




Análisis y Diseño de Algoritmos

¡ Bienvenidos !

Semestre Enero-Mayo 2019



**¿Porqué y para qué
Análisis y Diseño de
Algoritmos?**

**¿Cómo nos “divertiremos”
con este curso?**



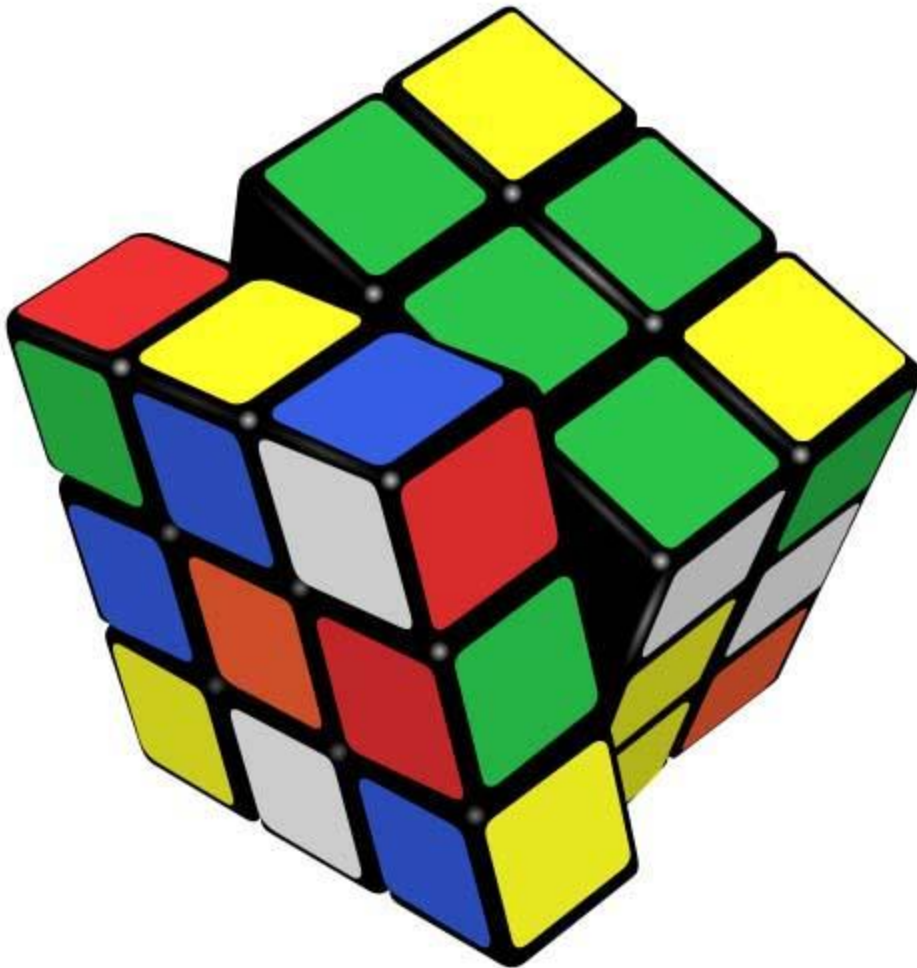
**The Soft Skills
Companies Need
Most in 2019**

Creativity
Persuasion
Collaboration
Adaptability
Time Management

**The Hard Skills
Companies Need
Most in 2019**

Cloud Computing
Artificial Intelligence
Analytical Reasoning
People Management
UX Design
Mobile Application Development
Scientific Computing
Game Development
Business Analysis
Customer Service Systems
Software Testing

Este curso tiene que ver con...



- **Problemas**
- **Soluciones**
- **Retos**
- **Eficiencia**
- **Optimización**
- **Estrategia**
- **Creatividad**



Meta del curso

Se espera que al acreditar este curso, hayas aumentado tu capacidad para resolver **eficientemente** nuevos problemas de programación, a través del conocimiento, aprendizaje y aplicación, de diversas técnicas y algoritmos, que impactarán en tu formación como **desarrollador de software**.

Objetivos generales



- Aprender a evaluar la **complejidad computacional** de algoritmos para tomar decisiones de **eficiencia**.
- Utilizar las principales técnicas de **diseño de algoritmos** para **resolver problemas**.
- Conocer los principales algoritmos que existen para **búsqueda de información**, manipulación de **grafos** y algunos **algoritmos numéricos**.
- Conocer los conceptos más relevantes relacionados con el **cómputo paralelo** para aplicarlos cuando el contexto lo permita.

I AM THE
LEFT BRAIN

Decisive!
011001011 LOGIC

Accurate
ANALYTIC

REASON

1 2 3 4 5 6 7 8 9

PRACTICAL
Strategic

CONTROL

SCIENCE

Realistic
Brain

WWW.CARTOONADAY.COM

I AM the
Right BRAIN!

Intuition
Love LOVE Love
Love thou art
Poetry

FREEDOM

Passion
vivid

creative

YEARNING

PEACE



¿Cómo impacta un algoritmo en la solución de un problema?



EJEMPLO: Caso de la multiplicación

- ¿De cuántas maneras se puede obtener la multiplicación de 2 números?
 - ✓ Con calculadora
 - ✓ Como nos enseñaron en primaria
 - ¿Habrá más formas?
- Multiplica **981 X 1234**

981 X 1234 (*à la russe*)

981	1,234	1,234
490	2,468	
245	4,936	4,936
122	9,872	
61	19,744	19,744
30	39,488	
15	78,976	78,976
7	157,952	157,952
3	315,904	315,904
1	631,808	631,808
	Σ	1,210,554

Temario



1. Introducción al Análisis de Complejidad

2. Divide y vencerás.

3. Programación dinámica.

4. Algoritmos voraces.

5. Backtracking.

6. Brach and Bound.

7. Algoritmos de tópicos selectos

8. Introducción al cómputo paralelo.

9. Clasificación de Algoritmos

**Diseño de
algoritmos**

Temario:

Problemas que se estudiarán



- Búsqueda
- Ordenamiento
- Algoritmos numéricos y matrices
- Grafos
 - Camino más corto
 - Arbol de extensión mínima
- Problema del viajero
- Problema de la mochila
- Heaps
- etc.

Características adicionales del curso...



- La clave del éxito en este curso es **la práctica**....
- ...**conceptual** y a través de la **programación**...
- Nos apoyaremos en muchos recursos que hay en Internet...
- ...**bancos de problemas** y **jueces en línea**...



Every problem
has a solution.
You just
have to be creative
enough to find it.

Travis Kalanick, Uber co-founder



**¿Te gustaría ser contratado
por una importante empresa
de Tecnología?**

Reto del semestre

en Análisis y Diseño de Algoritmos

Semestre Enero-Mayo 2019

¿Qué buscan las empresas de Tecnología para contratarte?



Entre otras cosas, tiene un peso importante tu **habilidad para resolver problemas con programación...**

Esto implica:

- Saber **programar** y hacerlo **eficientemente**
- Dominar uno o varios **lenguajes de programación**
- Diseñar y utilizar eficientemente **estructuras de datos**
- Aplicar **técnicas de diseño de algoritmos**



**... y en las entrevistas de trabajo,
normalmente te ponen a prueba
con el reto de resolver un pequeño
problema y programar su solución...**



**En nuestro curso,
incrementaremos las habilidades
para enfrentar este tipo
de entrevistas de trabajo...**



- **Ya sabes programar (tienes un pensamiento algorítmico)...**
 - En el curso fortaleceremos el hacerlo más eficientemente.
- **Ya conoces y utilizas un lenguaje de programación...**
 - En el curso ampliaremos el nivel de dominio de C++.
- **Ya conoces las estructuras de datos básicas y sabes diseñar nuevas...**
 - En el curso profundizaremos en su uso y ampliaremos sus aplicaciones.
- **...y aprenderás nuevas técnicas de diseño de algoritmos, teniendo siempre el contexto de la estrategia y eficiencia en la solución.**

El reto en una entrevista es...

It's your turn to be the Interviewer. The question is available in the left pane below the video



Language: Python ▼

Swap Roles

End Interview

```
1 # logs storage
2 M = [[0 for x in range(n)] for x in range(n)]
3
4
5 # Reset logs
6 def resetLogs():
7     for i in range(0,n):
8         for x in range(0,n):
9             M[i][j] = 0
10
11
12 # Log a call between 2 users
13 def logCall(source, destination):
14     M[source][destination] = M[source][destination] + 1
15
16
17 # get the amount of calls between 2 users
18 def getCallCount(source, destination):
19     print (M[source][destination])
```

The "Call Logger System Design" Problem

You are asked to plan a call log system that logs info about calls between n users.

The system should support the following interface:

- Log a call from user U_x to user U_y .
- Figure how many calls, if at all a given user U_x have made to a given user U_y
- Provide the list of all users that a given user U_x has called to
- Reset all logs and memory of the system

Design the system, choose appropriate data structures and implement the code to support the interface above. Analyze the runtime and space complexity of each interface method you implement.

Solution

The data could be represented by using a 2D matrix of integers of size $n \times n$.

The integer value on a matrix cell $M[x,y]$ will represent the number of calls that user U_x has made to user U_y .

Reset action will be done by writing zeros into all cells. This action will also be done when the system is first initiated:

```
function resetLogs():
    for i from 0 to n-1:
        for j from 0 to n-1:
```

Tips

- Encourage your peer to optimize for of runtime and space complexity combined
- If your peer is stuck, give an example of few calls and ask what would the interface actions do on this data set. Then ask how can you represent this data
- If you peer is still stuck, ask them to choose the most basic representation (even an array for every user on the system) and start improving it

- Proponer una **estrategia eficiente de solución al problema...**
- e **implementarla utilizando las herramientas adecuadas:** algoritmos, estructuras de datos y lenguaje de programación...
- En un **lapso de tiempo**, normalmente no mayor a las **2 horas...**





- Lo que más cuenta para impresionar y conseguir el trabajo es **resolver el problema correctamente**, y tener una **buena estrategia en el proceso de solución...**



**KEEP
CALM
AND BE A
PROBLEM
SOLVER**

Nuestro reto en el semestre...

RESOLVER PROBLEMAS COMO EN UNA ENTREVISTA DE TRABAJO

- En la penúltima semana del semestre, tendremos el **reto final** de resolver un problema de una empresa real...
- Quienes hayan hecho méritos durante el semestre, serán invitados a participar en el reto final...
- Habrá **3 rondas en el semestre** para practicar y generar los méritos para ser invitado al reto final...
- En cada ronda tendremos 2 momentos:
 1. **Momento de PRÁCTICA**, en donde habrá que resolver **5 problemas** antes de una fecha límite. Hay que demostrar que se resolvieron correctamente los 5 problemas para pasar al siguiente momento.
 2. **Momento de SIMULACIÓN**, en donde habrá **un problema a resolver en casa en una fecha y horario específico** (2 horas). Hay que resolver correctamente el problema para pasar a generar el mérito de participar en el reto final. Durante la prueba, se podrá tener entrevista por parte del profesor.
- El **reto final**, tendrá **un problema a resolver presencialmente** en salón especificado, en una fecha y horario específico (2 horas).

Temáticas y fechas de cada RONDA de práctica para el reto

- **Ronda #1**

- Problemas relacionados con **estructuras de datos lineales y algoritmos de búsqueda y ordenamiento básicos.**
- Fecha límite para entregar problemas de práctica: 5 de febrero
- Momento de simulación: jueves 7 de febrero

- **Ronda #2**

- Problemas relacionados con **estructuras de datos jerárquicas, recursividad, divide y vencerás y programación dinámica.**
- Fecha límite para entregar problemas de práctica: 5 de marzo
- Momento de simulación: jueves 7 de marzo

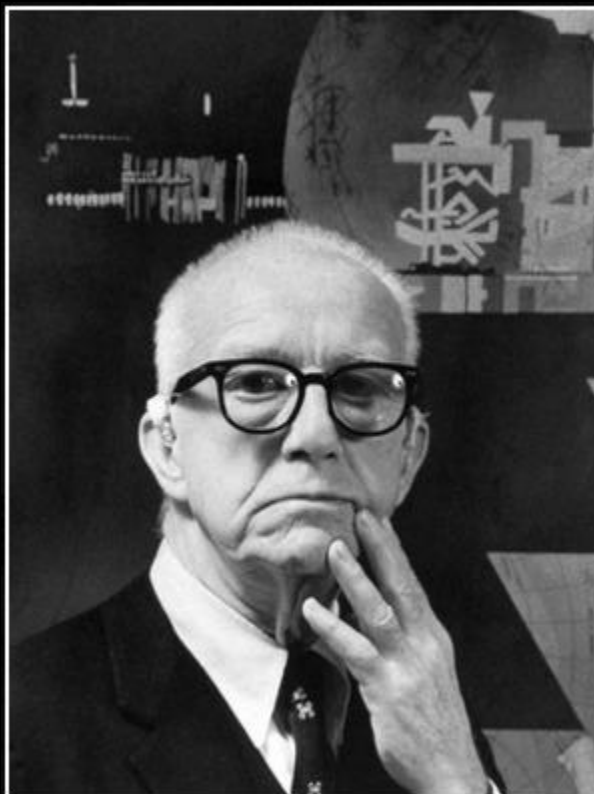
- **Ronda #3**

- Problemas relacionados con aplicaciones de **grafos, algoritmos voraces, y backtracking.**
- Fecha límite para entregar problemas de práctica: 9 de abril
- Momento de simulación: jueves 11 de abril

- **Reto final:** viernes 26 de abril (sesión presencial a las 18:30 hrs).

Valor del reto...

- El valor principal está en el desarrollo de habilidades para cuando tengas una entrevista de trabajo...
- ...así como en la práctica y conexión de lo que ya sabes, con lo nuevo que aprenderás en el curso...
- **5% de la calificación final** se obtiene con los puntos de este reto:
 - En cada una de las 3 rondas:
 - Se obtienen **0.2 puntos por cada problema resuelto** con las condiciones indicadas en el momento de práctica.
 - Si se entregan 5 problemas resueltos correctamente del momento de práctica, se podrá participar en el momento de simulación, el cual tiene un valor de **2 puntos**, si se logra resolver el problema asignado.
 - Los méritos para ser invitados a la sesión del reto final son los siguientes:
 - Haber resuelto correctamente al menos un problema de los momentos de simulación de cualquier ronda, y
 - Tener acumulados **al menos 4 puntos en las rondas**.
 - En el reto final, se obtendrán **3 puntos** si se logra resolver el problema.
- Con esta política de evaluación, quien participa en todas las oportunidades de las rondas y el reto final, podrá acumular un máximo de 12 puntos, siendo 7 los puntos extras posibles a obtener en la calificación final.
- Después de esta experiencia e independientemente de los puntos que logres, sin duda serás mejor solucionador de problemas con programación... y esto nadie te lo quita...



The reward for being a good
problem solver is to be heaped with
more and more difficult problems to
solve

— *R. Buckminster Fuller* —

AZ QUOTES

¡Cuidado!
este reto te hará
también más
competitivo para los
concursos de
programación...

Sitios de referencia y en donde se trabajará para comprobar soluciones

- www.hackerrank.com
- leetcode.com

Políticas de evaluación



Exámenes parciales

- 5 Conceptuales **25%**
- 2 Prácticos de programación **20%**

Examen final **25%**

Tareas y proyectos **25%**

Reto semestral **5%**

Estrategia *Mastery Learning* en el curso



- Los exámenes conceptuales parciales podrán presentarse hasta obtener una calificación de 100.
- Se presentan de nuevo en presencia del profesor después de revisar los errores cometidos.
- Las oportunidades son bajo previa cita, o en las fechas que se indique previo a la sesión del curso.
- **El derecho a presentar de nuevo un examen conceptual caduca al momento en que se presenta el siguiente examen.**
- Se considera para la calificación final la última calificación obtenida.

Otras políticas



- Fechas de exámenes intrasferibles.
- Entrega de tareas:
 - Por BB cuando se indique, o impresas en el salón de clases.
 - No es necesario portada y se pueden utilizar hojas recicladas. Utilizar encabezado con datos esenciales: **Nombre, matrícula, #Tarea, Fecha, Curso y profesor.**
 - Penalización del 20% por día de retraso.
- Reglamento de faltas a la **Integridad académica.**
- Asistencia:
 - Se tomará lista la mayoría de las sesiones de clase.
 - Falta con retardo de más de 20 minutos.
 - Impacto en el desempeño.

Conocimientos previos...



Programación estructurada y orientada a objetos:

- Estructuras de control: secuencia, decisión y repetición.
- Modularidad: uso de funciones, parámetros por valor y variables.
- Arreglos unidimensionales y bidimensionales.
- Archivos de texto.
- Recursividad.
- Objetos: definición de clases, métodos, atributos, mensajes, STL, etc.

Conocimientos previos...



Estructuras de datos :

- Listas encadenadas, pilas y filas.
- Arboles binarios de búsqueda.
- Hashing.
- Algoritmos de ordenamiento.
- Grafos


Matemáticas:

- Límites
- Series sencillas
- Combinatoria básica
- Álgebra matricial.

Herramientas a utilizar



- **Blackboard:** Contenidos oficiales, medio de entrega de tareas y de exámenes.
- **Facebook:** grupo privado para hacer comunicados constantes, compartir ideas, resolver dudas y registrar reportes de aprendizajes.
- **Videos** como evidencia de aprendizajes y como material de apoyo.
- **Ambiente de programación**
- **Excel:** simulaciones
- **Mentimeter, Kahoot y Google forms:** Encuestas y quizzes



Grupo en FB

Análisis y Diseño de Algoritmos – EM19

facebook.com/groups/ADAGrupo4MTYEM19/

TAREA #1



1. Encuesta
2. Repaso – examen de diagnóstico
3. Lectura de conceptos iniciales
4. Documental para reflexión y motivación
5. Investigación de sitios de apoyo
6. Video como evidencia, testimonio y presentación en FB:
Análisis y Diseño de Algoritmos – EM19



Encuesta inicial

<https://goo.gl/forms/2ZCTsHOatr6DFWPu1>



¡ Programar es un arte !

Are you a

**PROBLEM
SOLVER**

or a

**SOLUTION
CREATOR?**