COMPUTACIÓN

ADOLFO SANZ DE DIEGO

MÁSTER UAH

2 ACERCA DE

2.1 AUTOR

- **Adolfo Sanz De Diego**
  - Blog: asanzdiego.blogspot.com.es
  - Correo: asanzdiego@gmail.com
  - GitHub: github.com/asanzdiego
  - Twitter: twitter.com/asanzdiego
  - Linkedin: in/asanzdiego
  - SlideShare: slideshare.net/asanzdiego

3 COMPLEJIDAD COMPUTACIONAL

3.1 INTRODUCCIÓN

- La teoría de la complejidad computacional trata de **clasificar los problemas que pueden, o no pueden ser resueltos** con una cantidad determinada de recursos (tiempo y memoria).
- A grandes rasgos, teoría de la complejidad computacional trata de clasificar los problemas que pueden, o no pueden ser resueltos por una computadora.

3.2 MÁQUINA DE TURING

- Una máquina de Turing es un dispositivo teórico que manipula símbolos sobre una tira de cinta de acuerdo a una tabla de reglas.
- Estudiando sus propiedades abstractas, ha servido de **base para mucho desarrollo teórico** en las ciencias de la computación y en la teoría de la complejidad.

Vídeo "¿Qué es una máquina de Turing?"

3.3 ALGORITMO

- Un algoritmo es un **conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas** que permiten llevar a cabo una actividad mediante pasos sucesivos que no generen dudas a quien deba hacer dicha actividad.
- Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final.

Vídeo "¿Qué es un algoritmo?"

3.4 TIEMPOS POLINÓMICOS

- Cuando el tiempo de ejecución de un algoritmo se puede expresar usando una fórmula polinómica, se dice que dicho problema se puede resolver en un tiempo polinómico.
- Estos son **"buenos" algoritmos**.

3.5 TIEMPOS EXPONENCIALES

- Cuando el tiempo de ejecución de un algoritmo no se puede expresar usando una fórmula polinómica, se dice que dicho problema es de tiempo exponencial.
- Cuando un problema solo se puede resolver mediante algoritmos exponenciales, **se dice que es intratable**.

3.6 PROBLEMAS DE CLASE NP

- La clase de complejidad NP consta de los problemas "verificables" en tiempo polinómico, es decir, dada una posible solución, esta se puede verificar en un tiempo polinómico por una máquina de Turing.
- A grandes rasgos, NP corresponde a la clase de problemas que, de manera realista, **se pueden verificar con una computadora**.

3.7 PROBLEMAS DE CLASE P

- La clase de complejidad P contiene a aquellos problemas que se pueden resolver en tiempo polinómico por una máquina de Turing.
- A grandes rasgos, P corresponde a la clase de problemas que, de manera realista, **se pueden resolver con una computadora**.
- Los problemas de clase P siempre son de clase NP (si lo podemos resolver en tiempo polinómico, también podemos verificarlo en tiempo polinómico).

3.8 PROBLEMAS NP-COMPLETO

- La clase de complejidad NP-Completo consta de los problemas "verificables" en tiempo polinómico, pero que no se ha encontrado un algoritmo que pueda resolver dicho problema en tiempo polinómico por una máquina de Turing.
- A grandes rasgos, NP-Completo corresponde a la clase de problemas que **se pueden verificar de forma sencilla, pero solo se pueden resolver por fuerza bruta**.
- Los problemas de clase NP-Completo siempre son de clase NP.

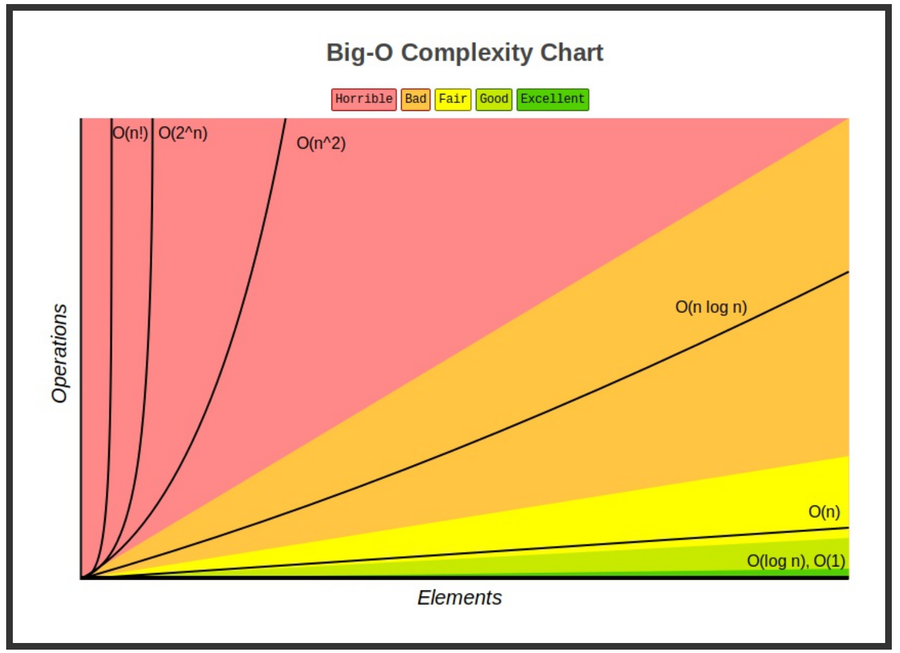
3.9 EL PROBLEMA ¿P=NP?

- La pregunta ¿P=NP? es uno de los 7 problemas del milenio.
- Si P=NP, cualquier problema polinómicamente verificable (NP) se podría resolver polinómicamente (P).
- Esto quiere decir que podríamos pues resolver los problemas de clase NP-Completo de forma polinómica.

Vídeo "¿Qué es eso del problema P versus NP?"

3.10 LA NOTACIÓN O GRANDE

- Es una notación matemática que nos **ayuda a describir el comportamiento de un algoritmo "al límite"** en función de sus elementos de entrada N.



Vídeo "Big O Notation"

3.11 TIPOS DE COMPLEJIDAD

notación	nombre
O(1)	constante
O(log n)	logarítmica
O(n)	lineal
O(n log n)	lineal logarítmica
O(n^2)	cuadrática
O(2^n)	exponencial
O(n!)	factorial

3.12 EJEMPLOS DE ALGORITMOS

Sorting Algorithms Animations

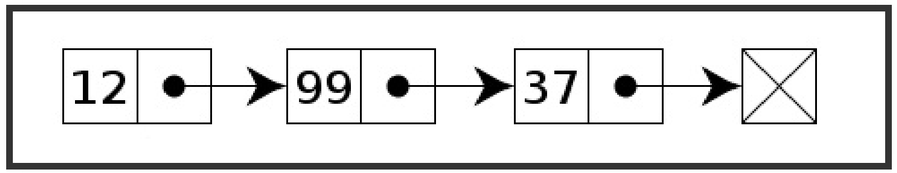
Binary search algorithm implementation in javascript

Vídeo "Algoritmo de Dijkstra"

4 ESTRUCTURAS DE DATOS

4.1 LISTAS ENLAZADAS

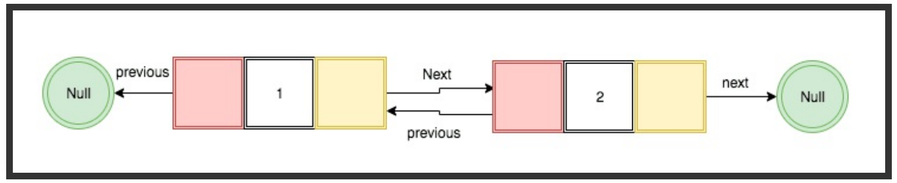
- Una lista enlazada es una colección de nodos donde **cada nodo tiene una conexión con el siguiente** nodo.



DataStructures: Implementation of Linked List in JavaScript

4.2 LISTAS DOBLEMENTE ENLAZADAS

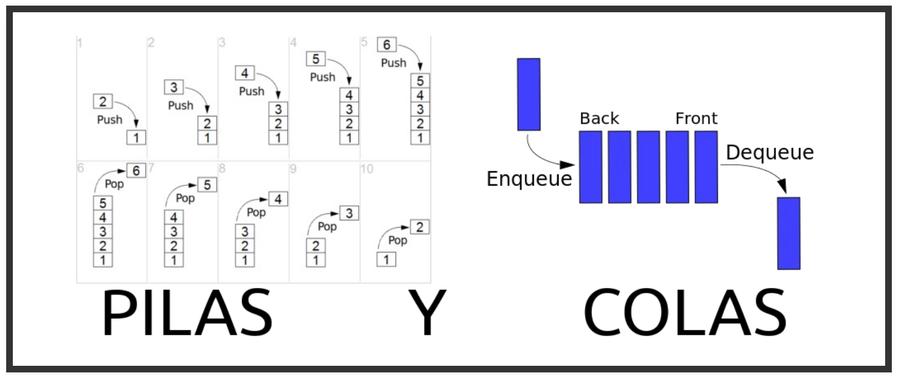
- En una lista doblemente enlazada, **cada nodo tiene una referencia al nodo anterior y al siguiente**.



Doubly Linked List Implementation in JavaScript

4.3 PILAS Y COLAS

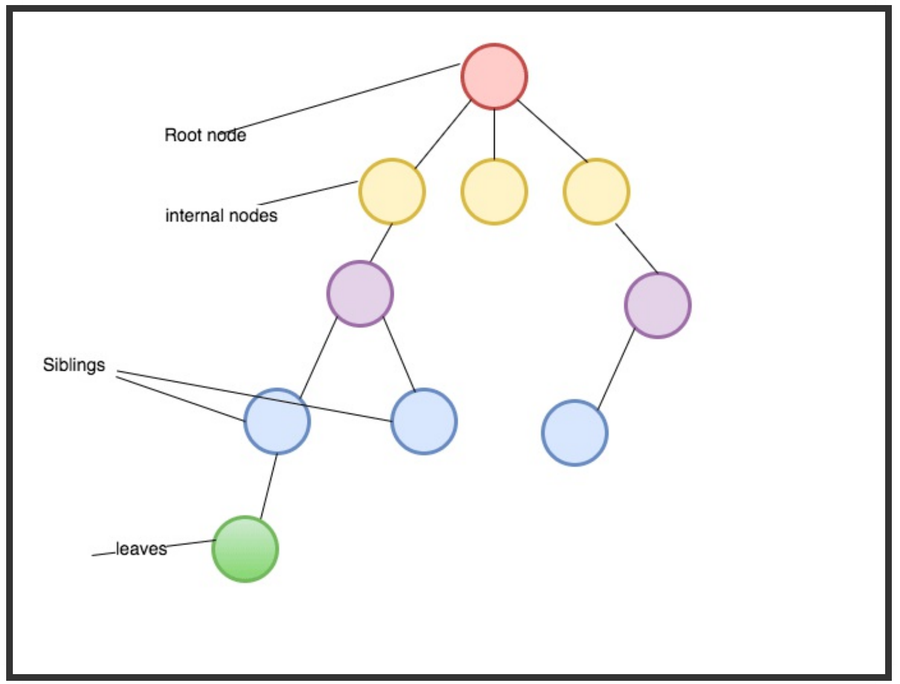
- Una **pila es un conjunto de elementos que utilizan el principio LIFO** (Last In First Out), es decir, el último elemento que introdujimos será el primero en ser retirado.
- Una **cola es un conjunto de elementos que utilizan el principio FIFO** (First In First Out), es decir, el primer elemento que introdujimos será el primero en ser retirado.



Stacks and Queues in JavaScript

4.4 ÁRBOLES

- Un árbol es una **estructura de datos no lineal** comparada con las pilas y colas, las listas enlazadas y las matrices que son estructuras de datos lineales.



Tree Data structure in Javascript

5 TEORÍA DE GRAFOS

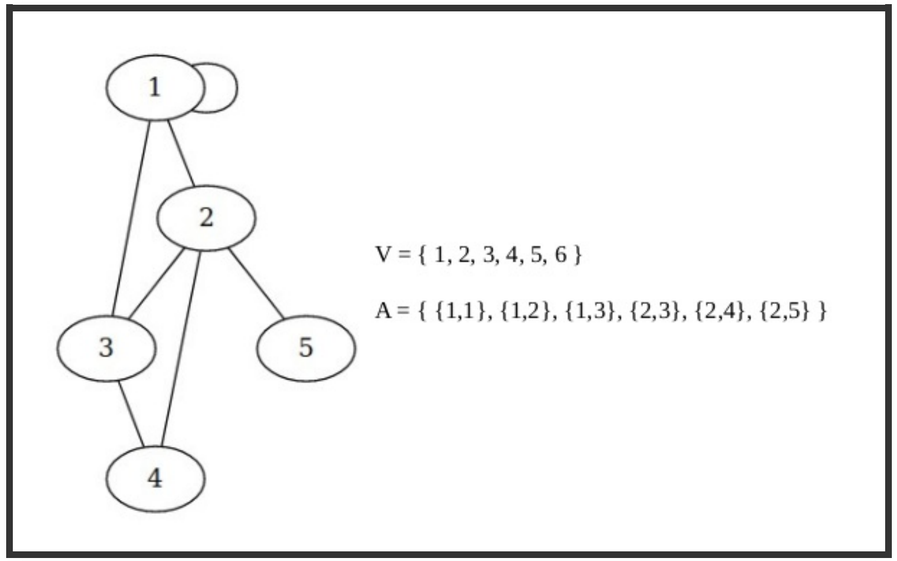
5.1 GRAFOS

- Un grafo G=(V,A) es una pareja ordenada en la que **V es un conjunto no vacío de vértices y A es un conjunto de aristas**.

Data structures: Introduction to graphs

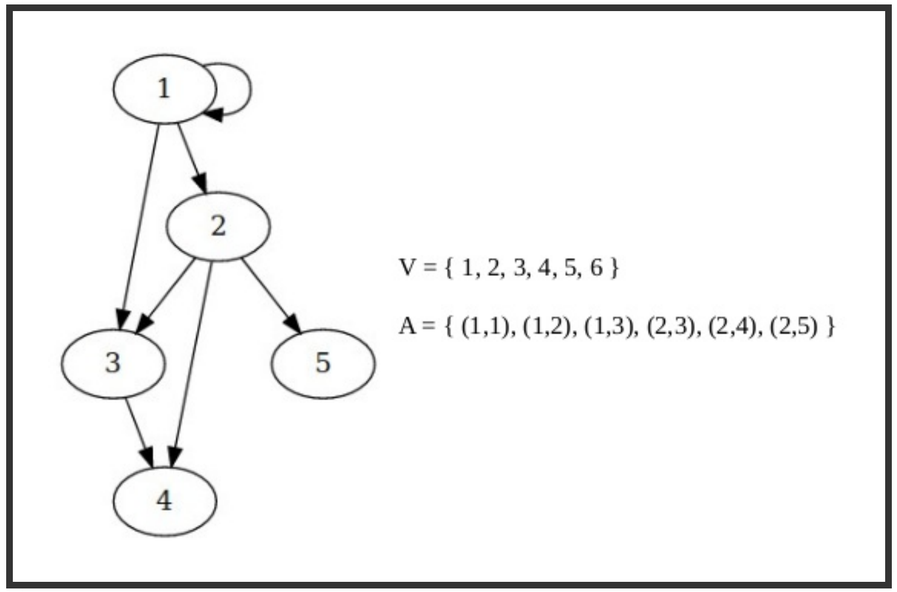
5.2 GRAFOS NO DIRIGIDO

- Si A consta de **pares no ordenados de vértices**, tales como {x,y} entonces se dice que x e y son adyacentes, y en el grafo se representa mediante una línea no orientada que una dichos vértices.



5.3 GRAFOS DIRIGIDO

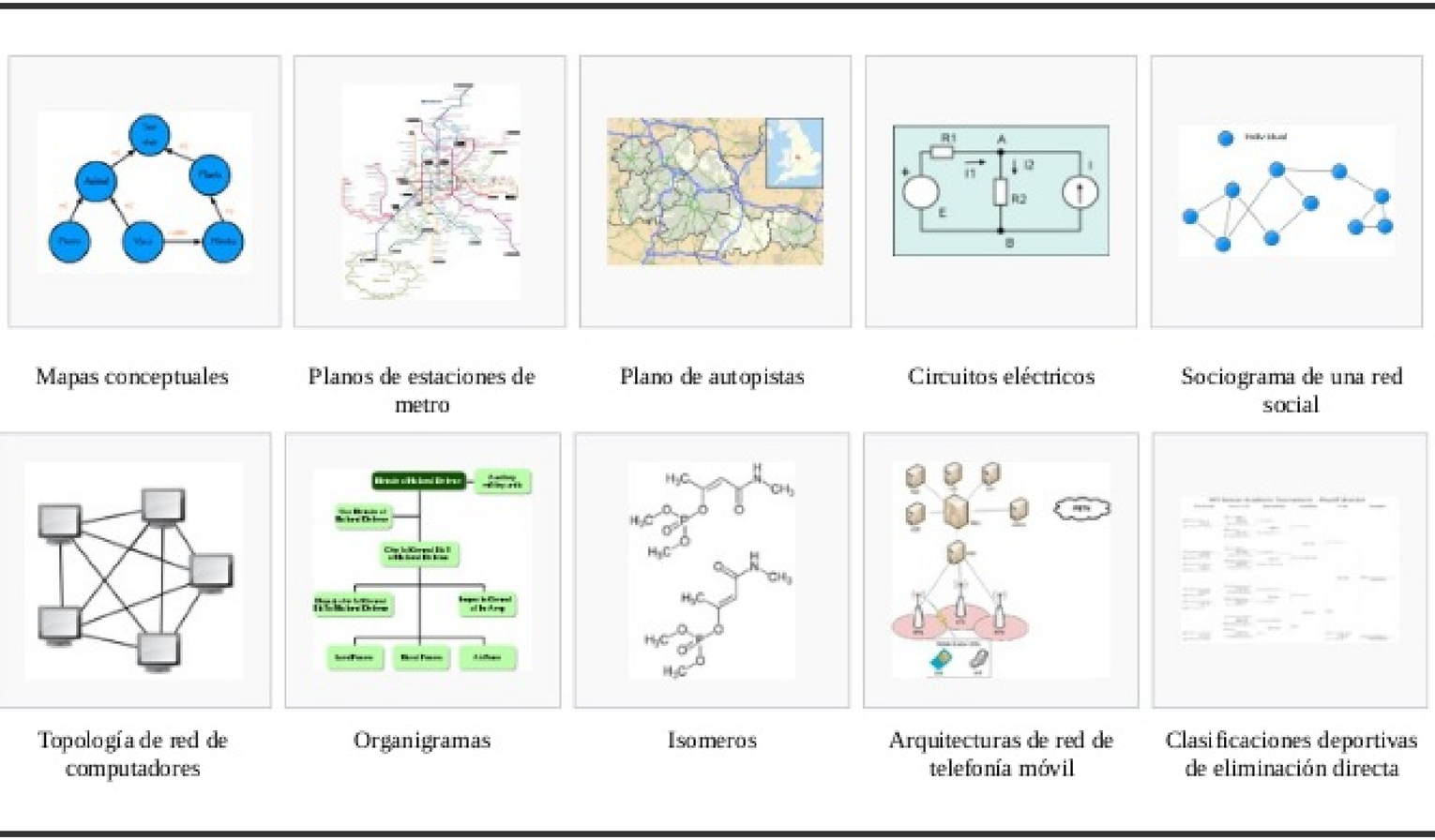
- Si A consta de **pares ordenados de vértices**, tales como (x,y) entonces se dice que x e y es un par ordenado, y en el grafo se representa con una flecha que una dichos vértices. A este tipo de grafos se le llama dígrafo y se denota D.



5.4 APLICACIONES

- Debido a la gran **cantidad de aplicaciones en la optimización de recorridos, procesos, flujos, algoritmos de búsquedas**, la teoría de grafos es actualmente muy usada en el campo de la informática, las ciencias de la computación y telecomunicaciones.

5.5 EJEMPLOS



Mapas conceptuales

Planos de estaciones de metro

Plano de autopistas

Circuitos eléctricos

Sociograma de una red social

Topología de red de computadores

Organigramas

Isomeros

Arquitecturas de red de telefonía móvil

Clasificaciones deportivas de eliminación directa

6 BIBLIOGRAFÍA

6.1 GENERALES

Big-O Algorithm Complexity Cheat Sheet

Curso de Khan Academy > Ciencias de la computación > Algoritmos

Algorithms and data structures implemented in JavaScript with explanations and links to further readings

6.2 TURING Y P=NP

Vídeo "¿Qué es un algoritmo?"

Vídeo "¿Qué es una máquina de Turing?"

Vídeo "¿Qué es eso del problema P versus NP?"

6.3 BIG O

Vídeo "Big O Notation"

Intro to Big O Notation

Intro to Logarithms Big O

6.4 VISUALIZADORES

Sorting Algorithms Animations

Vídeo "Visualization and Comparison of Sorting Algorithms"

Algorithm Visualizer

Graph Algorithm Playground

6.5 ALGORITMOS DE ORDENACIÓN

Sorting algorithms beginners guide

Bubble sort algorithm in JavaScript

Selection sort algorithm in javascript

How does merge sort algorithm works?

Quicksort algorithm implementation

What is radix sort?

Lista de reproducción "Algoritmos de ordenación"

6.6 OTROS ALGORITMOS

Binary search algorithm implementation in javascript

How to implement Dijkstra’s Algorithm in JavaScript

Vídeo "Algoritmo de Dijkstra"

6.7 ESTRUCTURAS DE DATOS

DataStructures: Implementation of Linked List in JavaScript

Doubly Linked List Implementation in JavaScript

Stacks and Queues in JavaScript

Tree Data structure in Javascript

Lista de reproducción "Estructuras de datos"

Data structures: Introduction to graphs