Guía de Ejercicios 5: Ordenamiento

[Version: 17 de mayo de 2024]

Objetivos:

- Introducir algoritmos de ordenamiento eficientes y entrenar su uso en el contexto de distintos problemas.

Ordenamiento

Ejercicio 1. Dados los algoritmos de ordenamiento selection sort e insertion sort.

(a) Implementarlos en C++ respetando los siguientes encabezados:

- `void selection_sort(vector<int> & v)`
- `void insertion_sort(vector<int> & v)`

(b) Para cada uno escribir el invariante del ciclo externo.

Ejercicio 2. Merge sort.

(a) Implementar en C++ una versión de mergesort que, en lugar de devolver el vector ordenado, modifique y ordene el vector que se pasa por referencia. El encabezado de la función debe ser: `void mergesort(vector<int> & v, int d, int h)`.

(b) Implementar una modificación de mergesort que devuelva la cantidad de inversiones del vector v.

Una inversión es un par de posiciones i, j del vector, tal que $i < j$ y $v[i] > v[j]$. Da una medida de cuán desordenado está el vector.

Por ejemplo:

- `{1,3,5,8,10,12}` tiene 0 elementos inversiones,
- `{1,3,12,5,8,10}` tiene 3 inversiones (los pares de posiciones $(2,3), (2,4), (2,5)$),
- `{12,10,8,5,3,1}` tiene 15 inversiones (todos los pares $i < j$ son inversiones).

El algoritmo debe mantener la complejidad temporal $O(n \log n)$.

Idea: la función auxiliar `merge` debe contar la cantidad de inversiones entre los elementos de una mitad contra los elementos de la otra mitad.

Ejercicio 3. Implementar la función `vector<int> mergear_muchos(vector<vector<int>> vs)` que toma por parámetro un vector de vectores de `int`. Se tiene como precondición que cada vector `vs[i]` está ordenado de forma creciente. Se debe devolver un vector que contenga la unión ordenada de todos los vectores de `vs`.

Ejemplo:

- si `vs = {{1,2,5,9},{6,8,10,11},{-1,6,9}}`
- se debe devolver `{-1,1,2,5,6,6,8,9,9,10,11}`.

Ejercicio 4. Implementar una modificación de Quicksort que divida el vector en tres partes en lugar de dos. Para eso, la función `dividir` debe elegir dos pivots p y q, con $p \leq q$, y ver para cada elemento e del vector cuál de las siguientes condiciones cumple:

- $e \leq p$,
- $p < e \leq q$,
- $q < e$.

Ejercicio 5. Dar un algoritmo para ordenar alfabéticamente un `string` sabiendo que sólo contiene letras minúsculas (sin tíldes ni eñe). El algoritmo debe tener una complejidad temporal en peor caso de $O(n)$ donde n es el tamaño del string.