Introducción a las estructuras de datos

Estructuras de datos

Introducción

- Uno de los aspectos fundamentales en el desarrollo de software es el acceso y manipulación de datos.
- La forma en la que estructuramos los datos tiene un enorme impacto en la facilidad de uso de los mismos.
- Adaptar las estructuras que usamos para manejar los datos al problema que se está tratando de resolver nos permite resolver los problemas de forma mucho más eficiente.
- Llamamos estructura de datos a una forma de organizar, almacenar y gestionar un conjunto de datos.
- Cada estructura de datos está optimizada para un conjunto de operaciones concreto.

Estructuras estáticas y dinámicas

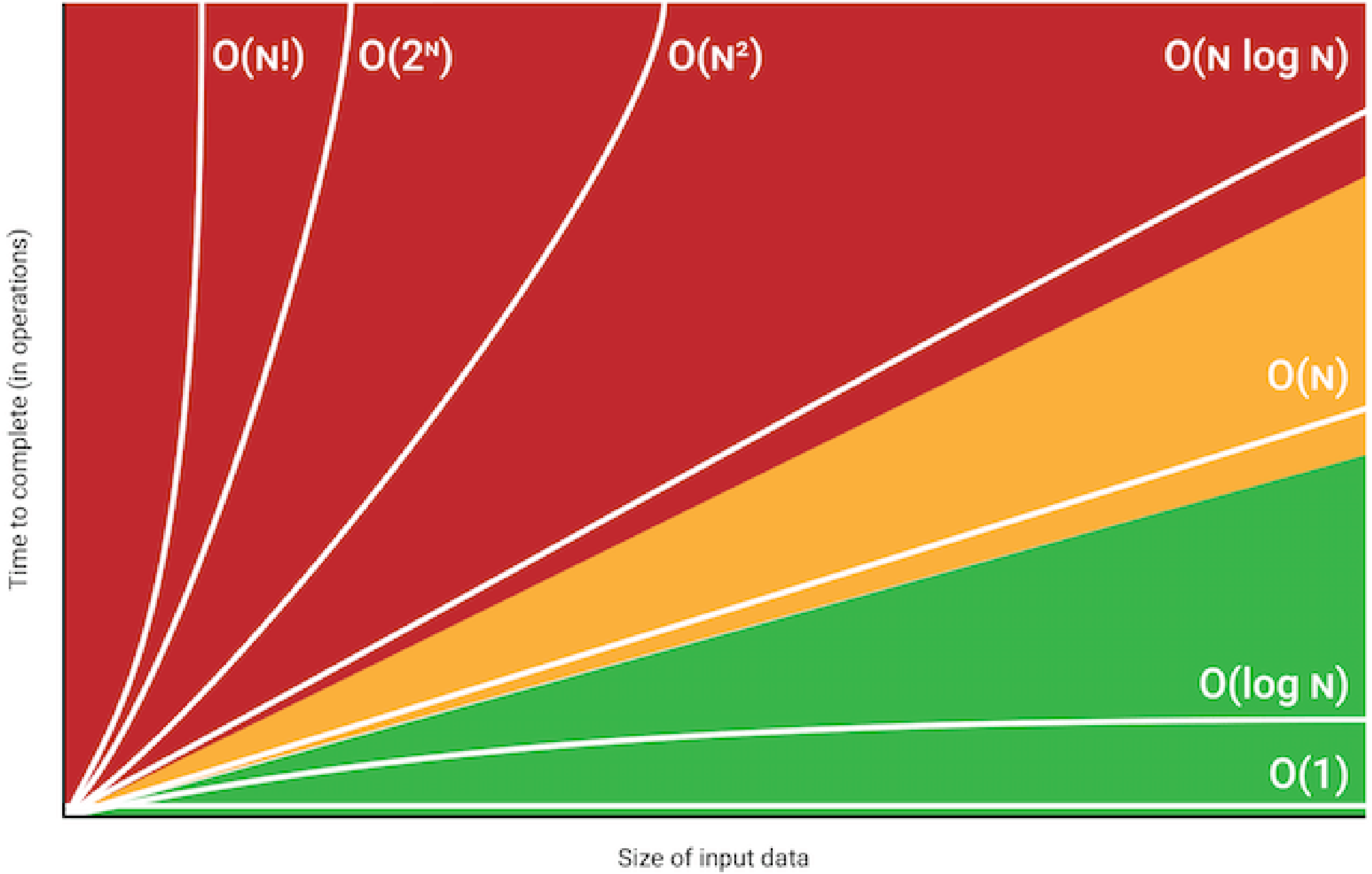
- La primera gran clasificación de las estructuras de datos hace referencia a la capacidad de las mismas a cambiar de tamaño dinámicamente.
- Las estructuras **estáticas** son aquellas que definen su tamaño en el momento de su creación y **no tienen la capacidad de aumentar ni disminuir su tamaño**.
- El ejemplo más común de estructuras estáticas es el **array**.
- Las estructuras **dinámicas** con aquellas con la **capacidad de aumentar o disminuir su tamaño** en función de las necesidades del programa que las está usando.
- Existe una gran variedad de estructuras dinámicas: **listas enlazadas, árboles binarios**...

Eficiencia de las estructuras de datos

- Cada estructura de datos ha sido diseñada para optimizar la ejecución de ciertas operaciones.
- Por ejemplo, el array es muy rápido a la hora de leer cualquiera de sus posiciones. Sin embargo, no resulta tan rápido insertar un dato, ya que implica crear un nuevo array, copiar los datos antiguos y añadir el dato nuevo al final.
- Durante esta UT vamos a estudiar las principales estructuras de datos y vamos a ver para qué operaciones están optimizadas.
- Para clasificar la eficiencia de las operaciones de las estructuras de datos utilizaremos la notación Big-O

Big-O

- La notación Big-O es una manera de medir la eficiencia de las operaciones de las estructuras de datos.
- Lo que se busca es describir con una fórmula matemática cuál es la relación entre la cantidad de datos (n) que se manejan y el coste computacional de una operación dada.
- Se utiliza para medir dos cosas: el espacio en memoria necesario para llevar a cabo una operación (complejidad espacial) y la cantidad de operaciones necesarias para llevar a cabo una operación (complejidad temporal)
- Vamos a centrarnos en la complejidad temporal.


O(N!)
O(2^N)
O(N²)
O(N log N)
O(N)
O(log N)
O(1)
Time to complete (in operations)
Size of input data

Big-O

- **O(1)**: también conocida como **constante**, significa que la operación se ejecuta en un tiempo fijo, independiente de la cantidad de datos que almacene la estructura de datos. Por ejemplo, el acceso a una posición de un array.
- **O(log n)**: también conocida como **logarítmica**, el tiempo de ejecución crece conforme aumentan los datos en la estructura, pero la curva se va aplanando conforme aumenta la cantidad de datos.
- **O(n)**: también conocida como **lineal**, el tiempo de ejecución crece de manera proporcional al tamaño de la estructura de datos.
- **O(n log n)**: el tiempo de ejecución crece algo más a lo proporcional en este caso.

Big-O

- **$O(n^2)$**: también conocida como complejidad **cuadrática**. El tiempo de ejecución aumenta mucho más rápido que el tamaño de la estructura y esto hace que no sea factible realizar la operación cuando la estructura tiene muchos datos.
- **$O(2^n)$**: también conocida como **exponencial**. El tiempo de ejecución aumenta a una velocidad aún mayor que la cuadrática, haciendo inviable la operación con tamaños incluso menores.
- **O(n!)**: también conocida como **factorial**. El tiempo de ejecución aumenta a una velocidad aún mayor que la exponencial, haciendo inviable la operación con tamaños incluso menores.