



Práctica 6

1. Implemente `qsort_`, siguiendo el algoritmo de ordenamiento rápido visto en clase. Determine si el algoritmo es estable.
 2. Re-implemente la función `particionar`, de manera que haya un solo bucle en el código del programa. Ayuda: puede revisar el artículo de Quicksort en Wikipedia en español.
 3. Busque cuál es la función `qsort` disponible en C. Vea su tipo, y explique las diferencias con el `qsort_` implementado en el ejercicio 1. Analice las diferencias con la generalidad vista en la práctica anterior.
 4. Modifique `qsort_` con las siguientes variantes de elección de pivot:
 - a) De pivot se elige aleatoriamente un elemento arreglo.
 - b) De pivot se elige el último elemento del arreglo.
 - c) De pivot se elige el elemento medio del arreglo.
 - d) De pivot se elige la mediana entre el primer elemento, el medio y el último.
- Agregue a la función `qsort_` un parámetro que permita decidir cuál elección de pivot utilizar.
5. Modifique `qsort_` de manera de llevar la cantidad de llamadas que se produjeron a la función (ya sea desde otras funciones, como llamadas generadas por la recursión). Ver la diferencia de cantidad de llamadas entre diferentes elecciones de pivot y diferentes arreglos a ordenar.
 6. Implemente `hsort` siguiendo el algoritmo de ordenamiento por heap.
 7. Implemente una función `void heapify(int *, int)` que transforma en max-heap un arreglo de manera in-situ.
 8. Implemente una función `void swapsift(int *, int)` que dado un max-heap intercambie el elemento mayor del heap con el último elemento del arreglo, y luego modifique el arreglo desde el primer elemento hasta el penúltimo para que vuelva a ser un maxheap.
 9. Implemente `hsort` utilizando las funciones `heapify` y `swapsift`. ¿Que ventajas trae esto? ¿Qué desventajas?
 - 10*. En la página 5 del paper “Introspective sorting and selection algorithms”¹, está la descripción del algoritmo introsort. Este algoritmo combina quicksort, heapsort e insertionsort. En esencia funciona como quicksort, pero evita los casos malos mediante heapsort. Actualmente es la implementación de `Array.Sort` en el framework .NET². Escriba una implementación de este algoritmo en C.

¹<http://www.cs.rpi.edu/~musser/gp/introsort.ps>

²<http://msdn.microsoft.com/en-us/library/6tff1f0bc%28v=vs.110%29.aspx>