

Práctica 5: algoritmos probabilistas

Algoritmos Avanzados. Grado en Ingeniería Informática



15 de enero de 2018

JORGE ARANDA GARCÍA

JOSE VICENTE BAÑULS GARCIA

# **1.ALGORITMOS NUMÉRICOS**

Para la implementación de este algoritmo se ha utilizado proporcionado en las diapositivas, pero se han hecho diferentes modificaciones puesto que éste era simplemente una plantilla que no devolvía soluciones válidas. Se adjuntan los ficheros AlgoritmoNumerico.java que incluye el código propio del algoritmo y main.java que incluye la ejecución y el cálculo de los intervalos de confianza solicitados.

# **2.ALGORITMOS DE LAS VEGAS**

## Explicación del algoritmo de las N-Reinas de Sherwood.

Los algoritmos de Sherwood son una particularización de los algoritmos de las vegas, en los que siempre se obtiene una solución correcta, a pesar de que los a cálculos aleatorios sean desafortunados y la eficiencia disminuya.

En el caso que nos concierne, hay que encontrar una solución al problema de las N\_Reinas mediante este tipo de algoritmos. Para ello, lo que hacemos es colocar un cierto número de reinas al azar (puede ser variable en nuestro caso, es decir que podemos cambiarlo según nuestras necesidades) y tras haber colocado ese número de reinas, que el algoritmo continúe como si fuese un algoritmo de vuelta atrás. Si encuentra una solución, todo ha funcionado correctamente. Si no encuentra una solución y se produce un fallo, sea el punto de ejecución que sea el algoritmo vuelve a empezar, es decir, vuelve a empezar colocando las reinas aleatorias y se ejecuta hasta que encuentre una solución.

## Implementación en Java del algoritmo.

Se incluyen los ficheros N\_Reinas.java, que incluye el código del algoritmo, implementado de cero, y el fichero Practica5.java para su ejecución y cálculo de los valores requeridos.

## El número de nodos visitados si hay éxito, el número esperado de nodos visitados si hay fracaso y el número esperado de nodos visitados repitiendo hasta obtener un éxito para colocar de 0 a 16 reinas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número de reinas al azar** | **P** | **v** | **f** | **t** |
| 0 | 1.0 | 10053.0 | 0.0 | 10053.0 |
| 1 | 1.0 | 4722.879 | 0.0 | 4722.87890625 |
| 2 | 1.0 | 2041.53 | 0.0 | 2041.530029296875 |
| 3 | 1.0 | 973.877 | 0.0 | 973.8770141601562 |
| 4 | 1.0 | 542.909 | 0.0 | 542.9089965820312 |
| 5 | 0.997 | 316.553 | 18.754 | 316.6094403262124 |
| 6 | 0.9703333333333334 | 183.214 | 38.944 | 184.40466615237227 |
| 7 | 0.878616666666667 | 92.583 | 42.25275 | 98.42033503029656 |
| 8 | 0.6711825757575763 | 48.12 | 33.19524 | 64.38259828260914 |
| 9 | 0.4450468586487464 | 31.068 | 23.234045 | 60.03979561549961 |
| 10 | 0.30574030498204524 | 23.226 | 16.99068 | 61.80758232210125 |
| 11 | 0.20591876912207885 | 19.626 | 14.459844 | 75.38726143462523 |
| 12 | 0.16478577179972403 | 17.914 | 13.333258 | 85.49341734908577 |
| 13 | 0.14105338280725413 | 17.337 | 13.039366 | 96.74040656255397 |
| 14 | 0.11822486230171275 | 17.065 | 13.083132 | 114.64498592116719 |
| 15 | 0.12146717387891963 | 17.0 | 13.065125 | 111.49583152547747 |
| 16 | 0.11218412420105162 | 17.0 | 13.189859 | 121.3834557663536 |

## ¿Cuántas reinas hay que colocar aleatoriamente para obtener la mejor solución?

Tal y como se aprecia en la tabla anterior, la mejor opción es la de colocar 9 reinas de forma aleatoria.

# **3.ALGORITMOS DE MONTE CARLO**

## Implementación en Java del algoritmo.

Se incluyen los ficheros MatrixMultiplicity.java que contiene el código del propio algoritmo y Practica5.java (archivo compartido con problema de las N\_Reinas), para ejecutar un ejemplo en concreto.

## ¿Cuál es la probabilidad teórica de obtener la respuesta correcta si ejecuto el algoritmo 10 veces?

Si C != AB, la probabilidad de que una llamada al algoritmo devuelva true es ½. Las probabilidades se multiplican puesto que cada una de las llamadas que se realizan son independientes. Por lo tanto, la probabilidad de que n llamadas devuelvan todas ellas false es como máximo 2-k. Por tanto, para k llamadas la probabilidad de acierto es (1-2-k) Por lo tanto, para n = 10, la probabilidad de que el algoritmo devuelva la respuesta correcta es mayor que 99,9%.

## ¿Qué probabilidad de éxito se obtiene para verificar las siguientes matrices?

Las matrices planteadas tienen dos errores en la fila 3, pero el algoritmo solo evalúa por filas enteras. Por lo tanto, la probabilidad de éxito es como máximo de ½, es decir si no evalúa esa fila el algoritmo devolverá éxito y si no la evalúa será fallo. Esta probabilidad, es para una ejecución, como se ha comentado en el apartado anterior cuantas más veces se ejecute el algoritmo, la probabilidad de éxito aumenta bastante.

# **4.CUESTIONES TEÓRICAS**

## Compara los algoritmos deterministas con los algoritmos probabilistas.

A veces cuando un algoritmo se enfrenta a una decisión es preferible seguir un curso de acción aleatorio (algoritmos probabilistas), en lugar de invertir tiempo en determinar cuál de las alternativas es la mejor (algoritmos deterministas).

A un algoritmo determinista nunca se le permite que no termine, mientras que a un algoritmo probabilista se le permiten estos casos siempre que la probabilidad de que ocurran sea baja.

SI existe más de una solución para un problema, un algoritmo determinista siempre encuentra la misma solución, a no ser que se implemente para encontrar varias o todas, mientras que un algoritmo probabilista puede encontrar soluciones distintas para los mismos datos.

A un algoritmo determinista no se le permite que calcule una solución incorrecta, mientras que los algoritmos probabilistas pueden equivocarse, con una probabilidad baja.

Repitiendo la ejecución un número suficiente de veces para el mismo dato, se puede aum+-+-entar el grado de confianza en obtener la solución correcta tanto como se quiera.

## Clasifica y compara brevemente los tipos de algoritmos probabilistas

* NO GARANTIZAN RESULTADO CORRECTO:

-Algoritmos numéricos: Generan intervalos de confianza.

-Algoritmos de Monte Carlo: Dan la respuesta exacta con una elevada probabilidad.

* GARANTIZAN RESULTADO CORRECTO:

-Algoritmo de Las Vegas: Si dan respuesta, es la correcta, pero pueden no dar respuesta.

## ¿Qué es el efecto Robin Hood?

Es uno de los mecanismos que tienen los algoritmos de Sherwood para reducir o eliminar la diferencia de eficiencia para distintos datos de entrada. Consiste en “robar” tiempo a los ejemplares “ricos” para dárselo a los “pobres”.

# **5.CONCLUSIONES**

Esta práctica ha sido especialmente interesante dado que nos ha permitido ver modificaciones de algoritmos que hemos ido desarrollando a lo largo de la carrera, mejorándolos y optimizándolos, así no siempre prima encontrar la solución más acertada, Sino que en ocasiones es preferible asumir cierto riesgo para mejorar la eficiencia.