Niñas PRO(grama doras)

Algoritmos simples de ordenamiento

Niñas PRO(grama doras)

Temario

- ★ Arreglos ordenados
- ★ Algoritmos de ordenamiento
- ★ Análisis de costos

Arreglos Ordenados

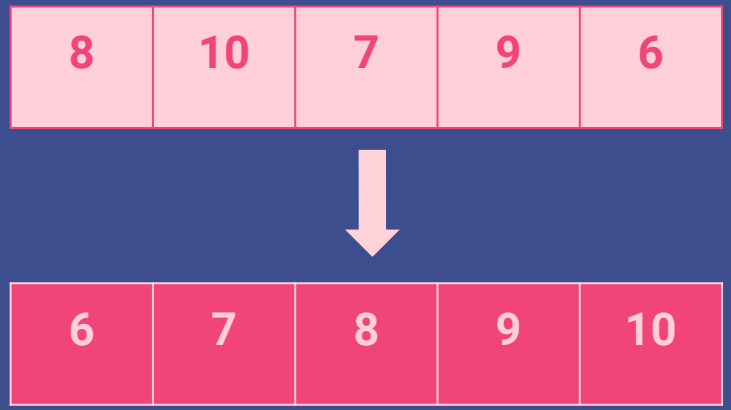
Un arreglo ordenado tiene todos sus elementos en un **orden particular**. Por ejemplo:

arreglo en orden alfabético

arreglo de números enteros en orden **descendente**

Algoritmos de Ordenamiento

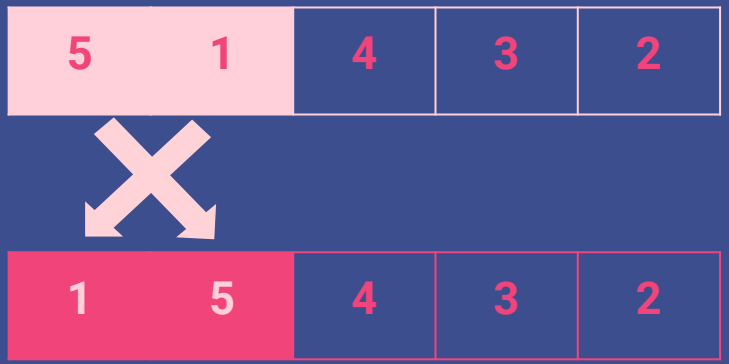
Los algoritmos de ordenamiento reciben un arreglo y devuelven otro con los elementos ordenados, en base a algún **criterio de ordenamiento**.



Bubble Sort

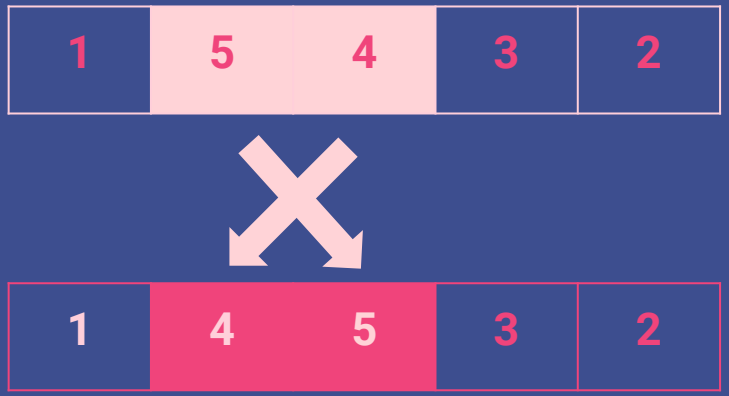
Se basa en comparar cada elemento del arreglo con el siguiente y cambiarlos de lugar si están desordenados.

Ejemplo: ordenaremos un arreglo de forma ascendente, para esto comenzamos con el primer par de elementos del arreglo:



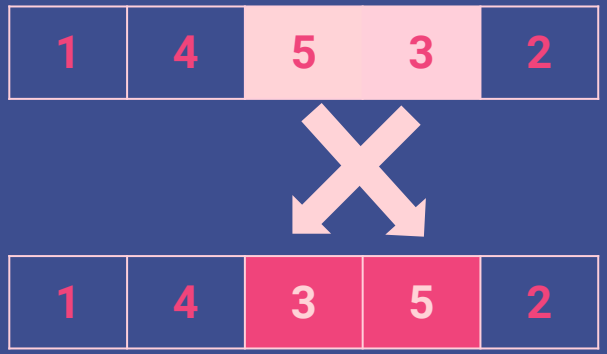
Bubble Sort

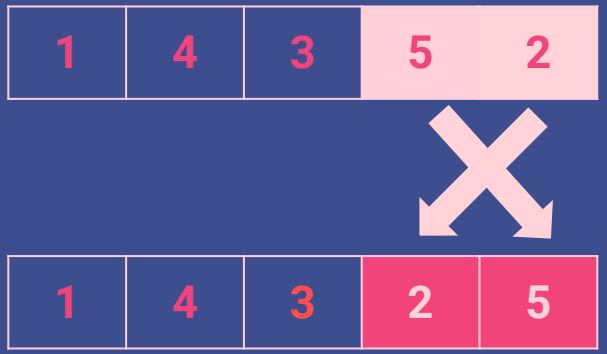
Continuamos comparando los siguientes dos elementos, como el de la izquierda es mayor, los cambiamos de posición:



Bubble Sort

De esa forma, vamos sucesivamente comparando y cambiando de lugar los elementos cuando están desordenados:





Actividad

Indica cómo se irá ordenando el arreglo en la siguiente iteración.

Niñas PRO(grama doras)

Código Bubble Sort


```
using namespace std;
int main()
{
	int n = 5; //tamaño del arreglo
	int array[n] = {5,1,4,3,2};
	for (int i = 0 ; i <  n - 1 ; i++)
	{
		for (int j = 0 ; j < n - i - 1; j++)
		{
			if (array[j] > array[j+1]) // Para orden descendente usamos <
			{
				int swap     = array[j];
				array[j]   = array[j+1];
				array[j+1] = swap;
			}
		}
	}

  return 0;
}
```

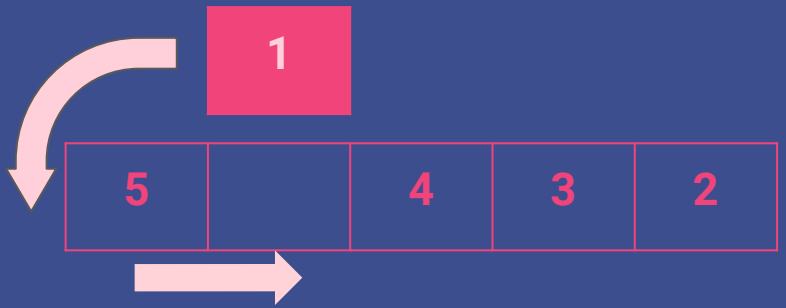
Insert Sort

Inserta cada elemento, empezando por el segundo, en la posición correcta, recorriendo los elementos de mayor orden hacia la derecha.

Ejemplo:

5	1	4	3	2

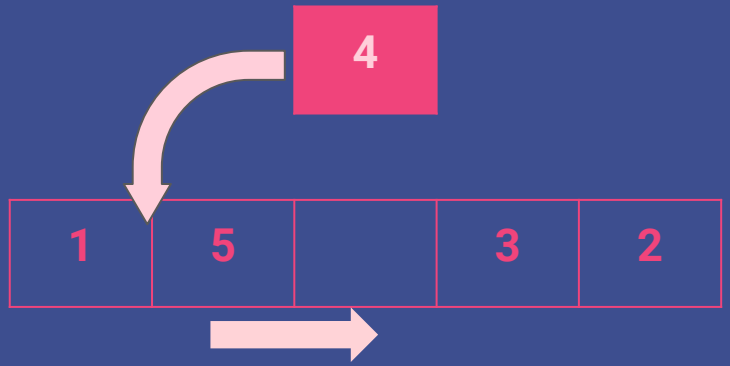
Buscamos la posición a la izquierda del número 1, como es menor que 5, lo colocamos en la primera posición y recorremos el 5 a la derecha:



1	5	4	3	2

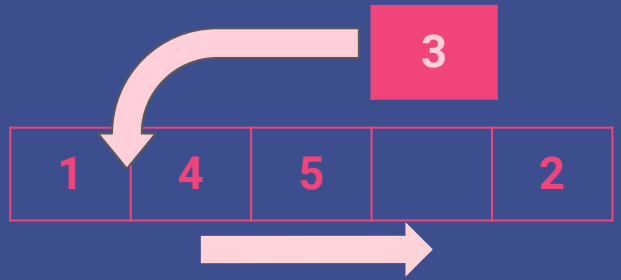
Insert Sort

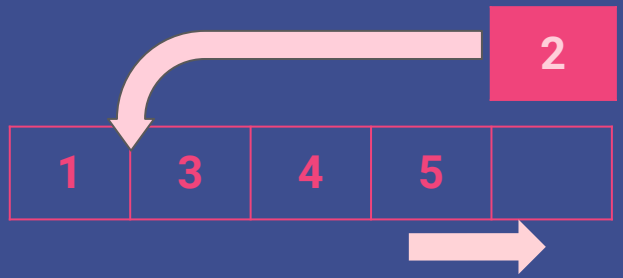
Luego, buscamos la posición del número 3 a la izquierda, corresponde colocarlo entre el 1 y el 5, así que recorremos el 5 a la derecha.



Insert Sort

Hacemos lo mismo para el 2 y el 4:





Código Insert Sort


```
using namespace std;
int main()
{
	int n = 5; //tamaño del arreglo
	int array[n] = {5,1,4,3,2};

	for(int i = 1; i < n; ++i){
		int j = i - 1;
		int elementoActual = array[i];
		while(j >= 0 && array[j] > elementoActual){
			array[j+1] = array[j];
			--j;
		}
		array[j+1] = elementoActual;
	}

  return 0;
}
```

Actividad

Indica cómo se irá ordenando el arreglo con el algoritmo Insert Sort.

Niñas PRO(grama doras)

Select Sort

Consiste en buscar el elemento de menor orden en la lista e intercambiarlo por el elemento en la posición 0, luego buscar el siguiente elemento de menor orden e intercambiarlo por el elemento en la posición 1 y así sucesivamente. De manera general:

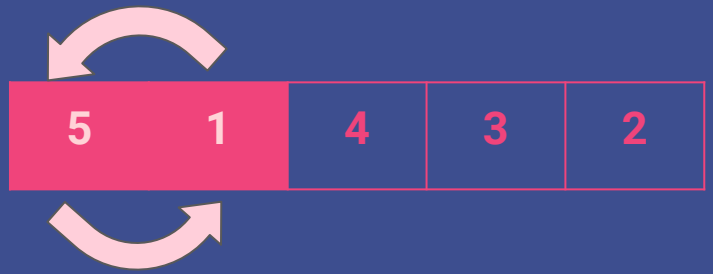
- Buscar el elemento de menor orden entre una posición *i* y el final del arreglo.
- Intercambiar el elemento de menor orden con el elemento en la posición *i* del arreglo.

Select Sort

Ejemplo:

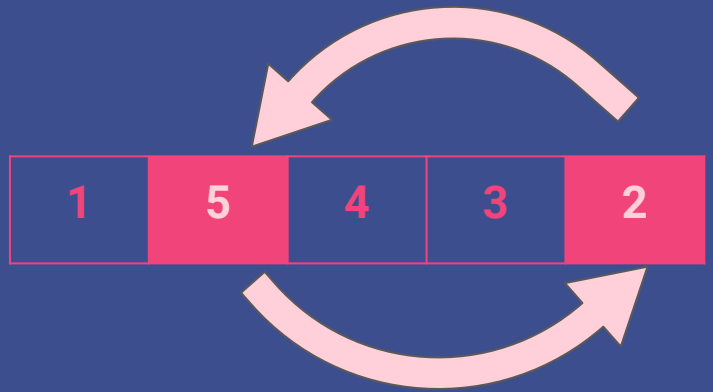
5	1	4	3	2

Buscamos el elemento de menor orden, que es el 1 y lo intercambiamos por el elemento en la primera posición del arreglo:



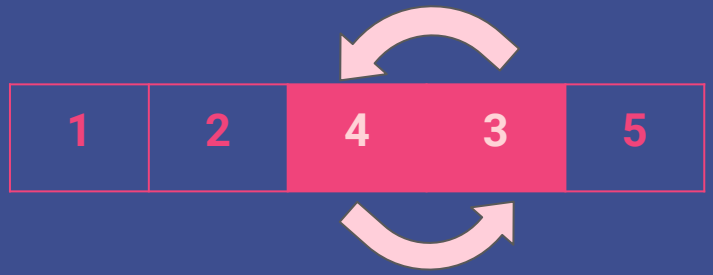
Select Sort

Buscamos el siguiente elemento de menor orden, que es el 2 y lo intercambiamos por el elemento en la segunda posición del arreglo:



Select Sort

Luego seguimos con el siguiente elemento de menor orden, que es el 3 y lo intercambiamos por el elemento en la tercera posición del arreglo:



Actividad

Indica cómo se irá ordenando el arreglo con el algoritmo Insert Sort.

10	20	0	40	30

Niñas PRO(grama doras)

Código Select Sort


```
using namespace std;

int main()
{
  int n = 5;               //tamaño del arreglo
  int array[n] = { 5,1,4,3,2 };
  int pos_min, temp;

  for (int i = 0; i < n - 1; i++)
  {    //inicialmente guardamos la posicion actual del arreglo en pos_min
       pos_min = i;

       for (int j = i + 1; j < n; j++)
       {
          if (array[j] < array[pos_min])
           //en pos_min vamos guardando la posicion del elemento de menor orden
           pos_min = j;
       }
```

Código Select Sort


```
                //si la posicion minima es distinta de la posicion actual, es porque
                //se ha encontrado un valor menor y se debe intercambiar posiciones
                if (pos_min != i)
                {
                        temp = array[i];
                        array[i] = array[pos_min];
                        array[pos_min] = temp;
                }
        }
}
```

Análisis de Costo

En el cuadro tenemos una comparación de los costos de los algoritmos que vimos hoy. La cantidad de veces que se ejecuta cada algoritmo de ordenamiento es variable, por esta razón se comparan el caso promedio, el mejor y el peor caso.

Algoritmo	Mejor caso	Peor caso	Caso promedio
`Bubble sort`	$O(n)$	$O(n^2)$	$O(n^2)$
`Insert sort`	$O(n)$	$O(n^2)$	$O(n^2)$
`Select sort`	$O(n^2)$	$O(n^2)$	$O(n^2)$

Niñas PRO(grama doras)

¿Qué veremos la próxima clase?

- ★ Funciones de ordenación
- ★ Librerías que permiten ordenar los tipos de datos básicos

Línea Gráfica

Lorena Gonzalez - Diseñadora
@soygonzaleztambien