Cátedras
Matemática Discreta y
Algoritmos y Estructura de Datos

[TRABAJO PRÁCTICO ANUAL Matemática Discreta – Algoritmos y Estructura de Datos]

Trabajo práctico de carácter anual para alumnos que cursen las materias Matemática Discreta y Algoritmos y Estructura de datos durante el ciclo lectivo 2016

Contenido

Instructivo para Matemática Discreta

Organización de los grupos

Cada grupo deberá auto-gestionarse, determinando un representante que se ocupará de las tareas administrativas del grupo. El código único de grupo se forma mediante el siguiente formato: KCCCCXX, donde K es el prefijo de sistemas, CCCC es el número de curso y XX es el identificador único de grupo dentro del curso, valor determinado por el docente de cada grupo. Es tarea del representante gestionar las modificaciones en el grupo, enviar el trabajo práctico para su corrección y recibir los resultados de la misma. Como la tarea se realizará en conjunto entre las cátedras Matemática Discreta y Algoritmos y estructura de datos los grupos serán organizados en Matemática Discreta aun cuando no todos los integrantes pertenezcan al mismo curso en algoritmos y estructura de datos.

Consultas sobre el trabajo práctico

Para consultas puntuales, puede utilizarse el foro disponible en el campus virtual. Por favor, no revelar resoluciones para otros grupos, intentar que las consultas sean de concepto.

Contenido del trabajo práctico

El trabajo práctico consta de cuatro secciones, cada una de ellas correspondiente a un tema que se irá dando en la materia Matemática Discreta a lo largo del curso. En cada sección hay varias tareas con su correspondiente puntuación. Una tarea correcta, sumará los puntos que allí se indica.
Además de las tareas 'teóricas', se agregan tareas para programadores. Aquellos grupos que estén cursando Algoritmos y Estructura de datos, deberán realizar las tareas para el programador como requisito del cumplimiento del trabajo práctico de esa asignatura.
La condición de aprobación para CADA sección es obtener 4 puntos sumados con todas las tareas presentadas que estén correctas.
Por lo tanto, deberán elegir las tareas que prefieran y que sumen al menos los 4 puntos solicitados, y presentarlas para su corrección.

Entrega del trabajo práctico

Una vez que el equipo tenga el trabajo práctico terminado, lo deberá subir como tarea al espacio del Aula generado a tal fin. En el foro de novedades se publicarán las fechas de entrega oportunamente.
Indicar en el asunto: IDENTIFICACION DEL CURSO – VERSION (el número de versión cambia con cada corrección, empieza en la v1).
El/Los archivo/s adjuntos deben estar precedidos por una carátula que indique explícitamente los integrantes ACTUALES del grupo. Ante la aprobación del trabajo práctico, recibirán un código de aprobación único que deberán traer impreso junto con la lista final de integrantes al momento de firmar la materia.

Publicación de notas

Mediante el código de aprobación que reciban, se podrá chequear la nota obtenida en una planilla de google drive que se publicará en el campus virtual apenas se haya realizado la corrección.

Instructivo para Algoritmos y Estructura de Datos

Metodología de Inscripción de grupos

- **¿Estás cursando Matemática Discreta?** Se utilizarán los mismos grupos. La inscripción estará a cargo de la Ing. Daniela Bello.
- **¿Cursaste Matemática Discreta y tenés las secciones "Para programadores" del Trabajo Práctico desarrolladas?** Envíanos tu trabajo práctico a tp.aed.utnba@gmail.com, se evaluará cada situación en particular.
- **¿Estás cursando solo Algoritmos y Estructura de Datos?** La inscripción de los grupos la realizarán a través del formulario http://goo.gl/forms/BJMGzNAv9n hasta el 31/05. Los grupos deben estar integrados de 8 a 10 estudiantes que pueden pertenecer a distintos cursos. Por cada grupo se debe designar un responsable que estará a cargo de la comunicación con la Cátedra.

Entregas del Trabajo Práctico

- **Primera Entrega:** Sección 1 y Sección 2. Fecha Límite 11/07. Desde el 12/07 hasta el 26/07 se realizará la corrección y se comunicará el resultado. Luego disponen de 2 semanas para realizar las correcciones necesarias.
- **Segunda Entrega:** Sección 1 y Sección 2. Fecha Límite 17/10. Desde el 18/10 hasta el 01/11 se realizará la corrección y se comunicará el resultado. Luego disponen de 2 semanas para realizar las correcciones necesarias.
- Si desean hacer la entrega antes de la fecha límite contarán con más tiempo para realizar correcciones.

Condiciones de Entrega

- Los archivos adjuntos deben estar precedidos por una carátula que indique explícitamente los integrantes ACTUALES del grupo.
- Se deberá incluir en la entrega el ejecutable del programa y un documento donde se incluya
    - Comentarios sobre la resolución
    - Pequeño instructivo de uso
    - Capturas con pruebas del programa funcionando
    - El diagrama de Lindsay de cada una de las secciones
    - Códigos fuentes en archivos ".cpp"

Condiciones de Aprobación

- Desarrollar correctamente todas las secciones "Para Programadores" del Trabajo Práctico.
- Una vez que se encuentren aprobadas las entregas grupales del trabajo práctico se procederá con una defensa INDIVIDUAL del mismo. Se fijará más de una fecha hacia el final del ciclo lectivo para rendir esta defensa.

Aclaraciones

- El lenguaje que se usará para desarrollar cada una de las secciones es C++.
- Hasta nuevo aviso las comunicaciones se realizarán vía e-mail - tp.aed.utnba@gmail.com
- Cuando se envíe un e-mail en el asunto colocar la IDENTIFICACIÓN DEL GRUPO (se les asignará cuando termine el período de inscripción)

Sección 1: Lógica

Presentación

El objetivo de esta sección es consolidar los conocimientos vistos en la lógica proposicional, ampliándolos y dejando lugar a la investigación. Los puntos teóricos incluyen utilizar temas cotidianos y poder pasarlos al campo de la lógica. Por otro lado, los puntos para programadores buscan hacer representaciones de proposiciones compuestas en computadoras (circuitos lógicos usados).

Tareas

- ✓ Buscar o generar dos silogismos lógicos reales (o sea, de uso cotidiano), estudiar sus valores de verdad, formalizarlos e indicar conclusiones (si se trata de uno real, indicar por qué; si es un 'falso silogismo' indicar cuál es el punto de fallo) → **2 puntos**.

- ✓ Generar reglas lógicas (proposiciones) que a partir de un hecho (proposición de valor conocido), permita llegar a otro hecho[1]. Este sistema de reglas, debe resolverse mediante deducción natural (Modus Ponens, Eliminación de la Conjunción, etc.) → **2 puntos**.

- ✓ Investigar sobre Deducción Automática (la Teoría de Herbrand), elaborar ejemplos propios e intentar explicar con términos propios el propósito de esta teoría → **2 puntos**.

- ✓ Investigar sobre Deducción Natural (uso de reglas Modus Ponens, Eliminación de la Conjunción, de la Disyunción, etc.). Generar ejemplos y explicar con términos propios las aplicaciones de esta teoría → **2 puntos**.

Tip: para ver ejemplos de los **silogismos**, consultar la parte I del TP de la materia 2013, almacenado en el campus virtual. Sobre la tarea de **reglas lógicas**, podrán encontrar ejemplos y aplicaciones en los TP de 2012 y 2013. En el 'Anexo' del Punto I del TP de 2012, encontrarán un apunte simplificado muy útil para resolver los problemas que requieran **deducción natural**.

Para Programadores

- ✓ Conociendo que en algoritmos se trabaja con los fundamentos de programación modular y estructurada, que junto a la POO conforman el paradigma imperativo. Sabiendo además que los problemas computacionales aceptan otro tipo de solución se pide:
  - Investigar que se entiende por paradigmas declarativo.
  - Determinar cuáles son las características propias de la programación lógica.
  - Analizar su funcionamiento y realizar un programa que utilice hechos y reglas.

  La temática del desarrollo queda a elección del grupo y se debe presentar un breve documento explicando qué conceptos de lógica aplicaron para este desarrollo.

- ✓ Generar un programa donde dada una proposición compuesta (de al menos 3 términos), se indique si se trata de una tautología, contradicción o contingencia. Las interfaces y los métodos de entrada del programa serán definidos por el grupo.

- ✓ Generar un programa de compuertas lógicas de al menos 2 proposiciones compuestas (de al menos 3 términos) donde dados los valores de verdad de las proposiciones simples, indica el valor de verdad de la proposición compuesta.

Notas

Por ejemplo, dado el siguiente conjunto de reglas que buscan una contraseña válida:

1. Si la contraseña empieza con 'a' entonces termina con 's'.
2. Si la contraseña tiene caracteres especiales entonces la contraseña es '@utnfrba'.
3. Si la contraseña termina con 's' o termina con 'v' entonces tiene longitud 4.
4. Si la contraseña tiene longitud 4 entonces la contraseña es 'alas'.

Si se tiene como hecho conocido que la contraseña empieza con 'a', se puede deducir con la regla 1 que termina con 's' y a partir de la regla 3 que tiene longitud de 4 caracteres, por lo tanto, la regla 4 nos indica que la contraseña es 'alas'. Nótese que la regla 2 no la utilizamos (regla 'falsa'). Este procedimiento se puede formalizar y resolver por deducción natural.

Sección 2: Relaciones

Presentación

En este apartado se proponen distintas aplicaciones orientadas particularmente a las relaciones de equivalencia, de orden y aplicaciones de relaciones de congruencia. El objetivo final de esta sección es ampliar sobre tópicos de la ingeniería en sistemas actuales y avanzados, usando métodos matemáticos de base aprendidos en la materia.

Tareas

- ✓ Encriptar un mensaje redactado por ustedes, utilizando un sistema de cifrado basado en corrimientos de caracteres (Encriptador del César). Realizar los cálculos correspondientes para encriptar o desencritpar los mensajes e indicar ventajas y desventajas del mismo → **1 punto**.

- ✓ Encriptar un mensaje redactado por ustedes, utilizando un sistema de cifrado RSA, basado en congruencias. Se debe armar el esquema desde el comienzo, armando la clave pública y la privada; los pasos para encriptar y los pasos para desencritpar los mensajes → **2 puntos**.

- ✓ Generar una relación de equivalencia que dado un conjunto de palabras de 5 letras (contraseñas, por ejemplo), puedan dividirse en clases seguras, medias y débiles. La relación elegida puede manejarse con las posiciones de las palabras, longitud, etcétera. Presentar un caso práctico donde la relación genere las particiones deseadas → **1 punto**.

- ✓ Investigar un método de ordenamiento algorítmico (Bubble Sort, Insertion Sort, Merge Sort, Heap Sort, Shell Sort, Quick Sort) indicando funcionamiento en términos propios y con ejemplos de funcionamiento. Asociarlo a la teoría de relaciones de orden → **2 puntos.**

Tips: encontrarán un instructivo simplificado del **algoritmo RSA** en el 'Anexo' del punto III del TP del año 2012 en el campus virtual.

Para Programadores

- ✓ Programar un encriptador de palabras mediante el método RSA, o dos métodos de encriptación simples (en caso de no querer usar el RSA); donde dado un programa, se muestre al mismo encriptado, con la opción de recuperar el mensaje original.

- ✓ Implementar un método de ordenamiento algorítmico con código propio, donde dado un conjunto de valores o vector, devuelva el conjunto ordenado, indicando cantidad de pasos que debieron realizarse para hacerlo. Incluir en el documento una breve introducción a la complejidad algorítmica y su relación con el algoritmo usado.

- ✓ Realizar un clasificador basado en relaciones de equivalencia, donde dado un conjunto de letras, números o palabras, realice las particiones correspondientes. Puede simular, por ejemplo, un sistema de archivos donde los elementos dados se incluyen en 'carpetas' (clases) según la relación armada.

Sección 3: Grafos

Presentación

En esta sección se intenta destacar la importancia de los grafos como herramientas capaces de modelar situaciones complejas en representaciones simplificadas sobre los que pueden realizarse diversos análisis. Se introducen además diversos algoritmos operados sobre grafos.

Tareas

- ✓ Tomar un mapa de ciudades, estaciones o puntos específicos y remarcar sobre el mismo: ríos, caminos, rutas o vías que las unan. Asignar a cada uno de estos tramos, un valor único de referencia. Luego, seleccionar un punto origen y un punto destino y aplicar un algoritmo de camino más corto **sin heurísticas** para hallar formalmente el mejor camino entre ambos lugares → **1 punto**.

- ✓ Igual al punto anterior, con la única diferencia de que se deberá definir una heurística para poder aplicar el algoritmo A*. Definir la lista de nodos abiertos y cerrados en cada momento para indicar el funcionamiento del algoritmo → **2 puntos**.

- ✓ Igual al punto anterior, definir la cantidad máxima de transportes, personas, barcos y otros, que pueden pasar por esos caminos y tomando dos ciudades, aplicar un algoritmo de flujo máximo para calcular numéricamente este valor para todo el tramo planteado → **2 puntos**.

- ✓ Sabiendo que un 'HOP' es un salto de un router a otro y que el 'TTL' (time to live) es la cantidad máxima de saltos que puede dar un mensaje antes de ser eliminado; definir cuantas conexiones (vistas como si fueran cables) debería haber como mínimo para que una red de 6 routers reciba todos los mensajes enviados por otro router, si el TTL vale 3 → **1 punto**.

Tips: encontrarán ejemplos de flujo máximo y costes de camino en el TP de 2013. Encontrarán un apunte acerca de **HOPS y TTL** en el Anexo IV del TP de 2012.

Para Programadores

- ✓ Realizar un algoritmo de 'camino más corto' (puede ser con código ya creado por otra persona) que resuelva problema de este tipo. La modelización puede ser de nodos que contengan caminos que salen de él y los costos de cada camino. Si se quiere agregar heurísticas, mejor**.**
- ✓ Realizar una implementación de una lista enlazada de registros y un árbol de búsqueda, se ocupe de mantener un conjunto de registros ordenada. Elaborar un informe breve sobre las condiciones particulares de cada estructura en cuanto a complejidad en la implementación, utilización de recursos y eficiencia en las búsquedas**.**

Sección 4: Lenguajes

Presentación

En este apartado se busca agilizar el uso de los conceptos presentados en la materia respecto a lenguajes, gramáticas y autómatas con un fin mayormente práctico.

Tareas

- ✓ Generar una gramática que cree contraseñas válidas a partir de ciertas reglas de seguridad definidas en sus producciones. Utilizar criterios como los de contener una mayúscula, longitud dada y elementos alfanuméricos → **1 punto**.

- ✓ Generar un autómata que sea capaz de reconocer contraseñas válidas a partir de una serie de reglas definidas previamente. El autómata debe ser no determinístico → **1 punto**.

- ✓ Investigar el funcionamiento de la máquina de Turing, dar ejemplos de usos y conclusiones respecto a los temas vistos en la materia → **2 puntos**.

- ✓ Analizar el funcionamiento de un compilador de código y asociarlo con los temas vistos en la materia → **1 punto**.

- ✓ Estudiar y aplicar en un ejemplo el método de transformación de un autómata finito no determinístico en un autómata finito determinístico → **2 punto**.

Para Programadores

- ✓ Programar un autómata reconocedor de contraseñas, indicando los estados por lo que pasó y si la contraseña indicada es válida o no. Documentar la estructura de la contraseña usada mediante expresión regular o gramática.

- ✓ Programar un analizador sintáctico para reconocer un análisis de caso simple y completo según las reglas de C++.