Fundamentos de la programación

8 Algoritmos de ordenación

Doble Grado en Matemáticas e Informática
Facultad de Informática
Universidad Complutense

Isabel Pita

(Adaptadas de los originales de Luis Hernández, Ana Gil y Alberto Verdejo)

Ordenación por inserción

Podemos utilizar *plantillas*:

```
template <typename T>
void ordenarInsercion(std::vector<T> & array) {
   size_t N = array.size();
   // parte ordenada array[0..i), parte por procesar array[i..N)
   for (size_t i = 1; i < N; ++i) { // desde el segundo hasta el último
      // parte ordenada array[0..i)
      T elemento = array[i];  // elemento a insertar
      size_t j = i; // desplazar los mayores de la parte ordenada
      while (j > 0 && elemento < array[j - 1]) {
         array[j] = array[j - 1];
         --j;
      }
      if (j != i) array[j] = elemento; // colocar en el hueco
   } // parte ordenada array[0..N)
}
```

Ordenación por selección

```
template <typename T, typename Comp = less<T>>
void ordenarSeleccion(std::vector<T> & array, Comp ord = Comp()) {
    size_t N = array.size();
    // ordenado array[0..i) y todos los elementos de la parte ordenada
    // son "menores" (o iguales) que los de la parte no ordenada
    for (size_t i = 0; i < N - 1; ++i) { // hasta el penúltimo
        // colocar el "menor" de la parte no ordenada (array[i..N))
        // en array[i]
        size_t menor = i;
        for (size_t j = i + 1; j < N; ++j)
            if (ord(array[j], array[menor])) menor = j;
        if (i < menor) {  // intercambiarlo con array[i]
            swap(array[i], array[menor]);
        }
    } // parte ordenada array[0..N)
}
```

Método de la burbuja

```
template <typename T, typename Comp = less<T>>
void ordenarBurbuja(std::vector<T> & array, Comp ord = Comp()) {
    size_t N = array.size();
    bool inter = true; size_t i = 0;
    // ordenado array[0..i) y todos los elementos de la parte
    // ordenada son "menores" (o iguales) que los de la parte no ordenada
    while (i < N - 1 && inter) {  // hasta el penúltimo
        // colocar el menor de la parte no ordenada en array[i]
        // intercambiando, desde el último, con los elementos mayores
        inter = false; // y comprobando si se realiza algún intercambio
        for (size_t j = N - 1; j > i; --j)
            if (ord(array[j], array[j - 1])) {
                swap(array[j], array[j - 1]);
                inter = true;
            }
        ++i;
    }
}
```

$O(N^2)$, Estable y Comportamiento natural.