aligned}$$

# BinarySearch


1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

**Elemento a buscar : 18**

**Elemento central : 21**

# BinarySearch

1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



- Si es mayor, cambiar máximo a posición del elemento central - 1 (intervalo azul).
- Si es menor, cambiar mínimo a posición del elemento central + 1 (intervalo rojo).

# BinarySearch

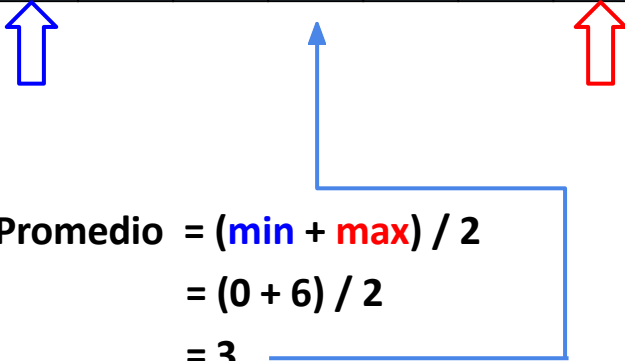
1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Como 21 es mayor que 18, se repite tomando el intervalo de la **izquierda**.

# BinarySearch

1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



$$\begin{aligned}\text{Promedio} &= (\text{min} + \text{max}) / 2 \\ &= (0 + 6) / 2 \\ &= 3\end{aligned}$$



# BinarySearch


1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

**Elemento a buscar : 18**

**Elemento central : 8**

# BinarySearch

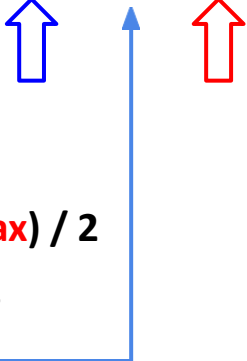
1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Como 8 no es mayor que 18, se repite tomando el intervalo de la **derecha**.

# BinarySearch

1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



$$\begin{aligned}\text{Promedio} &= (\text{min} + \text{max}) / 2 \\ &= (4 + 6) / 2 \\ &= 5\end{aligned}$$

# BinarySearch



1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

**Elemento a buscar : 18**

**Elemento central : 15**

# BinarySearch

1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Como 15 no es mayor que 18, se repite tomando el intervalo de la **derecha**.

# BinarySearch

1	4	7	8	11	15	18	21	24	25	26	27	30	33	35	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

**Elemento a buscar : 18**

**Elemento central : 18**

# Collections

- **Sort**
- **Reverse**
- **BinarySearch**
- **Shuffle**

API Java Collections: <https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html>

# Collections métodos

```
List<Integer> arr = new ArrayList<>();  
arr.add(7);  
arr.add(1);  
arr.add(4);  
arr.add(11);  
arr.add(8);
```

Tipo	Método	Descripción
void	<code>Collections.sort(List list);</code>	Ordena la lista especificada en orden ascendente, de acuerdo con el <b>orden natural</b> de sus elementos.
<code>Collections.sort(arr); //Ordena la lista ascendentemente [1, 4, 7, 8, 11]</code>		



# Collections métodos

```
List<Integer> arr = new ArrayList<>();  
arr.add(7);  
arr.add(1);  
arr.add(4);  
arr.add(11);  
arr.add(8);
```

```
void Collections.reverse(List list);
```

Invierte el orden de los elementos en la lista especificada.

```
Collections.reverse(arr); //Invierte el orden de los elementos [8, 11, 4, 1, 7]
```

# Collections métodos

```
List<Integer> arr = new ArrayList<>();
arr.add(1);
arr.add(4);
arr.add(7);
arr.add(8);
arr.add(11);
```

Tipo	Método	Descripción
int	<code>Collections.binarySearch(List list, int number);</code>	Busca en la lista el objeto especificado mediante <b>Búsqueda Binaria</b> .
	<pre>int index = Collections.binarySearch(arr, 8); //Retorna el índice del número dado, en este caso retorna 3</pre>	
void	<code>Collections.shuffle(List list);</code>	Permuta aleatoriamente la lista especificada.
	<code>Collections.shuffle(arr); //Desordena los elementos de la lista</code>	

# Arrays

- Sort
- BinarySearch
- ToString

API Java Arrays: <https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html>

# Arrays métodos

```
int vec[] = new int[5];  
vec[0] = 7;  
vec[1] = 1;  
vec[2] = 4;  
vec[3] = 11;  
vec[4] = 8;
```

Tipo	Método	Descripción
void	<code>Arrays.sort(int vec[]);</code>	Ordena el vector especificado en orden ascendente, de acuerdo con el <b>orden natural</b> de sus elementos.
	<code>Arrays.sort(vec); //Ordena el vector ascendentemente [1, 4, 7, 8, 11]</code>	
int	<code>Arrays.binarySearch(int vec[], int number);</code>	Busca en el vector el objeto especificado mediante <b>Búsqueda Binaria</b> .
	<code>int index = Arrays.binarySearch(vec, 8); //Retorna el índice del número dado, en este caso retorna 3</code>	

# Arrays métodos

```
int vec[] = new int[5];  
vec[0] = 7;  
vec[1] = 1;  
vec[2] = 4;  
vec[3] = 11;  
vec[4] = 8;
```

Tipo	Método	Descripción
void	<code>Arrays.toString(int vec[]);</code> <code>System.out.println(Arrays.toString(vec));</code> <code>//Imprime el vector con el siguiente formato: [7, 1, 4, 11, 8]</code>	Retorna una representación en String del vector especificado.