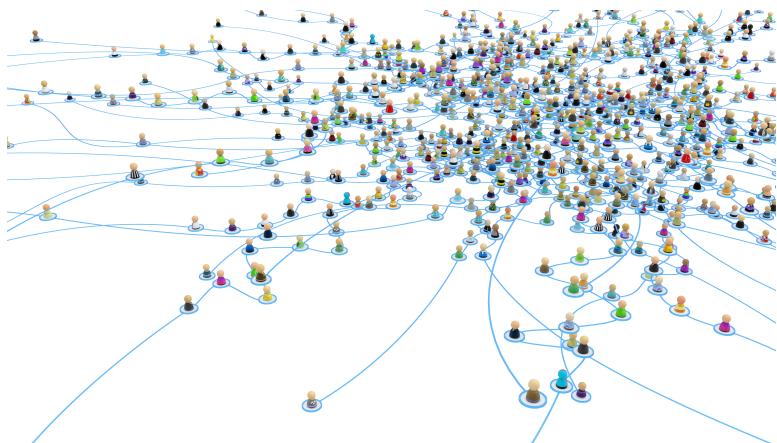




Instituto Politécnico Nacional Escuela Superior de Cómputo

Análisis y Diseño de Algoritmos
Profesor: Cristhian Avila Sanchez
Grupo: 3CV3
Adrian González Pardo
Semestre: 20/01



Ultima fecha modificada: 3 de febrero de 2020

Índice

1. Incio	3
1.1. Presentación	3
1.1.1. Prerequisitos de la materia (No estrictos)	3
1.1.2. Topicos de la materia	3
1.1.3. Evaluación de la materia	3
1.1.4. Libros que pueden ser de ayuda	3
2. Primer parcial	4
2.1. ¿Qué es un Algoritmo?	4
2.1.1. Computación	4
2.2. Complejidad Computacional	4
2.2.1. Enfoque temporal	5
2.2.2. Ordenamiento [Introducción a clase Polinomial (P)]	6
2.2.3. Algunos otros algoritmos e ideas de problemas de la Clase NP	7
2.2.4. Complejidades en NP vs P	7
2.2.5. Maquina Z	8
2.2.6. Número complejo	8
2.2.7. NP (Subclase NP-Completo)	8

1. Incio

1.1. Presentación

El fin de comprender el Análisis y Diseño de Algoritmos en la ingeniería, es con la tarea de poder resolver problemas en los que se pueda automatizar procesos con el uso del poder computable aun cuando estos pueden tener una mayor complejidad en la Automatización de dicho, por lo cual se abordaran problemas acorde a su tipo de clase de complejidad y se llevara un análisis en el que se planea llegar a una aproximación de solución o a su solución total.

1.1.1. Prerequisitos de la materia (No estrictos)

Lenguaje C (o de la familia de lenguajes C)	Teoría de la Computación
Probabilidad	Variable Compleja
Matemáticas Discretas	Análisis Vectorial
Algebra Lineal	Calculo I,II
Ecuaciones Diferenciales	Física

Nota: es necesario tener noción de estos topicos debido a ciertos temás en los que se puede abordar acorde a temas anteriormente vistos y por otro lado es necesario saber como trabaja a bajo nivel todo el proceso de calculo de complejidad temporal y espacial con uso de bajo recurso.

1.1.2. Topicos de la materia

De acuerdo al plan estudios propuesto en la página de la ESCOM se abordaran los siguientes temas tomando de guía el temario y los temas propuestos por el profesor.

Problemas P	Heuristicas
Complejidad Computacional	Programación Dínámica
Algoritmos Aleatorios	NP-Completos
Divide y Vencerás	NP-Difíciles
Espacios de Busqueda	No Computables

1.1.3. Evaluación de la materia

La materia para este semestre sera evaluada de acuerdo a los siguientes rubros.

I Examen 70 %

II Lista de problemas 10 %

III Practicas 10 %

IV Evaluación Practica 10 %

1.1.4. Libros que pueden ser de ayuda

- Algorithm Design - Kleinberg, Tardos
- The Nature of Computation - Christopher Moore
- Cormen
- Sipser
- Hopcroft

2. Primer parcial

2.1. ¿Qué es un Algoritmo?

Entre las ideas comunes de los que es un algoritmo estan:

Es una solución a un problema computacional mediante una Maquina de Turing que siempre se detiene (Llega a un estado de aceptación)
Es parecido a una receta: Recibe una Entrada y pasa a una Salida
Serie de pasos ordenados y estructurados
Se puede reutilizar para distintos problemas / con variantes

De acuerdo a unas aproximaciones formales podemos tener que un algoritmo es:

Solución a un problema computacional mediante una Maquina de Turing que siempre se detiene (Llega aun estado de aceptación) durante el procesamiento, parte de un estado inicial y va avanzando a través de los estados del automata
Algoritmo / Automata puede reutilizarse en diversas instancias del problema computacional (reutilización de la lógica o modulo computacional)

Entonces con esto podemos plantear algo muy importante que es:

2.1.1. Computación

Podemos definirla como un proceso natural en el que se busca una representación de un comportamiento físico, que a su vez es representado por un modelo matemático en el que se piensa puede representar un proceso, un comportamiento natural o incluso en el que se puede mostrar algo parecido a la vida.

De este modo un algoritmo acorde a esta idea debe ser:

- Eficaz: Que debe tener una resolución correcta
- Eficiente: Que debe ser capaz de realizarlo con recursos óptimos (Memoria / Procesamiento)

2.2. Complejidad Computacional

La complejidad computacional puede ser interpretada como la integración del número de recursos que utiliza un algoritmo para resolver un problema computacional.

Esta complejidad esta dividida en dos clases muy importantes:

- Temporal: Con relación al procesamiento
- Espacial: Con relación a la memoria

De este modo podemos pensar y añadir a las características de la resolución de un algoritmo:

- Eficacia: A una correctez en su ejecución
- Eficiencia: A una buena administración de sus recursos (Memoria - Espacio) (Procesamiento - Tiempo)

2.2.1. Enfoque temporal

Ahora como una primera aproximación: Asumiendo un espacio utilizado constante o manejable.

Definimos:

Función temporal:

$f(N)$: Es el número de pasos / operaciones a ejecutar.

N : Es el numero de datos de entrada.

Acorde a esto podemos pensar en que existen complejidades:

Lineal:

$$f(N) = kN; \quad k \in \mathbb{R}$$

Cuadrática:

$$f(N) \sim N^2$$

Polinomial:

$$f(N) \sim N^a; \quad a \in \mathbb{N}$$

$$f(N) \sim \alpha_a N^a + \alpha_{a-1} N^{a-1} + \dots + \alpha_2 N^2 + \alpha_1 N + \alpha_0; \quad \forall \alpha_i \in \mathbb{R} \text{ & } i \in [0, a]$$

Logarítmica:

$$f(N) \sim \log N$$

La idea de algunos enfoques temporales a nivel Logatírmicos pueden ser los *algoritmos de ordenamiento*, que graficamente su comportamiento se puede ver así:

Arreglo de ordenamiento y divisiones

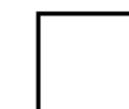
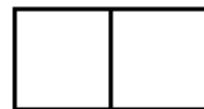
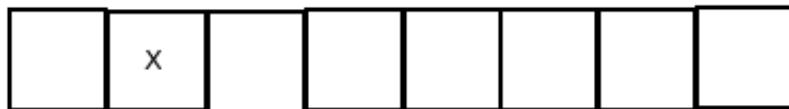


Figura 1

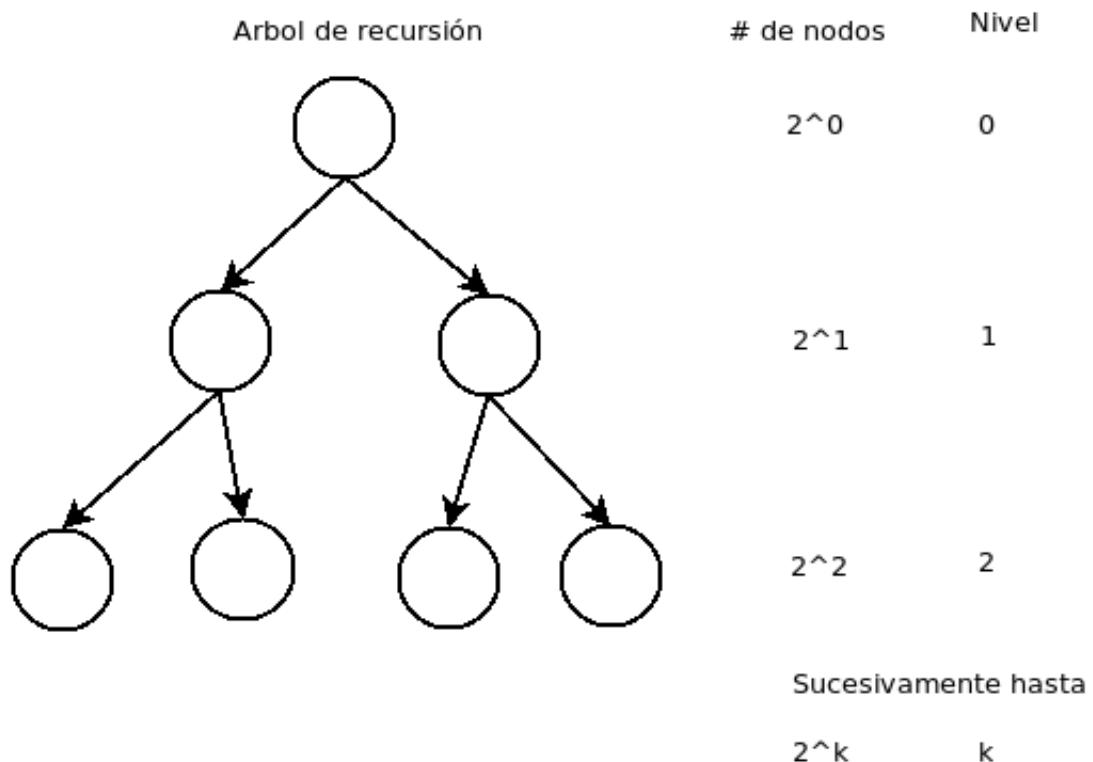


Figura 2

De este modo llevando un análisis de la Figura 1 y 2 que están relacionadas podemos decir que si se suman los elementos que existen por nivel tomando un valor discreto N siempre sumará N no importando el nivel por lo tanto de acuerdo al número de nodos que existe 2^k tal que $k \notin Z^-$ podemos decir que $k = \log_2 N$

$$\therefore 2^k = N$$

2.2.2. Ordenamiento [Introducción a clase Polinomial (P)]

De este modo podemos dar una pequeña introducción a los algoritmos de ordenamiento. Sabemos que existen algoritmos que trabajan con una colección de datos de un solo tipo [llámense estructuras de datos, clases (caso de que se esté usando el paradigma Orientado a Objetos) o variables] por lo que en algunas ocasiones es necesario realizar un ordenamiento a estos datos, por lo que se puede pensar en los algoritmos que sean sencillos de entender, por lo que más adelante algunos serán explicados a fondo de como trabajan y como se implementan a nivel pseudocódigo, por otro lado ya que se mencionó acerca de complejidades temporales los algoritmos son:

- Bubble sort: N^2
- Selection sort: N^2
- Merge sort: $N \log N$
- Heap sort: $N \log N$
- Quick sort: $N \log N$
- Aleatorio: $f(N)$ donde f es una función que puede ser representada como ecuación lineal, diferencial o de densidad probabilística.

2.2.3. Algunos otros algoritmos e ideas de problemas de la Clase NP

Si bien sabemos que a nivel computacional podemos almacenar conjuntos de bits (0 vs 1) si lo vemos asociado a direcciones de memoria o variables con capacidad de N bits podemos partir de que existe una complejidad de dar con la combinación de esos N bits de $f(N) = 2^N$ Tomando ciertas ideas de esta notación de complejidad de f podemos pensar en Grafos

\therefore Sea un Grafo $G = (V, E)$

Donde:

V = Es el conjunto de Vertices o nodos que existen

E = Es el conjunto de Aristas que se relacionan con los vertices

Si pensamos que G es completo (Es decir que cumple un modelo combinatorio)

$$|E| = \binom{N}{2} = \frac{N!}{2(N-2)!} = \frac{N(N-1)}{2} = M \quad (0)$$

Donde:

$$N = |V| \quad (1)$$

Sustituyendo 1 en 0 podemos tomar la idea y el formalismo matemático siguiente:

$$|E| = \binom{|V|}{2} = \frac{|V|!}{2(|V|-2)!} = \frac{|V|(|V|-1)}{2} = M \quad (3)$$

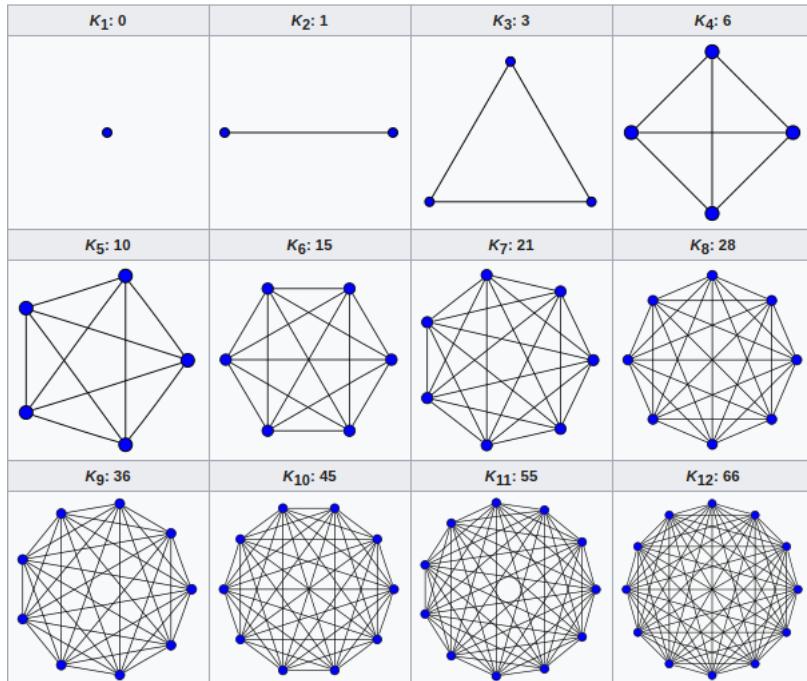


Figura de G desde $N = 1$ hasta $N = 12$ vertices
Cumpliendo 3

Ahora imaginemos el que pasaria si buscamos explorar o asemejar un Grafo completo G que va en crecimiento sucesivamente, es obvio que el hecho de ver el modelo combinatorio de G y de sus caracteristicas el hacerlo es más difícil aproximarla a un análisis Polinomial, lo cual podemos dar una pequeña introducción a la clase de problemas No Polinomiales (NP)

2.2.4. Complejidades en NP vs P

Si bien sabemos que esta complejidad representa un espacio combinatorio o exponencial algunos autores suelen expresar sus funciones de la siguiente forma.

Complejidad Combinatorial	$f(N, K) = \binom{N}{K}$
Complejidad Exponencial	$f(N, b) = b^N$

Una vez que tenemos idea de este tipo de clases de complejidad podemos pensar que pueda existir un Lema, Teorema o Transformación, en la que se pueda decir que los problemas de clase P estan contenidos o no en los problemas de clase NP $P \subseteq NP$ o $P \not\subseteq NP$ lo cual es uno de los tantos problemas que existe de **Clay Institute of Mathematics PvsNP**

Si partimos de esta idea y de lo que se intenta hacer en las ciencias de la computación podemos pensar que generalmente o implicitamente la computación busca la automatización u otro fin en el que su participación ayude a desarrollar algo del día a día por lo cual podemos pensar que la clasificación NP, es la que engloba estos problemas y conjunto de soluciones, en los que si pensamos a traves de formalismos ya existentes podemos decir que estos problemas son mapeados a cuestiones de Computación, Matemáticas y Física, lo cual puede representar un isomorfismo o transformación de ir de un tipo de solución a otra.

Esto pensando en la definición de lo qué es un algoritmo la cual representa una máquinaria en la que representa lenguajes y leyes de un universo, pensando en esto podemos hablar de un científico que logro llegar a una idea de Maquina Universal que Teóricamente abordo el matemático Alan Turing y Alonzo Church, que es la *maquina Z* de Konrad Zuse quien realmente es el padre de la computación

2.2.5. Maquina Z

Si bien conocemos la historia de la Maquina de Turing fue famosa por el hecho de generar un procesamiento bajo unas reglas de su tupla, por lo que fue famosa por el hecho de que sus avances fueron pertenecientes a la segunda guerra mundial, pero esta otra maquina que es mencionada ya que esta maquinaria genera y sigue los principios de las computadoras actuales, en el cual esta maquina era programable a nivel de código binario y al igual que operaciones a bajo nivel de bits, esta maquina podia realizar operaciones binarias, para años en los que el modelo de Turing existia para la 2^{da} guerra mundial, en esta maquina existia la Z₃, en la cual esta maquina realizaba operaciones aritmeticas, en la cual más tarde con su nuevo modelo Z₄ lograba tener una aritmética compleja, ahora que teniendo esto en cuenta podemos pensar acerca de **¿Qué representa un número complejo?**

2.2.6. Número complejo

Son aquellos números que complementan a sistemas de ecuaciones donde sus soluciones no son cubiertas al 100 % por el conjunto de números reales \mathbb{R} por lo que se extiende el espacio del conjunto de números a los complejos, los cuales son representados de la forma $z = x + iy$ donde $x, y \in \mathbb{R}$, ahora en términos de resoluciones aritméticas que realiza una computadora, podemos pensar acerca de que en la computación a nivel físico estos números intervienen en la interacción con los componentes para la transmisión y realimentación de los chips, como también en algunos otros casos en donde se procesan señales de audio o señales de pixeles para el coloreo de pantallas o renderizado de imágenes.

2.2.7. NP (Subclase NP-Completo)

Dentro de los problemas que se pueden solucionar con estos conceptos y previas ideas de matemáticas y algunas ideas implícitas de fenómenos físicos esta una subclase de los problemas NP, que son los problemas NP-Completos. Estos problemas que caen en esta clasificación son los siguientes:

Planificación	Optimización
Agrupamiento	Particionamiento
Redes	Teoría de Juegos
Círculo Hamiltoniano (Agente viajero)	Knapsack (Max/Min)

Caen dentro de esta clasificación ya que su solución a través de ideas de transformaciones lineales (matemáticamente hablando) pueden dar pie a la solución de otro problema.