

TALLER PROGCOMP: TRACK BÁSICO

BÚSQUEDA BINARIA

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

April 29, 2024

Part I

BUSCAR UN ELEMENTO

PROBLEMA

Problema

Dado una secuencia **ordenada** de elementos de tamaño n , se quiere saber el primer elemento (de izquierda a derecha) que cumpla cierta condición.

CASO DEL LOWER_BOUND

Problema

Determinar la posición del primer elemento que **NO SEA MENOR** a x .

CASO DEL LOWER_BOUND

Problema

Determinar la posición del primer elemento que **NO SEA MENOR** a x .

Por ejemplo, $x = 4$.

1	2	2	3	3	4	5	7
---	---	---	---	---	---	---	---

CASO DEL LOWER_BOUND

Problema

Determinar la posición del primer elemento que **NO SEA MENOR** a x .

Por ejemplo, $x = 4$.

1	2	2	3	3	4	5	7
F	F	F	F	F	T	T	T

CASO DEL LOWER_BOUND

Problema

Determinar la posición del primer elemento que **NO SEA MENOR** a x .

Por ejemplo, $x = 4$.

1	2	2	3	3	4	5	7
F	F	F	F	F	T	T	T

CASO DEL LOWER_BOUND

Problema

Determinar la posición del primer elemento que **NO SEA MENOR** a x .

Por ejemplo, $x = 4$.

1	2	2	3	3	4	5	7
F	F	F	F	F	T	T	T

CASO DEL LOWER_BOUND

Problema

Determinar la posición del primer elemento que **NO SEA MENOR** a x .

Por ejemplo, $x = 4$.

1	2	2	3	3	4	5	7
F	F	F	F	F	T	T	T

CASO DEL LOWER_BOUND

Problema

Determinar la posición del primer elemento que **NO SEA MENOR** a x .

Por ejemplo, $x = 4$.

1	2	2	3	3	4	5	7
F	F	F	F	F	T	T	T

TIPOS DE HACER BÚSQUEDA BINARIA

- ▶ Buscar el primer T
- ▶ Buscar el último T

BUSCAR EL PRIMER T

```
1  int main() {
2      int l; // limite izquierdo
3      int r; // limite derecho
4      while(l < r) {
5          int mid = (l + r) / 2;
6          if(condicion) {
7              r = mid;
8          } else {
9              l = mid + 1;
10         }
11     }
12     // l y r deberian ser iguales
13     return 0;
14 }
```

BUSCAR EL PRIMER T

```
1  int main() {
2      int l; // limite izquierdo
3      int r; // limite derecho
4      while(l < r) {
5          int mid = (l + r) / 2;
6          if(condicion) {
7              r = mid;
8          } else {
9              l = mid + 1;
10         }
11     }
12     // l y r deberian ser iguales
13     return 0;
14 }
```

¿Cuál sería la complejidad?

BUSCAR EL PRIMER T

```
1  int main() {
2      int l; // limite izquierdo
3      int r; // limite derecho
4      while(l < r) {
5          int mid = (l + r) / 2;
6          if(condicion) {
7              r = mid;
8          } else {
9              l = mid + 1;
10         }
11     }
12     // l y r deberian ser iguales
13     return 0;
14 }
```

¿Cuál sería la complejidad? $O(f(n) \cdot \log_2(r - l + 1))$

BUSCAR EL PRIMER T - CASO LOWER_BOUND

```
1  bool condicion(vector< int > &nums, int mid, int x) {
2      if(nums[mid] < x) return false;
3      return true;
4  }
5
6  int main() {
7      int x; // numero de la condicion del lower_bound
8      vector< int > nums; // se asume que lo llenas eventualmente
9      int l; // limite izquierdo
10     int r; // limite derecho
11     while(l < r) {
12         int mid = (l + r) / 2;
13         if(condicion(nums, mid, x)) {
14             r = mid;
15         } else {
16             l = mid + 1;
17         }
18     }
19     // l y r deberian ser iguales
20     return 0;
21 }
```

BUSCAR EL PRIMER T - CASO LOWER_BOUND

```
1  bool condicion(vector< int > &nums, int mid, int x) {
2      if(nums[mid] < x) return false;
3      return true;
4  }
5
6  int main() {
7      int x; // numero de la condicion del lower_bound
8      vector< int > nums; // se asume que lo llenas eventualmente
9      int l; // limite izquierdo
10     int r; // limite derecho
11     while(l < r) {
12         int mid = (l + r) / 2;
13         if(condicion(nums, mid, x)) {
14             r = mid;
15         } else {
16             l = mid + 1;
17         }
18     }
19     // l y r deberian ser iguales
20     return 0;
21 }
```

¿Cuál sería la complejidad?

BUSCAR EL PRIMER T - CASO LOWER_BOUND

```
1  bool condicion(vector< int > &nums, int mid, int x) {
2      if(nums[mid] < x) return false;
3      return true;
4  }
5
6  int main() {
7      int x; // numero de la condicion del lower_bound
8      vector< int > nums; // se asume que lo llenas eventualmente
9      int l; // limite izquierdo
10     int r; // limite derecho
11     while(l < r) {
12         int mid = (l + r) / 2;
13         if(condicion(nums, mid, x)) {
14             r = mid;
15         } else {
16             l = mid + 1;
17         }
18     }
19     // l y r deberian ser iguales
20     return 0;
21 }
```

¿Cuál sería la complejidad? $O(1 \cdot \log_2(n))$

BUSCAR EL ÚLTIMO T

```
1  int main() {
2      int l; // limite izquierdo
3      int r; // limite derecho
4      while(l < r) {
5          int mid = (l + r + 1) / 2;
6          if(condicion) {
7              l = mid;
8          } else {
9              r = mid - 1;
10         }
11     }
12     // l y r deberian ser iguales
13     return 0;
14 }
```

BUSCAR EL ÚLTIMO T

```
1  int main() {
2      int l; // limite izquierdo
3      int r; // limite derecho
4      while(l < r) {
5          int mid = (l + r + 1) / 2;
6          if(condicion) {
7              l = mid;
8          } else {
9              r = mid - 1;
10         }
11     }
12     // l y r deberian ser iguales
13     return 0;
14 }
```

¿Cuál sería la complejidad?

BUSCAR EL ÚLTIMO T

```
1  int main() {
2      int l; // limite izquierdo
3      int r; // limite derecho
4      while(l < r) {
5          int mid = (l + r + 1) / 2;
6          if(condicion) {
7              l = mid;
8          } else {
9              r = mid - 1;
10         }
11     }
12     // l y r deberian ser iguales
13     return 0;
14 }
```

¿Cuál sería la complejidad? $O(f(n) \cdot \log_2(r - l + 1))$

REFERENCES I