PROGRAMA DE ASIGNATURA	
ASIGNATURA: Matemática Discreta II	**AÑO:** 2021
CARACTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 3° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación	
REGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTACIÓN Y OBJETIVOS

Esta es una materia que si bien se llama Matem'atica Discreta II el nombre real deberia ser Algoritmos y Estructuras de Datos III mezclados con Matem'atica Discreta y Teoria de C'odigos de Correcci'on de Errores y Teori'a de Complejidad mas un toque de inteligencia artificial.

La parte principal de la materia es el estudio de algoritmos sobre grafos y networks, y especialmente el analisis de la correcci'on y las complejidades de los mismos para que los estudiantes comprendan que en muchas aplicaciones no basta dar un algoritmo sino que hay que demostrar su correctitud, dar una cota de su complejidad y demostrarla.
Esto es una mezcla de algoritmia y matem'atica.
El proyecto de programaci'on se basa en esta parte para aprender las dificultades de traspasar elementos te'oricos a codificaciones concretas.

Ademas de esta parte central la materia tambien cubre c'odigos de correcci'on de errores porque es un tema que necesitan en la materia Redes,
cubre P-NP porque encaja bien con la primera parte viendo que a veces no hay algoritmos polinomiales para resolver problemas y Algoritmos Gen'eticos porque a'un en esos casos igual algo hay que hacer para resolver los problemas.

CONTENIDO

Grafos y Coloreo

Repaso de la noción de grafo y otras cosas.
Notaciones.
Coloreo de Grafos.
Numero cromático.
Algoritmo de fuerza bruta.
Problema k-Color.
Definición de bipartito.
Conectitud.
Componentes conexas.
Repaso de BFS y DFS.
Algoritmo polinomial para determinar biparticidad
Propiedad: un grafo es bipartito si y solo
si no tiene ciclo impares.
Algoritmo Greedy de Coloreo.
Ejemplos de aplicaci'on.
Ejemplo de que no siempre Greedy devuelve el n'umero crom'atico.
Ejemplo de que tan mal puede diverger.

==========

Very Important Theorem: (central para el proyecto)
Sea G=(V,E) un grafo cuyos vértices estan coloreados con un coloreo propio c con r colores {0,1,...,r-1}.

Sea P una permutación de los números 0,1,...,r-1.

Sea V[i]={x en V tal que c(x)=i}, i=0,1,...,r-1.

Ordenemos los vértices poniendo primero los vértices de V[P(0)], luego los de V[P(1)],etc, hasta $V[P(r-1)]
Entonces Greedy en ese orden colorear'a G con r colores o menos.
==========
Propiedad: El número cromático es menor o igual que Delta +1.
Ejemplos donde se alcanza la cota.

Teorema de Brooks. (con prueba en ambos casos: no regular y regular)

Fundamentos de inteligencia artificial

Algoritmos de Busqueda.
Hill Climbing.
Simulated Annealing.
Algoritmos Genéticos:
Codificación del problema.
Fitness.
Reproducción de Población.
Terminación.

Selección, Crossover, Mutación, Reemplazo.

Algunas posibilidades de Mutación.

Algunos posibilidades de Crossover.
Single point, double, multiple points o mascara.
Crossover en el caso de permutation based codifications: crossover b'asico, Partial Mixing Crossover y Ciclico.

Algunas posibilidades de Selección:
Ruleta, SUS, Rank-based selection
Sigma based selection.

Otra posibilidades de estructura: catástrofes e islas. con migraciones.

Flujos Maximales

Grafos Dirigidos.
Ejemplos.
Networks.(redes)
Flujos sobre redes.
Valor de un flujo.
Flujos maximales.
Diversos ejemplos.
Algoritmo Greedy para encontrar flujo maximal.
Ejemplo donde no necesariamente encuentra flujo maximal.
Definici'on de corte y capacidad de un corte.
Caminos aumentantes de Ford-Fulkerson.
Algoritmo de Ford-Fulkerson.
Propiedad: Al aumentar el flujo a lo largo de un camino aumentante de Ford-Fulkerson lo que se obtiene sigue siendo flujo.

Max Flow Min Cut Theorem:

a)El valor de todo flujo es menor o igual que la capacidad de todo corte.
b)Si f es un flujo, las siguientes afirmaciones son equivalentes:
1) f es maximal.
2) Existe un corte S tal que v(f)=cap(S).
3) No existen f-caminos aumentantes.

Ejemplos de aplicacion
del algoritmo de Ford-Fulkerson.
Debilidades del algoritmo de Ford-Fulkerson:
Ejemplo donde la complejidad no depende del numero de vertices o lados.
Ejemplo donde el algoritmo no termina.
Refinamientos:
Algoritmos fuertemente polinomiales:
Algoritmo de Edmonds-Karp. Complejidad.
Algoritmo de Dinic o Dinitz. Complejidad de sus 2 versiones.
Algoritmos de pre-flow/push: algoritmo ``wave'' de Tarjan. Complejidad.

BIBLIOGRAFÍA

BIBLIOGRAFÍA BÁSICA

Matem'atica Discreta. N. Biggs,1989

Applied Combinatorics. Roberts, 1989, Prentice-Hall.

Data Structures and Network Algorithms. R.E. Tarjan, 1983, Society
for Industrial and Applied Mathematics.

\Computers and Intractability: A Guide to the Theory of NP-completeness.
Garey and Johnson, 1979, Bell Telephone Laboratories.

.pdfs hechos por Daniel Penazzi y entregados a los alumnos.

Combinatorial Optimization: Algorithms and Complexity.
Papadimitriou-Steiglitz, 1998, Dover Publications.

BIBLIOGRAFÍA COMPLEMENTARIA

Applied Combinatorics. A. Tucker, 2nd Ed., 1984

Network Flows: Theory, Algorithms and Applications.
Ahoja-Magnani-Orlin, 1993, Prentice-Hall.

EVALUACIÓN

FORMAS DE EVALUACIÓN

Los alumnos deberan aprobar las siguientes partes:

1) Una parte teorica, que se tomara en el final.
Se tomara en forma virtual salvo que la situacion de la facultad y del pais permita tomarla en forma presencial.

2) Una parte practica que se tomara en el final.
Idem al punto anterior.

3) Una parte de programacion, que se tomara a lo largo del curso (obviamente en forma virtual).
Los alumnos deberan entregar un proyecto de programacion cuyas especificaciones estaran dadas en la pagina del Aula Virtual.

El proyecto ES UNO SOLO pero vendra dado EN VARIAS ETAPAS para evitar el problema detectado en años anteriores que los alumnos dejaban todo el proyecto hasta que solo faltaban pocos dias para entregarlo y luego no llegaban a tiempo.
De esta forma aun si dejan todo para ultimo momento en cada etapa tendran mas posibilidades de completar la etapa.
Cada etapa del proyecto se apoya en funciones programadas en las etapas anteriores.
De todos modos una parte importante del proyecto es "specification driven programming": cada funcion definida en el proyecto tiene sus especificaciones que deben cumplirse de forma tal que se deben programar las funciones pej de la etapa 2 del proyecto haciendo calls a las funciones de la etapa 1 del proyecto, independientemente de como han sido programadas.
De esta forma se pueden evaluar las funciones de la parte 2, pej, haciendo calls a MIS funciones de la parte 1 y no la de los estudiantes.
Esto evita "propagacion de errores" y que errores en la parte 1 afecten a la correccion de las siguientes partes.
De todos modos el proyecto es estructuralmente uno solo, con un tema principal y la nota del proyecto ES GLOBAL, las distintas etapas no tienen notas parciales.

REGULARIDAD
Aprobar al menos el 60% de los Trabajos Prácticos o de Laboratorio.
Como hay un solo proyecto, deberán aprobarlo.

PROMOCIÓN
La materia no se promociona.