Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignaturas: Diseño de Algoritmos - Algoritmos II
Primer Cuatrimestre de 2017

Guía Práctica No. 5: Programación Dinámica y Memoization

Ej. 1. Lea el capítulo 8 de *Introduction to the Design and Analysis of Algorithms* (Levitin 2003). Qué tienen en común y en qué se diferencian las técnicas *Divide & Conquer/Decrease & Conquer* y *Programación Dinámica*?

Ej. 2. Considere el problema de calcular el número combinatorio $\binom{n}{m}$. El algoritmo recursivo natural para el cálculo de $\binom{n}{m}$ sufre de algunos problemas de eficiencia. Intente mejorar este algoritmo utilizando *Memoization*. Compare empiricamente este algoritmo con la versión recursiva original.

Ej. 3. Implemente en Java el algoritmo basado en *Programación Dinámica* para calcular el n-ésimo número de Fibonacci.

Ej. 4. Considere el problema de determinar el orden óptimo en el cual conviene multiplicar una secuencia de n matrices. A partir de una solución recursiva para este problema, diseñe un algoritmo para resolver este problema utilizando la técnica de *Programación Dinámica*. Implemente este algoritmo en Java, y compare empiricamente su programa con la solución original.

Ej. 5 Dado un problema P , y una solución recursiva S (correcta) para el mismo. Siempre se puede llevar S a una solución basada en Programación Dinámica? Justifique su respuesta.

Ej. 6. Averigüe en qué consiste el problema de la mochila (*knapsack*), planteado en el capítulo 8 de *Introduction to the Design and Analysis of Algorithms*. Dado que puede existir en general más de una solución optimal para este problema, diseñe un algoritmo que determine si existe una única solución óptima para el problema.

Ej. 7. Implemente en Java un programa que calcule la distancia de Damerau-Levenshtein entre dos cadenas de caracteres. En caso de que su solución sufra de problemas de ineficiencia, intente salvarlos diseñando e implementando un algoritmo basado en *Memoization* para resolver el problema.