Facultad de ingeniería, UNAM

División de Ingeniería Eléctrica

Asignatura: Estructura de Datos Y Algoritmos I

Profesor: Castillo Hernández Gabriel

Grupo: 11

Integrantes:

Calzada Martínez Jonathan Omar

**ALGORITMOS**

Un algoritmo es un conjunto de pasos finitos para realizar una tarea específica.

En Ciencias de La computación un algoritmo es un conjunto de pasos para que un programa realice una tarea específica. Los algoritmos son la ciencia en la Ciencias de la computación.

Algunos ejemplos de algoritmos famosos:

* Hangouts de Google: Utiliza algoritmos de comprensión para poder transmitir videos en vivo de manera rápida.
* Google maps.

Utiliza algoritmos de búsqueda para poder encontrar la ruta más rápida y óptima para un destino.

* Pixar colorea un modelo 3D basado en la iluminación de un curato virtual. Utiliza un algoritmo de renderizado.

**Juego de adivinar**

Se expone un juego en donde la computadora tiene un programa en donde se elige un numero del al 16 de manera aleatoria.

Se pone el listado del 1 al 16 y tenemos que encontrar la manera de saber el numero que la maquina eligió, esto con el menor número de intentos posibles.

Algo fácil es ir intentando uno por uno hasta dar con el resultado, si el numero elegido por la maquina fuera el 1 entonces podríamos decir que nos fue muy bien, pero que pasaría si el numero es el ultimo y la lista sería del 1 al 300. Esta técnica para encontrar el numero seria muy tardada, esta es una búsqueda lineal.

Pero esto es más fácil cuando lo dividimos por bloques de esta manera si el número es el que esta en medio ya quedo resulto el problema y si el número está en la primera parte se descarta la otra mitad de el listado.

**Encontrar un camino**

En ocasiones nos encontramos con problemas muy similares sin darnos cuenta. En muchas ocasiones necesitamos encontrar un camino para realizar una tarea o llegar a un lugar y necesitamos saber cual es el más corto o el mejor camino dependiendo de nuestras necesidades.

Por ejemplo:

Para Pac-Man, necesitamos un mapa del laberinto. Este mapa muestra las conexiones entre casillas abiertas adyacentes en el laberinto (o la falta de conexiones si es que hay una pared en medio) y el problema es encontrar un camino que lleve al fantasma a Pac-Man.

Google mapas o waze

En donde podemos solicitar la ruta para llegar a un lugar por ejemplo: Para Pac-Man, necesitamos un mapa del laberinto. Este mapa muestra las conexiones entre casillas abiertas adyacentes en el laberinto (o la falta de conexiones si es que hay una pared en medio) y el problema es encontrar un camino que lleve al fantasma a Pac-Man.

**Búsqueda Binaria**

La búsqueda binaria es un algoritmo eficiente para encontrar un elemento en una lista ordenada de elementos. Funciona al dividir repetidamente a la mitad la porción de la lista que podría contener al elemento, hasta reducir las ubicaciones posibles a solo una. Usamos la búsqueda binaria en el [juego de adivinar](https://es.khanacademy.org/computing/computer-science/algorithms/intro-to-algorithms/a/a-guessing-game) en la lección introductoria.

Pseudocódigo para la búsqueda binaria

Al describir un algoritmo para un ser humano, a menudo es suficiente una descripción incompleta. Algunos detalles pueden quedar fuera de una receta de un pastel; la receta supone que sabes cómo abrir el refrigerador para sacar los huevos y que sabes cómo romper los huevos. La gente puede saber intuitivamente cómo completar los detalles faltantes, pero los programas de computadora no. Es por eso que necesitamos describir completamente los algoritmos computacionales.

**Notación Asintótica**

El tiempo de ejecución de un algoritmo depende de cuánto tiempo le tome a una computadora ejecutar las líneas de código del algoritmo, y eso depende de la velocidad de la computadora, el lenguaje de programación y el compilador que traduce el programa del lenguaje de programación al código que se ejecuta directamente en la computadora, entre otros factores.

La segunda idea es que debemos enfocarnos en qué tan rápido crece una función con el tamaño de la entrada. A esto lo llamamos la tasa de crecimiento del tiempo de ejecución.

Podemos calcular el tiempo de ejecución de un algoritmo a un sin contar una cantidad constante de tiempo esto se puede calcular gracias a los logaritmos.

Ordenamiento

Ordenar una lista de elementos en orden ascendente o descendiente puede ayudar tanto a un ser humano como a una computadora a encontrar algún elemento de una lista por ejemplo en la búsqueda binaria.

### Función swap

Un paso clave en muchos algoritmos de ordenamiento (incluyendo el ordenamiento por selección) es intercambiar la ubicación de dos elementos en un arreglo. Aquí hay una función swap que parece podría funcionar, pero no:  
-el código escribe [9, 9, 4] cuando debería escribir [9, 7, 4].

# Pseudocódigo del ordenamiento por selección

1. Encuentra la carta más baja. Intercámbiala con la primera carta.
2. Encuentra la segunda carta más baja. Intercámbiala con la segunda carta.
3. Encuentra la tercera carta más baja. Intercámbiala con la tercera carta.
4. Repite encontrar la siguiente carta más baja e intercambiarla en la posición correcta hasta que el arreglo esté ordenado.

Este algoritmo se llama ordenamiento por selección porque seleccionar repetidamente el siguiente elemento más bajo y lo intercambia a su lugar.

El ordenamiento por selección itera sobre los índices en el arreglo; para cada índice, el ordenamiento por selección llama indexOfMinimum y swap. Si la longitud del arreglo es n*n*n, hay n*n*n índices en el arreglo.

Como cada ejecución del cuerpo del bucle ejecuta dos líneas de código, podrías pensar que el ordenamiento por selección ejecuta 2 n2*n*2, n líneas de código. ¡Pero no es cierto! Recuerda que indexOfMinimum y swap son funciones: cuando cualquiera de las dos se llama, se ejecutan algunas líneas de código.

**Ordenamiento por Inserción**

El ordenamiento por selección busca en el sub-arreglo no ordenado el siguiente elemento que será incluido en el sub-arreglo ordenado.

El ordenamiento por inserción inserta repetidamente un elemento en el sub-arreglo ordenado que está a su izquierda. Al inicio, podemos decir que el sub-arreglo que solo contiene al índice 0 está ordenado, ya que contiene un solo elemento, y ¿cómo un solo elemento podría no estar ordenado con respecto a sí msimo? Debe estar ordenado. Trabajemos un ejemplo. Aquí está nuestro arreglo inicial:

**Recursión**

En [ciencias de computación](https://es.wikipedia.org/wiki/Ciencias_de_la_computaci%C3%B3n), una forma de atajar y solventar problemas. De hecho, recursión es una de las ideas centrales de ciencia de computación.[1](https://es.wikipedia.org/wiki/Recursi%C3%B3n_(ciencias_de_computaci%C3%B3n)#cite_note-1)​ Resolver un problema mediante recursión significa que la solución depende de las soluciones de pequeñas instancias del mismo problema.[2](https://es.wikipedia.org/wiki/Recursi%C3%B3n_(ciencias_de_computaci%C3%B3n)#cite_note-2)​

# Algoritmos de divide y vencerás

El ordenamiento por mezcla se ejecuta en un tiempo \Theta(n \lg n)Θ(*n*lg*n*) en todos los casos y el ordenamiento rápido se ejecuta en un tiempo \Theta(n \lg n)Θ(*n*lg*n*) en el mejor caso y en el caso promedio, aunque el tiempo de ejecución de su peor caso es \Theta(n^2)Θ(*n*2). Aquí hay una tabla de estos cuatro algoritmos de ordenamiento y sus tiempos de ejecución:

**Divide y vencerás**

El ordenamiento por mezcla como el ordenamiento rápido usan un paradigma algorítmico común que se basa en la recursividad. Este paradigma, **divide y vencerás**, separa un problema en subproblemas que se parecen al problema original, de manera recursiva resuelve los subproblemas y, por último, combina las soluciones de los subproblemas para resolver el problema original.

El ordenamiento rápido tiene un par de otras diferencias con el ordenamiento por mezcla. El ordenamiento rápido trabaja in situ. Y el tiempo de ejecución de su peor caso es tan malo como el del ordenamiento por selección y el ordenamiento por inserción: \Theta(n^2)Θ(*n*2). Pero el tiempo de ejecución de su caso promedio es tan bueno como el del ordenamiento por mezcla: \Theta(n \lg n)Θ(*n*lg*n*). ¿Entonces por qué pensar acerca del ordenamiento rápido cuando el ordenamiento por mezcla es por lo menos igual de bueno? Eso es porque el factor constante escondido en la notación Θ grande para el ordenamiento rápido es bastante bueno. En la práctica, el ordenamiento rápido tiene un mejor desempeño que el ordenamiento por selección y tiene un desempeño significativamente mejor que el ordenamiento por inserción.

Tom Cormen y Devin Balkcom

**https://es.khanacademy.org/computing/computer-science/algorithms**