Taller ProgComp: Track Básico

Búsqueda Binaria en la librería estándar

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

April 22, 2024

Part I

BUSCAR UN ELEMENTO

PROBLEMA

Problema

Dado una secuencia **ordenada** de elementos de tamaño n, se quiere saber si es que un elemento se encuentra en la secuencia y si es que se encuentra en cuál es su posición.

Solución Trivial

- Ir desde la posición 0 hasta la $n - 1$:
  - Si se encuentra el elemento retornar la posición
  - Si no se encuentra retornar el tamaño del arreglo

```
int find(vector< int > &v, int x) {
  for(int i = 0; i < v.size(); i++) {
    if(v[i] == x) {
      return i;
    }
  }
  return v.size();
}
```

Solución Trivial - Librería Estándar

```cpp
int main() {
  vector< int > v;
  int x;
  // se asume que aqui se rellena el vector y la variable x

  vector< int >::iterator it = find(v.begin(), v.end(), x);
  if(it != v.end()) {
    cout << "Se encontro el elemento!" << "\n";
    cout << "El valor es: " << *it << "\n";
    int pos = it - v.begin();
    cout << "La posicion es: " << pos << "\n";
  } else {
    cout << "No se encontro el valor D:" << "\n";
    int size = it - v.begin();
    cout << "Miren lo curioso: " << size << "\n";
  }
  return 0;
}
```

Solución Trivial - Librería Estándar

```cpp
int main() {
  vector< int > v;
  int x;
  // se asume que aqui se rellena el vector y la variable x

  vector< int >::iterator it = find(v.begin(), v.end(), x);
  if(it != v.end()) {
    cout << "Se encontro el elemento!" << "\n";
    cout << "El valor es: " << *it << "\n";
    int pos = it - v.begin();
    cout << "La posicion es: " << pos << "\n";
  } else {
    cout << "No se encontro el valor D:" << "\n";
    int size = it - v.begin();
    cout << "Miren lo curioso: " << size << "\n";
  }
  return 0;
}
```

¿Cuál sería la complejidad de estos algoritmos?

Solución Trivial - Librería Estándar

```cpp
int main() {
  vector< int > v;
  int x;
  // se asume que aqui se rellena el vector y la variable x

  vector< int >::iterator it = find(v.begin(), v.end(), x);
  if(it != v.end()) {
    cout << "Se encontro el elemento!" << "\n";
    cout << "El valor es: " << *it << "\n";
    int pos = it - v.begin();
    cout << "La posicion es: " << pos << "\n";
  } else {
    cout << "No se encontro el valor D:" << "\n";
    int size = it - v.begin();
    cout << "Miren lo curioso: " << size << "\n";
  }
  return 0;
}
```

¿Cuál sería la complejidad de estos algoritmos? $O(n)$

Solución Trivial - Librería Estándar

```cpp
int main() {
  vector< int > v;
  int x;
  // se asume que aqui se rellena el vector y la variable x

  vector< int >::iterator it = find(v.begin(), v.end(), x);
  if(it != v.end()) {
    cout << "Se encontro el elemento!" << "\n";
    cout << "El valor es: " << *it << "\n";
    int pos = it - v.begin();
    cout << "La posicion es: " << pos << "\n";
  } else {
    cout << "No se encontro el valor D:" << "\n";
    int size = it - v.begin();
    cout << "Miren lo curioso: " << size << "\n";
  }
  return 0;
}
```

¿Cuál sería la complejidad de estos algoritmos? $O(n)$
Si el arreglo estuviera desordenado, no tenemos mejor forma. Pero el arreglo esta **ordenado**...

Part II

Búsqueda Binaria en STL

Búsqueda Binaria

Idea

Aprovechemos que la secuencia de elementos esta ordenada ;).
Tomemos el elemento de la mitad del arreglo, llamemoslo a_m:

- Si $x > a_m$, nos queda buscar entre $a_{m+1}, a_{m+2}, \ldots, a_{n-1}$
- Sino, nos queda buscar entre $a_0, a_1, \ldots, a_m$

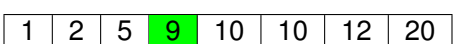
VISUALMENTE

Busquemos el elemento 10.

1	2	5	9	10	10	12	20

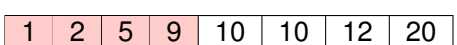
VISUALMENTE

Busquemos el elemento 10.



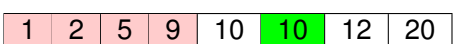
VISUALMENTE

Busquemos el elemento 10.



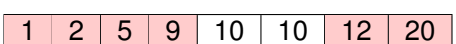
VISUALMENTE

Busquemos el elemento 10.



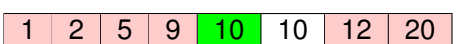
VISUALMENTE

Busquemos el elemento 10.



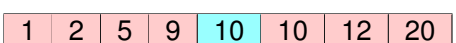
VISUALMENTE

Busquemos el elemento 10.



VISUALMENTE

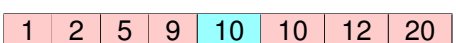
Busquemos el elemento 10.



Lo encontramos!

VISUALMENTE

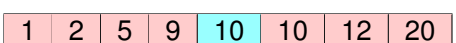
Busquemos el elemento 10.



Lo encontramos! ¿Cuál sería la complejidad?

VISUALMENTE

Busquemos el elemento 10.



Lo encontramos! ¿Cuál sería la complejidad? $O(\log n)$

Búsqueda Binaria - Librería Estándar

```cpp
int main() {
  vector< int > v;
  int x;
  // se asume que aqui se rellena el vector y la variable x
  // recordar que el vector/arreglo debe estar ordenado

  if(binary_search(v.begin(), v.end(), x)) {
    cout << "Se encontro el numero!" << "\n";
  } else {
    cout << "No se encontro el numero!" << "\n";
  }
  return 0
}
```

Búsqueda Binaria - lower_bound

Definición

La posición del primer elemento que **NO SEA MENOR** al elemento que se esta búscando.

Ejemplo

Busquemos la primer elemento que no sea menor al 10

1	2	5	9	10	10	12	20

Búsqueda Binaria - lower_bound

Definición

La posición del primer elemento que **NO SEA MENOR** al elemento que se esta búscando.

Ejemplo

Busquemos la primer elemento que no sea menor al 10

1	2	5	9	10	10	12	20

Búsqueda Binaria - lower_bound

Definición

La posición del primer elemento que **NO SEA MENOR** al elemento que se esta búscando.

Ejemplo

Busquemos la primer elemento que no sea menor al 11

1	2	5	9	10	10	12	20

Búsqueda Binaria - lower_bound

Definición

La posición del primer elemento que **NO SEA MENOR** al elemento que se esta búscando.

Ejemplo

Busquemos la primer elemento que no sea menor al 11

1	2	5	9	10	10	12	20

Búsqueda Binaria - upper_bound

Definición

La posición del primer elemento que **SEA MAYOR** al elemento que se esta búscando.

Ejemplo

Busquemos la primer elemento que sea mayor al 10

1	2	5	9	10	10	12	20

Búsqueda Binaria - lower_bound

Definición

La posición del primer elemento que **SEA MAYOR** al elemento que se esta búscando.

Ejemplo

Busquemos la primer elemento que sea mayor al 10

1	2	5	9	10	10	12	20

Búsqueda Binaria - upper_bound

Definición

La posición del primer elemento que **SEA MAYOR** al elemento que se esta búscando.

Ejemplo

Busquemos la primer elemento que sea mayor al 11

1	2	5	9	10	10	12	20

Búsqueda Binaria - lower_bound

Definición

La posición del primer elemento que **SEA MAYOR** al elemento que se esta búscando.

Ejemplo

Busquemos la primer elemento que sea mayor al 11

1	2	5	9	10	10	12	20

Búsqueda Binaria - Librería Estándar

```cpp
int main() {
  vector< int > v;
  int x;
  // se asume que aqui se rellena el vector y la variable x
  // recordar que el vector/arreglo debe estar ordenado

  vector< int >::iterator it = lower_bound(v.begin(), v.end(), x);
  if(it != v.end()) {
    cout << "El primer elemento que no sea menor que " << x << ": " << "\n";
    cout << *it << "\n";
    int pos = it - v.begin();
    cout << "Posicion: " << pos << "\n";
  } else {
    cout << x << " es mayor a todos los numeros que hay en el vector" << "\n";
  }
  return 0
}
```

Búsqueda Binaria - Librería Estándar

```cpp
int main() {
  vector< int > v;
  int x;
  // se asume que aqui se rellena el vector y la variable x
  // recordar que el vector/arreglo debe estar ordenado

  vector< int >::iterator it = upper_bound(v.begin(), v.end(), x);
  if(it != v.end()) {
    cout << "El primer elemento que sea mayor que " << x << ": " << "\n";
    cout << *it << "\n";
    int pos = it - v.begin();
    cout << "Posicion: " << pos << "\n";
  } else {
    cout << x << " es mayor a todos los numeros que hay en el vector" << "\n";
  }
  return 0
}
```

TRUCASO

Problema

Dado una secuencia de números ordenada, ¿cuántas veces aparece el elemento x?

TRUCASO

Problema

Dado una secuencia de números ordenada, ¿cuántas veces aparece el elemento x?

Solución

- Utilizando `lower_bound` y `upper_bound`

Problema

Dado una secuencia de números ordenada, ¿cuántas veces aparece el elemento x?

Solución

- Utilizando `lower_bound` y `upper_bound`

¿Cuántas veces aparece el 10?

REFERENCES I