El problema de ordenar

Introducción

El problema de ordenar es uno de los más importantes en ciencias de la computación. Por un lado, hay muchísimas situaciones algorítmicas en las que primero tenemos ordenar. Por otro lado, los algoritmos para ordenar guardan muchas de las ideas y heurísticas que necesitamos en varios otros algorimos. Finalmente, la misma idea de ordenar responde muchas preguntas de aplicación directamente.

Ordenar objetos

De manera intuitiva, el problema algorítmico que queremos resolver es el siguiente:

Problema. Ordenar.

Entrada. Nos dan una colección de objetos. Nos dan una manera de compararlos dos a dos para decir cuál es mayor o igual que el otro.

Salida. Una lista ordenada de esos mismos objetos, es decir, poner a estos objetos de la forma $a_1\leq a_2\leq a_3\leq \ldots\leq a_n$, en donde los a_i's son los mismos objetos con los que comenzamos.

Algunas sutilezas

Tenemos que tener varias cosas en cuenta cuando queremos resolver el problema de ordenar.

- ¿Queremos que los objetos queden ordenados de manera creciente o decreciente?
- ¿Queremos ordenar únicamente la magintud que nos está permitiendo comparar, o todos los objetos?

Por ejemplo, dada la información

```
Luis - 200
Paco - 4000
Laura - 4500
Josefina - 750
```

podríamos únicamente querer clasificar los montos, donde por respuesta esperamos 4500, 4000, 750, 200; o bien, podríamos querer clasificar los objetos *completos* por monto:

```
Laura - 4500
Paco - 4000
Josefina - 750
Luis - 200
```

- ¿Qué hacemos si hay objetos en donde el criterio a comparar es igual? Una opción es que no nos importe en qué orden darlos. Otra opción es que querramos ordenar secundariamente por otro criterio.
- ¿Cómo se va a ordenar en los casos en los que el orden que queremos no es obvio? Por ejemplo, con las palabras no es totalmente obvio si queremos ordenarlas alfabéticamente, por longitud, por frecuencia de uso, etc.

Una de las formas en las que podemos ponernos de acuerdo en varias de las sutilezas anteriores es mediante la designación de un *orden parcial* en el universo de objetos que nos interesan. Como recordatorio, un *orden parcial* es una relación $\leq$ en un conjunto X para la cual se cumplen las siguientes propiedades:

- **Reflexividad:** $a\leq a$ para todo $a\in X$.
- **Antisimetría:** Si $a\leq b$ y $b\leq a$, entonces $a=b$.
- **Transitividad:** Si $a\leq b$ y $b\leq c$, entonces $a\leq c$.

De esta manera, podemos escribir el problema de ordenar objetos más formalmente como sigue:

Problema. Ordenar.

Entrada. Una colección $a_1, a_2,\ldots,a_n$ de elementos en un conjunto X y un orden parcial $\leq$ en X.

Salida. Elementos $b_1,b_2,\ldots,b_n$ que como colección sean la misma que $a_1,a_2,\ldots,a_n$, pero que cumplan que

$$b_1\leq b_2\leq b_3\leq \ldots \leq b_n.$$

Propiedades deseables de algoritmos de ordenación

Hay muchas formas de ordenar objetos. Cualquiera de estas formas debe de ser correcta. Pero fuera de eso, es difícil decir cuál es la mejor forma de ordenar de manera absoluta. Es por ello que hay varias propiedades que pueden tener los algoritmos de ordenación que los hacen suceptibles a ser mejores en ciertos contextos que en otros. No necesariamente todos los algoritmos de ordenación cumplen todas las siguientes propiedades y de hecho en ocasiones habrá que elegir una sobre otra.

- **Estabilidad.** La forma de romper empates cuando en el orden parcial se tiene que son iguales es mediante el orden original en el que nos dieron los objetos.
- **No destructivos.** Además de dar la lista ordenada, también preservan en memoria la lista original.
- **Usen poco espacio computacional.** De preferencia, cuando sea posible, nos gustaría que los algoritmos que usemos usen la mínima cantidad posible de espacio extra, además del espacio $O(n)$ que ya estamos usando para almacenar la lista de objetos en general.
- **Usen poco tiempo computacional.** Nos gustaría que tomaran la menor cantidad de tiempo posible. Más adelante veremos que lo mejor que podemos esperar en general es tiempo $\Theta(n\log n)$.
- **Sean sensibles al universo que tenemos.** Si tenemos instancias muy particulares, o un universo de objetos limitado, o un orden parcial X muy específico, es posible que sea posible ordenar con tiempo más pequeño que $O(n\log n)$. En ese caso es mejor utilizar algoritmos *ad hoc*.
- **Se comporten bien con colecciones casi ordenadas.** Si como entrada recibimos objetos que ya están «prácticamente ordenados», entonces el algoritmo debería «tomar menos tiempo».

Tarea moral

Los siguientes problemas te ayudarán a practicar lo visto en esta entrada. Para resolverlos, necesitarás usar herramientas matemáticas, computacionales o ambas.

1. Ordena manualmente la siguiente lista de números:
   $$13,12,41,23,7,11,9,28,13,2,9,2,11,21$$
   ¿Qué pasos hiciste? ¿Como cuánto te tardaste? ¿Qué suposiciones hiciste acerca del orden? ¿Cómo se lo explicarías a alguien más?

2. Imagina que tienes 5 tarjetas boca abajo sobre una mesa, puestas de izquierda a derecha. Cada tarjeta tiene un número que no ves. Tienes que pagar un peso cada vez que veas una tarjeta y un peso cada vez que intercambies dos de ellas. Estos son los únicos pasos permitidos. ¿Cuál es la mínima cantidad de pesos que tienes que pagar para que con estas operaciones puedas garantizar llegar al acomodo en el que las tarjetas están ordenadas de izquerda a derecha?
3. Explica por qué la «no destructibilidad» no es tan compatible con el «uso de poco espacio». ¿Hay otras propiedades deseables de algoritmos de ordenación que crees que se opongan entre sí?

4. Para darte una idea más amplia de las propiedades buscadas en algoritmos de ordenación y de cuáles y cuántos existen, revisa la siguiente entrada en Wikipedia: https://en.wikipedia.org/wiki/Sorting_algorithm
5. Para divertirte viendo cómo ciertos bailarines hacen coreografías con algoritmos de ordenación, revisa el siguiente canal de YouTube: https://www.youtube.com/user/AlgoRythmics/videos

Anterior
Ejemplos de análisis de eficiencia

Siguiente
Ordenar cuadráticamente

Por Leonardo Ignacio Martínez Sandoval