

Algoritmos básicos de vectores

Mg. Ing. Facundo S. Larosa
Ing. Ezequiel Gorandi

Informática I

Departamento de Ingeniería Electrónica

Ejemplo disparador

- Se desea hacer un programa que registre una serie de valores de temperatura y calcule el valor mínimo y máximo registrados y luego imprimirlos ordenados de menor a mayor...

¿Cómo hacemos?

Algoritmos básicos

- Búsqueda de valores extremos de un vector:
 - Máximo
 - Mínimo
- Ordenamiento de un vector:
 - Método de selección (*selection sort*)
 - Método de burbujeo (*bubble sort*)

Búsqueda de valores extremos de un vector

Los operadores relacionales ($<$, $>$, $<=$, $>=$) son binarios. Por lo tanto, para encontrar el valor máximo o mínimo de un vector, será necesario recorrerlo comparando de a dos elementos

Búsqueda de valores extremos de un vector

//Declaración de un vector


```
int vec[5]={3,2,5,-4,-2};
```

//Búsqueda del menor valor

1) Suponemos el primer valor como mínimo

2) Comparar el elemento siguiente del vector con el valor mínimo supuesto.

3) Si es menor, este es el nuevo valor mínimo supuesto.

vec[0]	3	 Es menor?
vec[1]	2	
vec[2]	5	
vec[3]	-4	
vec[4]	-2	

vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	-4
vec[4]	-2

Búsqueda de valores extremos de un vector

//Declaración de un vector


```
int vec[5]={3,2,5,-4,-2};
```

//Búsqueda del menor valor

4) Comparar el elemento siguiente del vector con el valor mínimo supuesto.

vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	-4
vec[4]	-2

Es menor?



5) Si es menor, este es el nuevo valor mínimo supuesto.

vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	-4
vec[4]	-2


Búsqueda de valores extremos de un vector

//Declaración de un vector

```
int vec[5]={3,2,5,-4,-2};
```

//Búsqueda del menor valor

6) Comparar el elemento siguiente del vector con el valor mínimo supuesto.

vec[0]	3	Es menor? 
vec[1]	2	
vec[2]	5	
vec[3]	-4	
vec[4]	-2	

7) Si es menor, este es el nuevo valor mínimo supuesto.

vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	-4
vec[4]	-2

Búsqueda de valores extremos de un vector

//Declaración de un vector

```
int vec[5]={3,2,5,-4,-2};
```


//Búsqueda del menor valor

8) Comparar el elemento siguiente del vector con el valor mínimo supuesto.

9) Si es menor, este es valor mínimo del vector.

vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	-4
vec[4]	-2

Es menor?



vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	-4
vec[4]	-2

Hemos encontrado el menor valor del vector!!

A programar

- 1) Ingrese 10 valores numéricos que representarán muestras de presión provenientes de un tanque presurizado. Luego, imprimir el mínimo y el máximo de la serie. Determinar si el promedio de los valores es más cercano al mínimo o al máximo y la diferencia. Obtener conclusiones sobre la posible utilización de este cálculo.
- 2) Para el ejercicio anterior, determinar la posición del valor máximo y del valor mínimo.

Métodos de ordenamiento

Existen una variedad de métodos de ordenamiento:

- Método de selección (*selection sort*)
- Método de burbujeo (*bubble sort*)
- Método por mezcla (*merge sort*)
- Método por montículos (*heap sort*)
- Método rápido (*quick sort*)
- Método Shell (*Shell sort*)
- Otros...

Métodos de ordenamiento básicos

Existen una variedad de métodos de ordenamiento:

- Método de selección (*selection sort*)
- Método de burbujeo (*bubble sort*)
- Método por mezcla (*merge sort*)
- Método por montículos (*heap sort*)
- Método rápido (*quick sort*)
- Método Shell (*Shell sort*)
- Otros...

Método de ordenamiento por selección

Este método consiste en buscar un valor extremo del vector (por ejemplo, el mínimo o el máximo) y ubicarlo en la posición adecuada de acuerdo al criterio de ordenamiento. Luego, se vuelve a realizar la misma operación sobre la fracción del vector restante.

Veámoslo con un ejemplo sencillo para luego inferir una metodología general...

Método de ordenamiento por selección

//Declaración de un vector "desordenado"

```
int vec[5]={3,2,5,1,-4};
```

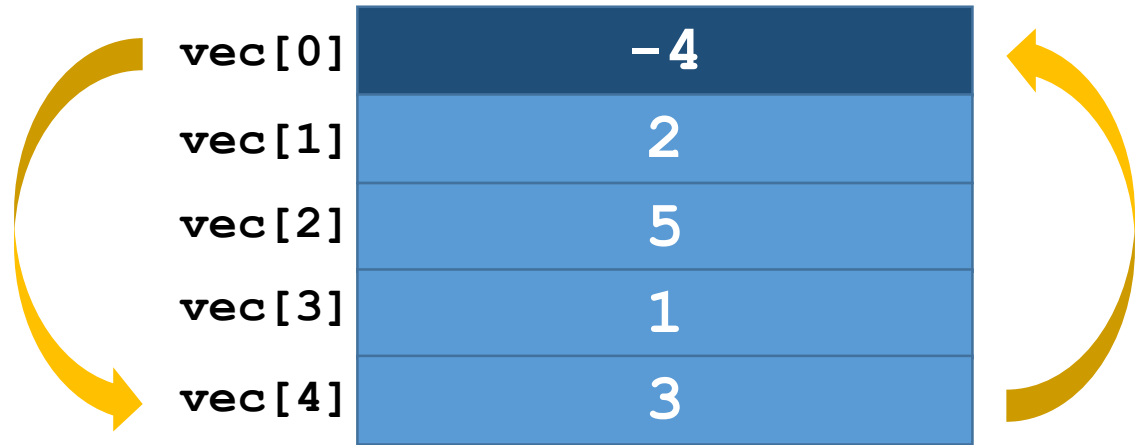
//Vamos a ordenar el vector en orden creciente (de menor a mayor)

vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	1
vec[4]	-4

Método de ordenamiento por selección

- 1) Buscar el valor mínimo* del vector
- 2) Intercambiar el valor mínimo por el del lugar que le corresponde

vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	1
vec[4]	-4



The diagram illustrates the second step of the selection sort algorithm. It shows two vertical arrays representing the state of the vector. The left array shows the initial state where the minimum value -4 is at index 4. The right array shows the result after swapping the element at index 0 (value 3) with the element at index 4 (value -4). Two curved yellow arrows indicate the swap: one from index 0 to index 4 and another from index 4 to index 0.

vec[0]	-4
vec[1]	2
vec[2]	5
vec[3]	1
vec[4]	3

* El valor extremo elegido podría ser el máximo o el mínimo indistintamente

Método de ordenamiento por selección

3) Buscar el valor mínimo de la fracción restante del vector

vec[0]	-4
vec[1]	2
vec[2]	5
vec[3]	1
vec[4]	3

4) Intercambiar el valor mínimo por el del lugar que le corresponde

vec[0]	-4
vec[1]	1
vec[2]	5
vec[3]	2
vec[4]	3

Método de ordenamiento por selección

5) Buscar el valor mínimo de la fracción restante del vector

vec[0]	-4
vec[1]	1
vec[2]	5
vec[3]	2
vec[4]	3

6) Intercambiar el valor mínimo por el del lugar que le corresponde

vec[0]	-4
vec[1]	1
vec[2]	2
vec[3]	5
vec[4]	3

Método de ordenamiento por selección

7) Buscar el valor mínimo de la fracción restante del vector

vec[0]	-4
vec[1]	1
vec[2]	2
vec[3]	5
vec[4]	3

8) Intercambiar el valor mínimo por el del lugar que le corresponde

vec[0]	-4
vec[1]	1
vec[2]	2
vec[3]	3
vec[4]	5

Con este último paso el ordenamiento ya está completado!

Método de ordenamiento por selección: Resumen

Iteración 0

3
2
5
1
-4

-4
2
5
1
3

Iteración 3

-4
1
2
5
3

-4
1
2
3
5

Iteración 1

-4
2
5
1
3

-4
1
5
2
3

Iteración 2

-4
1
5
2
3

-4
1
2
5
3

Complejidad del algoritmo

Para un vector de 'n' elementos se requieren:

- 'n-1' comparaciones en el primer paso (para "decantar" el mínimo o máximo entre 'n' variables)
- 'n-2' comparaciones en el segundo paso (para "decantar" el mínimo o máximo entre 'n - 1' variables)
-

En definitiva se requiere una cantidad de comparaciones igual a:

$$(n-1) + (n-2) + \dots + 1 = \sum_{i=1}^{n-1} i = \sum_{i=1}^{n-1} \binom{i}{1}$$
$$\sum_{i=1}^{n-1} \binom{i}{1} = \binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{1}{2}n(n-1) = \frac{1}{2}(n^2 - n)$$

Método de ordenamiento por selección: Implementación

*/*Vamos a implementar una función a la que se le pasa un vector de enteros Junto con su tamaño y nos devuelve el vector ordenado*/*

```
void selSort (int * vec, int n)
{
```



```
}
```


Método de ordenamiento por selección: Implementación

*/*Vamos a implementar una función a la que se le pasa un vector de enteros junto con su tamaño y nos devuelve el vector ordenado*/*

```
void selSort (int * vec, int n)
{
```

```
    int i,j,posMin,aux;
```

```
    for(i=0;i<n-1;i++)
    {
```

**Búsqueda
del
mínimo**

```
        posMin=i;
        for(j=i+1;j<n;j++)
        {
            if(vec[j]<vec[posMin])
                posMin=j;
        }
```

```
        aux=vec[i];
        vec[i]=vec[posMin];
        vec[posMin]=aux;
```

Intercambio (swap) de variables

```
    }
```

```
}
```

Método de ordenamiento por burbujeo

//Declaración de un vector "desordenado"

```
int vec[5]={3,2,5,1,-4};
```

//Vamos a ordenar el vector en orden creciente
//(de menor a mayor)

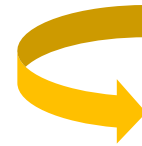
vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	1
vec[4]	-4



Método de ordenamiento por burbujeo

- 1) Comparar dos variables adyacentes 2) Intercambiar los valores si el orden no es el que corresponde

vec[0]	3
vec[1]	2
vec[2]	5
vec[3]	1
vec[4]	-4

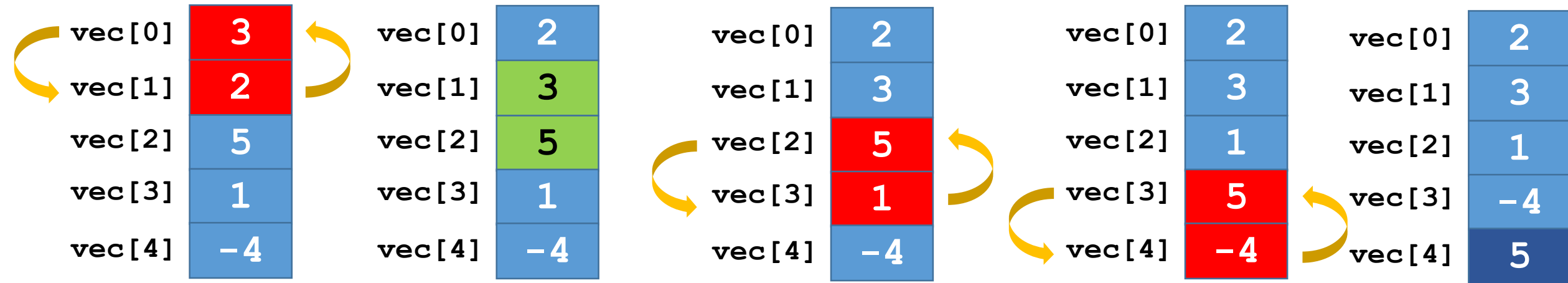


vec[0]	2
vec[1]	3
vec[2]	5
vec[3]	1
vec[4]	-4



Método de ordenamiento por burbujeo

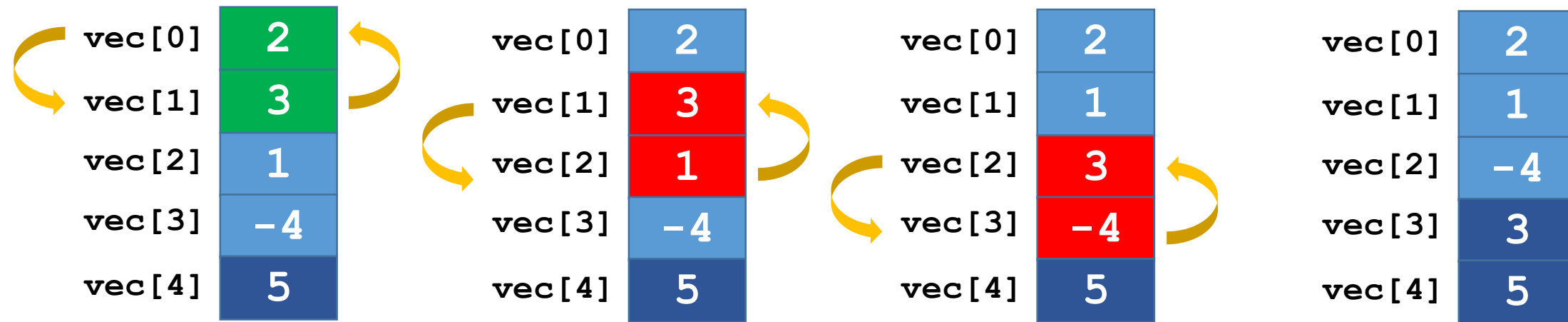
3) El procedimiento se repite hasta llegar al final del vector



Al finalizar la primera iteración, el último elemento del vector está ordenado

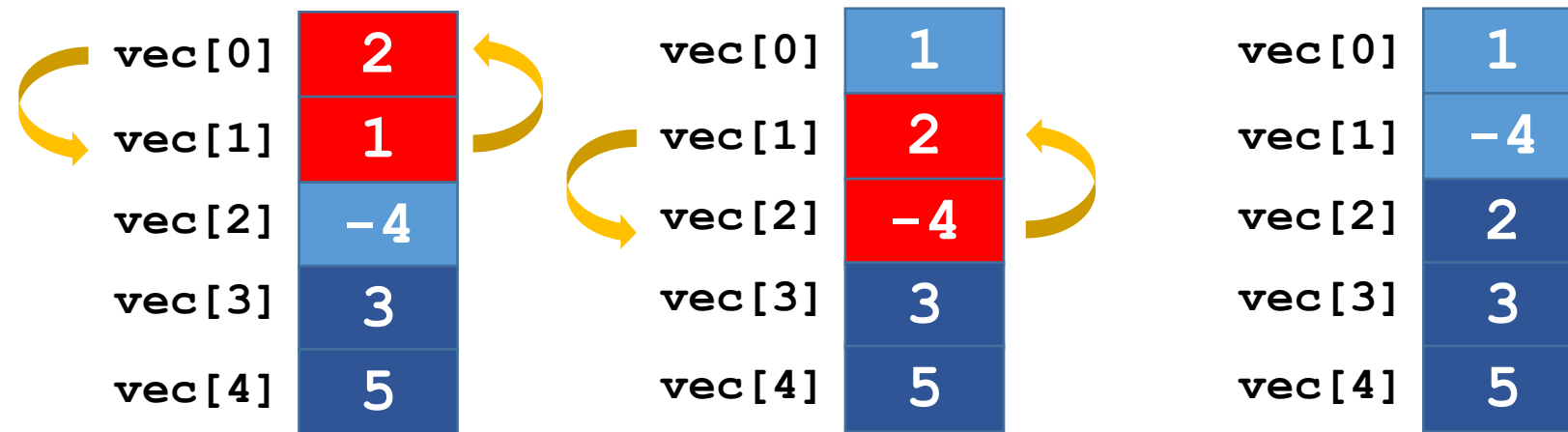
Método de ordenamiento por burbujeo

4) El procedimiento se realiza nuevamente sobre la fracción del vector no ordenada hasta que se completa el ordenamiento...



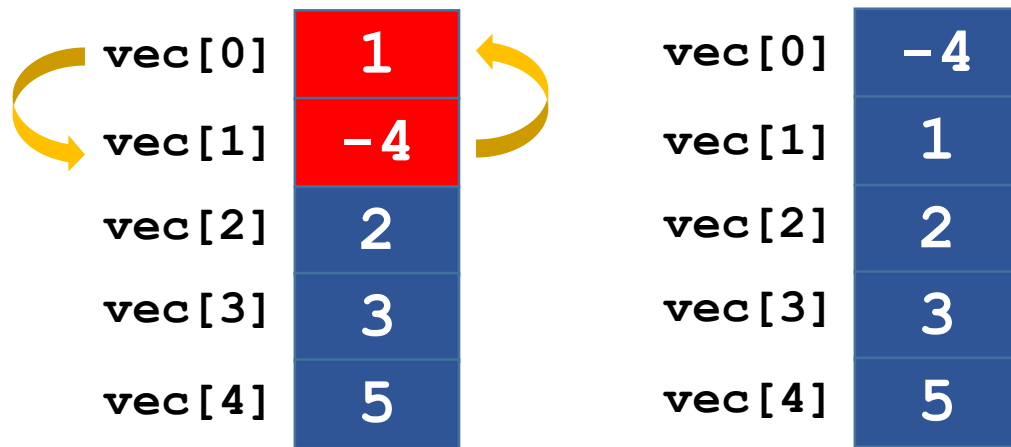
Método de ordenamiento por burbujeo

4) El procedimiento se realiza nuevamente sobre la fracción del vector no ordenada hasta que se completa el ordenamiento...



Método de ordenamiento por burbujeo

4) El procedimiento se realiza nuevamente sobre la fracción del vector no ordenada hasta que se completa el ordenamiento...



Al completar la última iteración el vector está ordenado

Método de ordenamiento por burbujeo : Resumen

Iteración 0

3
2
5
1
-4

2
3
5
1
-4

2
3
5
1
-4

2
3
1
5
-4

2
3
1
-4
5

Iteración 3

1
-4
2
3
5

-4
1
2
3
5

Iteración 1

2
3
1
-4
5

2
3
1
-4
5

2
1
3
-4
5

2
1
-4
3
5

Iteración 2

2
1
-4
3
5

1
2
-4
3
5

1
-4
2
3
5

Método de ordenamiento por burbujeo: Implementación

*/*Vamos a implementar una función a la que se le pasa un vector de enteros junto con su tamaño y nos devuelve el vector ordenado*/*

```
void bubbleSort (int * vec, int n)
{
```



```
}
```

Método de ordenamiento por burbujeo: Implementación

*/*Vamos a implementar una función a la que se le pasa un vector de enteros junto con su tamaño y nos devuelve el vector ordenado*/*

```
void bubbleSort (int * vec, int n)
{
    int i,j,aux;

    for(i=0;i<n-1;i++)
        for(j=0;j<n-1-i;j++)
            if(vec[j]>vec[j+1])
            {
                aux=vec[j];
                vec[j]=vec[j+1];
                vec[j+1]=aux;
            }
}
```

Intercambio (swap) de variables