Manuel Arturo Chinchilla Sánchez - 2013009344

Resumen #10

Chapter 7: Backtracking Algorithms

Backtracking Algorithms

La estrategia básica es escribir programas en los que se pueda retroceder a los puntos de decisión anteriores, en caso de que alguna decisión conduzca a callejones sin salida. Aprovechando el poder de la repetición, sin embargo, puede evitar la codificación de los detalles del proceso de retroceso de manera explícita y desarrollar dominios de problemas.

Algunos puntos importantes son:

- Se pueden resolver la mayoría de problemas que requieren Backtracking usando el siguiente enfoque recursivo:

  Si ya estás en una solución, reportar éxito.

  for (cada posible opción en la posición actual) {

  Hacer que la elección y dar un paso a lo largo de la ruta.

  Utilice la recursividad para resolver el problema de la nueva posición.

  Si la llamada recursiva tiene éxito, reportar el éxito al siguiente nivel superior.

  Volver a la toma de decisión para restaurar el estado al inicio del bucle.

  }

  Reporte de fracaso.

- El historial completo de llamadas recursivas en un problema de Backtracking incluso para aplicaciones relativamente simples suele ser demasiado complejo de entender en detalle. Para los problemas que implican ninguna cantidad significativa de Backtracking, es esencial para aceptar el recursivo salto de fe.
- A menudo puede encontrar una estrategia ganadora para juegos de dos jugadores mediante la adopción de un enfoque de Backtracking recursivo. Debido a que el objetivo en este tipo de juegos implica minimizar las posibilidades de ganar a tu oponente, el enfoque estratégico convencional se denomina algoritmo minimax.

- Es posible codificar el algoritmo minimax de una manera general que mantiene los detalles de cualquier juego específico separado de la aplicación de la estrategia minimax sí. Este enfoque hace que sea más fácil de adaptar un programa existente a los nuevos juegos.

Una de las ideas principales en este texto es tener la noción de abstracción, que es el proceso de separar los aspectos generales de un problema, de manera que ya no son oscurecidos por los detalles de un dominio específico. Se puede no estar terriblemente interesado en un programa que reproduzca un solo juego; después de todo, un solo juego es bastante aburrido una vez te diste cuenta. Lo que probablemente disfrutarías más es un programa que sea lo suficientemente general como para adaptarse a varios juegos.

El primer paso en la creación de tal generalización radica en reconocer que hay varios conceptos que son comunes en todos los juegos. El más importante de estos conceptos es el estado. En cualquier juego, hay una colección de datos que define exactamente lo que está sucediendo en cualquier punto en el tiempo. Para un juego como el ajedrez, el estado tendría lugar incluyendo qué piezas se encuentran actualmente en los cuadrados. Cualquiera que sea el juego, sin embargo, debería ser posible combinar todos los datos relevantes juntos en una sola estructura de registro y, a continuación se refieren a ella utilizando una sola variable. Otro concepto común es el de un movimiento. En el ajedrez, una medida podría consistir en un par que indica el inicio y finalización coordenadas de la pieza que se está moviendo, aunque este enfoque es, de hecho, complicado por la necesidad de representar varios movimientos esotéricos como enroque o la promoción de un peón. En cualquier caso, un movimiento también puede ser representado por una única estructura que incluye cualquier información que es apropiado para ese juego particular. El proceso de abstracción consiste en parte de la definición de estos conceptos como tipos generales, con nombres como stateT y moveT, que trascienden los detalles de cualquier juego específico. La estructura interna de este tipo será diferente para los diferentes juegos, pero el algoritmo de resumen se puede referir a estos conceptos en una forma genérica.