Técnicas Algorítmicas

Prof. Agustín Gravano

agravano@utdt.edu

Resumen y Objetivos del Curso

Este curso introduce un conjunto de técnicas para el diseño de algoritmos. Al finalizar, se espera que el alumno cuente con herramientas para la resolución de problemas de cierta dificultad (problemas clásicos como ocho reinas, torres de Hanoi, mergesort, etc.). En particular, se cubrirán los siguientes temas: recursión algorítmica, estimación de la complejidad temporal de programas recursivos, técnica de dividir y conquistar, técnica de backtracking, programación dinámica y heurísticas de programación.

Métodos y Materiales

Se emplearán una variedad de métodos de enseñanza y materiales. Las clases consistirán de difusión teórica y resolución de problemas. Habrá guías de problemas para ejercitar los contenidos de cada clase, que serán de realización obligatoria. La bibliografía del curso será en inglés, por lo que se requiere al menos de un nivel intermedio de lectura y comprensión en ese idioma. Se utilizarán también términos técnicos propios de la jerga de programación que típicamente se expresan en inglés. Se requieren conocimientos previos básicos de programación y experiencia con el lenguaje Python.

Evaluación

La nota estará basada en el siguiente esquema de ponderación:

- Participación en clase, asistencia y puntualidad 10%
- Tres evaluaciones individuales 45%
- Un trabajo práctico grupal 45%

Participación en clase

La participación en clase es una parte importante del curso, por lo que la misma será evaluada en forma continua. La evaluación de la participación estará basada en cuán al día están los

alumnos con los temas de la materia, en particular respecto de la ejercitación obligatoria. Dado que los contenidos de esta materia se acumulan clase a clase, es fundamental mantenerse al día para poder participar interactivamente del dictado de clases, y así optimizar la transmisión de conocimientos.

Evaluaciones individuales

Habrá tres evaluaciones individuales, al comienzo de las clases 2, 3 y 4. En cada una de estas evaluaciones, el alumno deberá resolver un ejercicio práctico de los temas vistos en la clase anterior, de un nivel de dificultad similar a las guías de problemas. Es requisito para aprobar el curso la aprobación de al menos dos de las tres evaluaciones individuales.

Trabajo práctico (TP)

Al final del curso habrá un trabajo práctico, a realizarse en grupos de dos o tres personas (a determinar) por fuera del horario de cursada. Para el TP no se admite colaboración entre miembros de distintos grupos. Es requisito para aprobar el curso la aprobación del TP.

Bibliografía

- T.H. Cormen y otros, "Introduction to Algorithms", tercera edición, MIT Press, 2009.
- “The Python Tutorial”, https://docs.python.org/3/tutorial/.

Contenidos del Curso

Sesión #1: Recursión algorítmica

- Definición de funciones recursivas: casos base y recursivo.
- Ejecución de algoritmos recursivos: espacios de variables, consumo de memoria. Comparación con algoritmos iterativos.
- Estimación de la complejidad temporal de algoritmos recursivos.
- Ejemplos: factorial, sumatoria, máximo de una lista.

Sesión #2: Dividir y conquistar

- Resolución de problemas mediante algoritmos recursivos con tres fases:
  1) división del problema en subproblemas de menor complejidad (*divide*);
  2) resolución recursiva de los subproblemas (*conquer*);

3) fusión de las soluciones parciales para construir la solución al problema original (*merge*).

- Ejemplos: máximo de una lista, mergesort, Hanoi.

Sesión #3: Programación dinámica

- Presentación de la programación dinámica como técnica para reducir la complejidad temporal de algoritmos, gracias al uso de subproblemas superpuestos y subestructuras óptimas.
- Ejemplos: sucesión de Fibonacci, distancia de Levenshtein.

Sesión #4: Backtracking

- Introducción a los problemas de satisfacción de restricciones.
- Backtracking como técnica algorítmica para recorrer por “fuerza bruta” el espacio de búsqueda, así alcanzar las soluciones de un problema.
- Ejemplos: viajante de comercio, 8 reinas, permutaciones de una lista.

Sesión #5: Heurísticas de programación

- Presentación de técnicas para encontrar de manera eficiente soluciones aproximadas a problemas de satisfacción de restricciones.
- Heurísticas greedy, hill-climbing y GRASP.
- Ejemplos: viajante de comercio, problema de la mochila.