PROGRAMA DE ASIGNATURA

ASIGNATURA: Matemática Discreta II	**AÑO**: 2012
CARÁCTER: Obligatoria	
CARRERA: Licenciatura en Ciencias de la Computación	
RÉGIMEN: cuatrimestral	**CARGA HORARIA:** 120 hs.
UBICACIÓN en la CARRERA: Tercer año – Primer cuatrimestre	

FUNDAMENTACIÓN Y OBJETIVOS

Que los alumnos adquieran:

- experiencia en el desarrollo de algoritmos complejos y en el análisis de su complejidades.
- Comprensión de conceptos de flujos maximales, matchings, complejidad computacional, P-NP, códigos de corrección de errores, importancia de algoritmos polinomiales.

CONTENIDO

Unidad I: Coloreo

Repaso de la noción de grafo. Notaciones. Coloreo de Grafos. Número cromático. Algoritmo de fuerza bruta. Problema *k*-Color. Algoritmo 1-COLOR. Definición de bipartito. Repaso de BFS.
Propiedad: un grafo es bipartito si y solo si no tiene ciclo impares. Algoritmo polinomial para 2-COLOR. Algoritmo Greedy de Coloreo. Ejemplo de aplicación. Ejemplo de que no funciona. Propiedad: $\chi \leq \Delta + 1$. Ejemplos de que se alcanza la cota. Algoritmo de Brooks. Teorema de Brooks (sin prueba).

Unidad II: Flujos
Grafos Dirigidos. Ejemplos. Networks. Flujos. Propiedad: *out(s) – in(s) = in(t) – out(t)*. Valor de un flujo. Definición de corte y capacidad de un corte. Propiedad: *v(f) = f(S,S) - f(S,S).* Corolario: el valor de cualquier flujo es menor o igual que la capacidad de cualquier corte. Corolario: Si el valor de un flujo es igual a la capacidad de un corte entonces el flujo es maximal y el corte es minimal. Teorema : Existen flujos maximales. Algoritmo “Naive” o “Greedy” para encontrar un flujo maximal.

Ejemplo donde funciona. Ejemplo donde falla. Como modificar el algoritmo para que funcione. El Teorema del flujo maximal-Corte minimal (Max-Flow-Min-Cut Theorem). Algoritmo de Ford-Fulkerson. Ejemplos de aplicación del algoritmo de Ford-Fulkerson. Debilidades del algoritmo de Ford-Fulkerson: ejemplo donde la complejidad no depende del número vértices o lados. Ejemplo donde el algoritmo no termina. Refinamientos: Algoritmos fuertemente polinomiales: Algoritmo de Edmonds-Karp. Complejidad. Algoritmo de Dinic. Complejidad. Algoritmos de pre-flow/push: algoritmo “wave”. Complejidad.

Unidad III: Matchings
Matchings en grafos bipartitos. Matchings perfectos y Matchings completos. Ejemplos. Algoritmos para encontrar matchings como aplicación de los algoritmos para encontrar flujos maximales. Modificaciones. Uso de matrices. Definición de Γ (*S*). Condición de Hall. Teorema de Hall. Teorema del matrimonio. (Todo grafo bipartito regular tiene un matching perfecto). Problemas de Matchings Óptimos en grafos bipartitos con pesos. Resolución de “bottleneck problem”: problema del asignamiento óptimo cuando se desea minimizar el máximo (o maximizar el mínimo) de los pesos.
Resolución del problema del asignamiento óptimo cuando se desea minimizar (o maximizar) la suma de los pesos: Algoritmo Húngaro. Codificación de complejidad $O(n^3)$ del algoritmo Húngaro.

Unidad IV: P y NP
La clase P. La clase NP. Ejemplos. El problema SAT. El problema *k*-COLOR. Ejemplo: 2-COLOR es P. Reducción polinomial. Reducción de 3-COLOR a SAT. Las clases de problemas NP-hard y NP-completo. Teorema de Cook (sin prueba): SAT es NP-completo. Teorema: 3-SAT es NP-completo. 2-SAT está en P. Horn-SAT está en P. Teorema: 3-COLOR es NP-completo.

Unidad V: Tópicos de Inteligencia Artificial
Algoritmos de búsqueda. Hill Climbing. Simulated Annealing.
Algoritmos genéticos: Codificación del problema. Fitness. Reproducción de Población. Terminación.
Elementos de la Reprodución: Selección, Crossover, Mutación, Reemplazo.
Algunas posibilidades de Selección para Reproducción: Roulette, SUS, Rank-based selection.
Algunas posibilidades de Crossover. Single point, double or multiple points.
Algunas posibilidades de Mutación. Algunas posibilidades de Reemplazo: Ambos progenitores, Progenitor más débil, Individuos más débiles, Randmo. Ventajas y desventajas.
Arquitectura alternativa a la tradicional: Steady State.
No Free Lunch Theorem. Ejemplos de aplicación.

Unidad VI: Códigos
Definiciones básicas. Distancia de Hamming. Detección de errores. Corrección de

errores. Ejemplos de códigos. Chequeo de paridad. Códigos de repetición. Cota de Hamming. Códigos lineales. Propiedad: *C* lineal entonces δ (*C*) es igual al mínimo peso no nulo. Matrices Generadoras. Códigos lineales como espacios fuilas de una matriz. Códigos lineales como núcleos de matrices. Matrices de chequeo. Equivalencias entre matrices generadoras y de chequeo. Propiedad: todo código lineal tiene un matriz de chequeo. Proposición: Si en la matriz de chequeo no hay columnas repetidas ni nulas entonces el código correspondiente corrige al menos un error. Algoritmo para corregir un error. Códigos de Hamming. Códigos perfectos. Propiedad: Hamming es perfecto. Propiedad: δ (*C*) es igual al menor número de columnas linealmente dependientes de una matriz de chequeo. Singleton Bound. Códigos MDS.

Unidad VII: Códigos Cíclicos
Rotación de una palabra. Códigos cíclicos. Códigos cíclicos mirados como polinomios. Propiedad: todo código lineal (sobre $\mathbb{Z}_2$) tiene un único polinomio no nulo de menor grado. Polinomio generador de un código cíclico. Propiedades del polinomio generador. Uso del polinomio generador para codificación: dos métodos. Matrices generadoras asociadas a los dos métodos. Obtención en forma directa a partir del polinomio generador de una matriz de chequeo con la identidad a izquierda. Polinomio chequeador. Corrección de errores: error trapping.

Unidad VIII: Códigos de Reed-Solomón
Cuerpos finitos. Definiciones y construcción. Propiedades básicas de los cuerpos finitos. Teorema del elemento primitivo. Códigos de Reed-Solomon. Teorema: los códigos de Reed-Solomon son MDS.

BIBLIOGRAFÍA

BIBLIOGRAFÍA BÁSICA

- Matemática 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 de algunos temas en página de Daniel Penazzi.

BIBLIOGRAFÍA COMPLEMENTARIA

- Applied Combinatorics. A. Tucker, 2nd Ed.,1984.
- Network Flows: Theory, Algorithms and Applications. Ahoja-Magnani-Orlin, 1993, Prentice Hall.
- Combinatorial Optimization: Algorithms and Complexity. Papadimitriou-Steiglitz, 1998, Dover Publications.

METODOLOGÍA DE TRABAJO

La asignatura se organiza en clases teóricas y prácticas, de cuatro horas reloj cada una.
Las clases teóricas son expositivas, y las clases prácticas se organizan en comisiones donde los alumnos resuelven de manera independiente o grupal ejercicios prácticos, bajo la supervisión y acompañamiento de los docentes.
Además los alumnos implementan un proyecto en grupos de no más de 5 estudiantes, en su propio tiempo, bajo la orientación y supervisión de los docentes.

EVALUACIÓN

FORMAS DE EVALUCIÓN

- Se tomarán dos parciales (solo prácticos) más un recuperatorio para los que hayan aprobado uno solo.
- Se deben entregar proyectos de programación, el número y el tipo especificados en clase.
- El examen final consta de una parte práctica y una parte teórica, ambas valen 50% de la nota y deben aprobarse por separado. Los alumnos libres deben hacer un ejercicios extra, que vale -1 punto si no se hace o se hace mal. Los alumnos que no aprueben el proyecto deben hacer otro ejercicio extra más, que vale -7 puntos si no se hace o se hace mal.

CONDICIONES PARA OBTENER LA REGULARIDAD

1. EXÁMENES PARCIALES

- Aprobar los dos parciales con al menos 4 (cuatro), o uno de ellos con al menos 4 (cuatro) y aprobar el recuperatorio.

2. PROYECTO DE LABORATORIO

- Aprobar un número mínimo de proyectos.