INSERTION SORT

WAGNER EMANOEL COSTA

INSERTION SORT

Algoritmo 17 Insertion Sort(A)

```
1: para j ← 2 até Comprimento[A] faça
2:     chave ← A[j]
3:     i ← j − 1
4:     enquanto i > 0 e A[i] > chave faça
5:         A[i + 1] ← A[i]
6:         i ← i − 1
7:     fim enquanto
8:     A[i + 1] ← chave
9: fim para
```

INSERTION SORT

```
import java.util.Comparator;
```

```
public class Insertion {

    // This class should not be instantiated.
    private Insertion() { }
```

INSERTION SORT

```
// is v < w ?
private static boolean less(Comparable v, Comparable w) {
    return v.compareTo(w) < 0;
}
```

```
// is v < w ?
private static boolean less(Object v, Object w, Comparator comparator) {
    return comparator.compare(v, w) < 0;
}
```

```
// exchange a[i] and a[j]
private static void exch(Object[] a, int i, int j) {
    Object swap = a[i];
    a[i] = a[j];
    a[j] = swap;
}
```

INSERTION SORT

```
public static void sort(Comparable[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        for (int j = i; j > 0 && less(a[j], a[j-1]); j--) {
            exch(a, j, j-1);
        }
        assert isSorted(a, 0, i);
    }
    assert isSorted(a);
}
```

INSERTION SORT

```
public static void sort(Comparable[] a, int lo, int hi) {
    for (int i = lo; i <= hi; i++) {
        for (int j = i; j > lo && less(a[j], a[j-1]); j--) {
            exch(a, j, j-1);

        }
    }
    assert isSorted(a, lo, hi);
}
```

INSERTION SORT

```
public static void sort(Object[] a, Comparator comparator) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        for (int j = i; j > 0 && less(a[j], a[j-1], comparator); j--) {
            exch(a, j, j-1);
        }
        assert isSorted(a, 0, i, comparator);
    }
    assert isSorted(a, comparator);
}
```

INSERTION SORT

```
public static void sort(Object[] a, int lo, int hi, Comparator comparator) {
    for (int i = lo; i <= hi; i++) {
        for (int j = i; j > lo && less(a[j], a[j-1], comparator); j--) {
            exch(a, j, j-1);
        }
    }
    assert isSorted(a, lo, hi, comparator);
}
```

INSERTION SORT

```
public static int[] indexSort(Comparable[] a) {
    int N = a.length;
    int[] index = new int[N];
    for (int i = 0; i < N; i++)
        index[i] = i;

    for (int i = 0; i < N; i++)
        for (int j = i; j > 0 && less(a[index[j]], a[index[j-1]]); j--)
            exch(index, j, j-1);

    return index;
}
```

INSERTION SORT

```
// exchange a[i] and a[j]  (for indirect sort)
private static void exch(int[] a, int i, int j) {
    int swap = a[i];
    a[i] = a[j];
    a[j] = swap;
}
```

```
private static boolean isSorted(Comparable[] a) {
    return isSorted(a, 0, a.length - 1);
}
```

```
// is the array sorted from a[lo] to a[hi]
private static boolean isSorted(Comparable[] a, int lo, int hi) {
    for (int i = lo+1; i <= hi; i++)
        if (less(a[i], a[i-1])) return false;
    return true;
}
```

INSERTION SORT

```
private static boolean isSorted(Object[] a, Comparator comparator) {
    return isSorted(a, 0, a.length - 1, comparator);
}
```

```
// is the array sorted from a[lo] to a[hi]
private static boolean isSorted(Object[] a, int lo, int hi, Comparator comparator) {
    for (int i = lo + 1; i <= hi; i++)
        if (less(a[i], a[i-1], comparator)) return false;
    return true;
}
```

```
// print array to standard output
private static void show(Comparable[] a) {
    for (int i = 0; i < a.length; i++) {
        StdOut.println(a[i]);
    }
}
```

INSERTION SORT

```
public static void main(String[] args) {
    String[] a = StdIn.readAllStrings();
    Insertion.sort(a);
    show(a);
}
```

DÚVIDAS? PERGUNTAS?

SELECTION SORT

WAGNER EMANOEL COSTA

SELECTION SORT

```
import java.util.Comparator;
```

```
   */
 public class Selection {

      // This class should not be instantiated.
      private Selection() { }
```

```
public static void sort(Comparable[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        int min = i;
        for (int j = i+1; j < N; j++) {
            if (less(a[j], a[min])) min = j;
        }
        exch(a, i, min);
        assert isSorted(a, 0, i);
    }
    assert isSorted(a);
}
```

SELECTION SORT

```
public static void sort(Object[] a, Comparator c) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        int min = i;
        for (int j = i+1; j < N; j++) {
            if (less(c, a[j], a[min])) min = j;
        }
        exch(a, i, min);
        assert isSorted(a, c, 0, i);

    }
    assert isSorted(a, c);
}
```

SELECTION SORT

```
// is v < w ?
private static boolean less(Comparable v, Comparable w) {
    return v.compareTo(w) < 0;
}
```

```
// is v < w ?
private static boolean less(Comparator c, Object v, Object w) {
    return c.compare(v, w) < 0;
}
```

```
// exchange a[i] and a[j]
private static void exch(Object[] a, int i, int j) {
    Object swap = a[i];
    a[i] = a[j];
    a[j] = swap;
}
```

SELECTION SORT

```
// is the array a[] sorted?
private static boolean isSorted(Comparable[] a) {
    return isSorted(a, 0, a.length - 1);
}

// is the array sorted from a[lo] to a[hi]
private static boolean isSorted(Comparable[] a, int lo, int hi) {
    for (int i = lo + 1; i <= hi; i++)
        if (less(a[i], a[i-1])) return false;
    return true;
}

// is the array a[] sorted?
private static boolean isSorted(Object[] a, Comparator c) {
    return isSorted(a, c, 0, a.length - 1);
}

// is the array sorted from a[lo] to a[hi]
private static boolean isSorted(Object[] a, Comparator c, int lo, int hi) {
    for (int i = lo + 1; i <= hi; i++)
        if (less(c, a[i], a[i-1])) return false;
    return true;
}
```

SELECTION SORT

```
// print array to standard output
private static void show(Comparable[] a) {
    for (int i = 0; i < a.length; i++) {
        StdOut.println(a[i]);
    }
}
```

```
    public static void main(String[] args) {
        String[] a = StdIn.readAllStrings();
        Selection.sort(a);
        show(a);
    }
}
```

DÚVIDAS? PERGUNTAS?

UM ÓTIMO DIA A TODOS