Manuel Arturo Chinchilla Sánchez - 2013009344

Resumen #13

Chapter 8: Dynamic Programming.

Dynamic Programming.

Nos da una manera de diseñar algoritmos personalizados que buscan sistemáticamente todas las posibilidades, mientras que almacena los resultados para evitar recalcular (proporcionando así la eficiencia). Al almacenar las consecuencias de todas las decisiones posibles y utilizar esta información de una manera sistemática, se minimiza la cantidad total de trabajo.

Caching vs. Computation.

Recalcular repetidamente una cantidad dada es inofensivo, a menos que el tiempo dedicado al hacerlo se convierte en un peso para el rendimiento. Entonces es mejor almacenar los resultados del cálculo inicial y mirarlos luego en vez de recalcular de nuevo. Un ejemplo de esto es la Sucesión de Fibonacci.

Longest Increasing Sequence.

Hay tres pasos necesarios para resolver un problema de programación dinámica:

1. Formular la respuesta como una relación de recurrencia o algoritmo recursivo.
2. Demostrar que el número de diferentes valores de los parámetros asumidos por su recurrencia están limitados por una (esperemos pequeña) polinómica.
3. Especificar un orden de evaluación para la recurrencia por lo que los resultados parciales que necesita siempre están disponibles cuando los necesite.

The Partition Problem.

Este problema se presenta cuando necesitamos dividir el trabajo entre varios trabajadores, pero las divisiones de las tareas no son del mismo tamaño.

Problema: División de un entero sin reordenamiento

Entrada: Un arreglo S de números no negativos $\{s_1,..., s_n\}$ y un entero k.

Salida: División de S en k o menos rangos, para reducir al mínimo la suma máxima sobre todos los rangos, sin reordenar cualquiera de los números.

Esto es llamado problema de División Lineal y surge a menudo en procesos paralelos. Y lo que buscamos un equilibrio entre el trabajo realizado a través de los procesadores para reducir al mínimo el tiempo de ejecución total transcurrido.

Limitations of Dynamic Programming: TSP.

La programación dinámica no siempre funciona. Es importante ver por qué puede fallar, para ayudar a evitar las trampas que conducen a algoritmos incorrectos o ineficientes.

¿Cuándo es correcto usar Programación Dinámica?

Los algoritmos de Programación dinámica son sólo correctos como las relaciones de recurrencia en las que se basan. La programación Dinámica puede ser aplicada a cualquier problema en el que se observe el principio de Optimización. Esto significa que las soluciones parciales pueden ser óptimamente extendidas en relación con el estado después de la solución parcial.

¿Cuándo son eficientes los algoritmos de Programación Dinámica?

El tiempo de ejecución de cualquier algoritmo de programación dinámica es una función de dos cosas: (1) el número de soluciones parciales hay que realizar, y (2) el tiempo que se necesita para evaluar cada solución parcial. La primera cuestión, a saber el tamaño del estado de espacio suele ser la preocupación más acuciante