Guía de Ejercicios 7: Complejidad algorítmica

Objetivos:

- Introducir los conceptos fundamentales de complejidad algorítmica (siempre en peor caso).
- Trabajar con programas de las clases de complejidad logarítmica, lineal y cuadrática.

Ejercicio 1. Se tiene un mazo de M naipes. Escribir un algoritmo que determine si el mazo está ordenado (de menor a mayor) o no. Calcular su complejidad algorítmica (en peor caso).

Ejercicio 2. Se tienen N cofres cerrados con llave y M llaves, con $M > N$. Se sabe que una y solo una llave abre cada cofre, y las restantes llaves son inútiles (no abren ningún cofre). Escribir un algoritmo para abrir todos los cofres. Calcular su complejidad algorítmica (en peor caso).

Ejercicio 3. Se tienen N cofres y dentro de cada cofre hay un mazo de M naipes. Se sabe que en uno y solo uno de los cofres el mazo que contiene está ordenado (de menor a mayor). Escribir un algoritmo que determine cuál es el cofre que tiene el mazo ordenado. Calcular su complejidad algorítmica (en peor caso).

Ejercicio 4. Se tienen N cofres cerrados con llave y M llaves. Se sabe que una y solo una de las M llaves sirve para abrir *todos* los cofres; las llaves restantes son inútiles (no abren ningún cofre). Escribir un algoritmo para determinar cuál es la llave útil. Calcular su complejidad algorítmica (en peor caso).

Ejercicio 5. Estimar el orden de complejidad algorítmica (en peor caso) de los programas escritos en las guías anteriores para los siguientes problemas:

(a) Dado un string, devolver su inversa (Guía 2, Ejercicio 4(f)).

(b) Dados dos strings, devolver la cantidad de veces que el primero está contenido en el segundo (Guía 2, Ejercicio 4(g)).

(c) Dado un número natural `n`, construir una lista con los primeros `n` números naturales impares, ordenados de menor a mayor (Guía 5, Ejercicio 3(d)).

(d) Dada una lista de enteros `l` y un entero `n`, construir una nueva lista, resultante de sumarle `n` a cada elemento de `l` (Guía 5, Ejercicio 3(e)).

(e) Dada una lista de strings, contar la cantidad total de caracteres (Guía 5, Ejercicio 3(f)).

(f) Dado un string `txt` de longitud arbitraria y un string `sep` de un carácter de longitud, separar `txt` en cada aparición de `sep`, construyendo así una nueva lista (Guía 5, Ejercicio 3(h)).

Ejercicio 6. Considerar esta variante del algoritmo SELECTIONSORT visto en clase:

> Para cada `i` entre `len(A)` y 1 (inclusive):
>
> Buscar el mayor elemento en `A[:i]`.
>
> Intercambiarlo con `A[i-1]`.

(a) Pensar un predicado invariante que describa el comportamiento del ciclo principal.

(b) Implementar el algoritmo en Python.

(c) Mostrar que tiene complejidad $O(\texttt{len(A)}^2)$.

Ejercicio 7.

(a) El algoritmo BUBBLESORT consiste en recorrer la lista `A` de izquierda a derecha, dejando ordenado en cada paso al par de elementos `A[i]` y `A[i+1]`. Esta pasada por toda la lista debe repetirse `len(A)` veces. Implementar el algoritmo en Python. ¿Por qué funciona?

(b) Mostrar que BUBBLESORT tiene complejidad $O(\texttt{len(A)}^2)$.

Ejercicio 8. Para este ejercicio, adaptar los programas de búsqueda vistos en clase.

(a) Escribir un programa que, dados una lista de enteros `A` y un entero `x`, retorne todos los elementos de `A` menores o iguales que `x`. Estimar su complejidad algorítmica (en peor caso).

(b) Igual al punto anterior, pero sabiendo que la lista está ordenada.

Ejercicio 9. Escribir un algoritmo que resuelva cada uno de los siguientes problemas con el orden de complejidad algorítmica indicado. Mostrar que el algoritmo hallado tiene el orden indicado.

(a) Calcular la media de una lista `A` de enteros. $O(\texttt{len(A)})$

(b) Calcular la mediana de una lista `A` de enteros. $O(\texttt{len(A)}^2)$

(c) Determinar, dado un `n` natural, si `n` es (o no) primo. $O(\sqrt{\texttt{n}})$

(d) Dada una lista `A` de 1s y 0s representando un número en base binaria, obtener el número correspondiente en base decimal. $O(\texttt{len(A)})$

(e) Dada una lista de enteros ordenada de menor a mayor, devolver `True` si todos sus elementos son positivos, o `False` si hay algún elemento negativo. $O(1)$

(f) Dados un entero `x` y una lista con `n` listas de enteros, cada una ordenada de menor a mayor y con a lo sumo `m` elementos, calcular la cantidad de listas que contienen al entero `x`. $O(\texttt{n} \cdot \log \texttt{m})$