Auxiliar 9 - Ordenamiento Avanzado

Profesores: Jeremy Barbay
Patricio Poblete
Auxiliares: Daniela Campos, Cristóbal Muñoz
Sven Reisenegger, Bernardo Subercaseaux

P1. HeapSort

- Cómo ordenar un arreglo de datos utilizando un heap?
- Cuál es el coste de este algoritmo?
- Cuáles son sus ventajas y desventajas?

Solución:

- Se insertan todos los valores en el heap, luego se extrae el mínimo/máximo n veces y se va llenando el arreglo de manera ordenada.
- insertar(x) y extraerMin() tienen coste $log(n)$, por lo que en total se hacen $2 * nlog(n)$ operaciones, por lo que este algoritmo tiene mejor, promedio y peor caso $o(nlog(n))$.
- Una ventaja es que es muy simple de implementar (teniendo un Heap),además tiene el mejor orden posible para un algoritmo de ordenamiento basado en comparaciones. Sus desventajas son que necesita un Heap, y que no es estable al ordenar elementos.

P2. Tornillos y Tuercas:

Se quiere ordenar dos conjuntos de n tuercas y n tornillos de distintos tamaños, donde cada tornillo tiene una tuerca correspondiente. Considere un modelo en el que sólo se puede comparar un tornillo con una tuerca, es decir, no se pueden comparar tornillos entre si ni tuercas entre si. El resultado de cada comparación puede ser que el tornillo es mas grande que la tuerca, mas pequeño o igual.

Describa un algoritmo que ordena ambos conjuntos, de menor a mayor y que funcione dentro de $O(n^2)$ en el peor caso, pero dentro de $O(n \lg n)$ en promedio sobre la aleatoriedad del algoritmo.

Solución:

Para ordenar el conjunto se utilizará quicksort, modificado ligeramente para realizar las comparaciones cruzadas. No se muestra el método particionar ya que este no cambia.

```
void sort(int[] tuercas, int[] pernos, int inicio, int fin) {
    // particionamos los pernos usando de pivote la primera tuerca
    int pivote = particionar(tuercas[pivote], pernos, inicio, fin);

    // retorna el índice del perno que es igual a la tuerca
    // y lo usamos de pivote en las tuercas
    particionar(pernos[pivote], tuercas, inicio, fin);

    // ahora los 2 arreglos están divididos en 3 partes
    // con los pivotes al medio en el mismo índice, se sigue recursivamente
    sort(tuercas, pernos, inicio, pivote);
    sort(tuercas, pernos, pivote, fin);
}
```

P3. Radix Sort

Ordene la siguiente lista de palabras usando el método de bucket sort (radix sort). En cada pasada, indique por cuál columna se está clasificando y muestre el resultado de ese proceso:

- casas
- rema
- casa
- c
- terma
- hola
- casan
- tema
- beyblade
- masa

Solución:

Se agregan caracteres vacíos a la derecha de todas las palabras para que sean del mismo largo, se considera este caracter menor que todos los otros, ya que casa es menor que casas (por lo menos en el diccionario). Se ordena de derecha izquierda ya que el primer caracter es el más significativo lexicográficamente.

original	$8^a/7^a/6^a$ columna	5^a columna	4^a columna	3^a columna	2^a columna	1^a columna
casas	casas	rema	c	c	c	beyblade
rema	rema	casa	rema	hola	casa	c
casa	casa	c	casa	rema	masa	casa
c	c	hola	hola	tema	casan	casan
terma	terma	tema	tema	terma	casas	casas
hola	hola	masa	masa	casa	rema	hola
casan	casan	terma	casan	masa	tema	masa
tema	tema	beyblade	casas	casan	terma	rema
beyblade	masa	casan	beyblade	casas	beyblade	tema
masa	beyblade	casas	terma	beyblade	hola	terma