Universidad del Valle

PROYECTO FINAL

SIMULACIÓN COMPUTACIONAL

ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

INTEGRANTES:

JUAN DAVID MAZUERA - 1823118

NICOLAS GIRALDO - 1356195

DANIEL VELASQUEZ GARCIA- 1710070

PRESENTADO A:

ING. JOSHUA TRIANA MADRID

UNIVERSIDAD DEL VALLE

CALI, VALLE

2018

Tabla de Contenido

INTRODUCCIÓN

El Problema de las N-Reinas:

Antes de resolver el problema, debemos enfrentarnos, debemos conocerlo para luego planear una solución.

En el juego de ajedrez la reina amenaza a aquellas piezas que se encuentren en su misma fila, columna o diagonal. El problema de las 8 reinas (o n-reinas ya que dependen del número asignado) consiste en poner sobre un tablero de ajedrez ocho reinas sin que estas se amenacen entre ellas.

En la siguiente imagen se muestra un ejemplo sobre una posible solución para este problema:

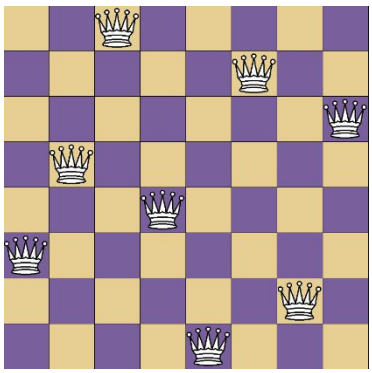

Fig.1

- Movimientos posibles de una reina en el tablero:

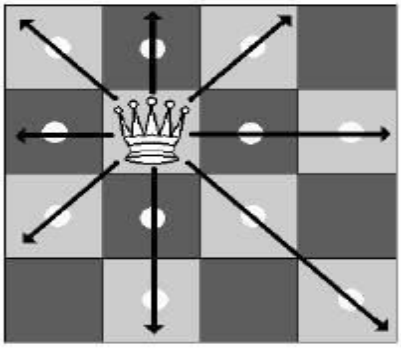

FIG.2

DESCRIPCIÓN DEL MODELO

Debe desarrollar un sistema de eventos discretos de tal forma que:

● Varias personas quieren retar al maestro, quiere decir que cuando un maestro está resolviendo un problema le toca esperar

● El orden de llegadas de las personas debe ser parametrizable pero debe tener desviación, la cronología del modelo se basará en las iteraciones dadas por el algoritmo solución.

● Al momento de calcular la orden de salida, debe hacerse con el supuesto que una iteración del algoritmo representa una unidad de tiempo (Por ejemplo, si el maestro necesito 30 iteraciones para resolver el problema, quiere decir que tardó 30 unidades de tiempo).

● El sistema debe funcionar para n cantidad de retadores y N cantidad de reinas

IMPLEMENTACIÓN

El modelo anteriormente descrito se implementó en Python 3.6.6, el modelo con un enfoque funcional consta de 3 métodos principales:

- challengeMaster
- validateQueens
- validateQueensRandom

❖ **challengeMaster**

El metodo challengeMaster ejecuta los métodos validateQueens y validateQueensRandom con un probabilidad de 50 - 50, retorna la solución dada por el método escogido aleatoriamente

❖ **validateQueens**

El método validateQueens da una solución de forma determinística para el problema de las N reinas, el método se vale de la recursión para hallar a partir de un fila N, la siguiente posición en la fila N +1, en caso de que una solución no sea encontrada la recursión vuelve a la fila N pero sumando a la columna M una unidad para generar un nueva solución, el algoritmo termina cuando las posiciones de las reinas almacenadas en la lista queens suman el número de reinas a hallar; el método usa 3 arreglos para la validación de la posición y un método de nombre validateQueen.

validateQueen evalúa la N y M donde M no este en la lista columns, donde N-M no este en la lista diag45 y donde N+M no este en la lista diag135, si alguna de estas 3 condiciones no se cumple se dice que la posición no es válida y se procede a continuar con M+1, en caso de que M+1 sea mayor al número de columnas se regresa a la fila N-1 con el fin de reevaluar la solución encontrada en la anterior fila.

❖ **validateQueensRandom**

El método validateQueensRandom da una solución apoyándose en un algoritmo las vegas, el método genera una columna M aleatoria entre [0,Q-1] donde Q es el número de reinas a hallar, si la posición N,M es validada por el método validateQueenRandom que cumple la misma función de validateQueen se procede a generar una columna aleatoriamente de nuevo para la fila N+1, en caso de que el algoritmo falle un número de veces determinado por la variable chances, se limpian las listas queens, columns, diag45 y diag135 y se inicia desde el punto inicial, el algoritmo puede fallar si la recursión excede el límite de recursividad en python.

ANÁLISIS DE LA SOLUCIÓN

La solución planteada apunta a la eficiencia de código, haciéndolo corto, reutilizable y entendible, se usó la recursión como herramienta para dar solución al problema de las N reinas con el fin de que una solución S estuviese contenida en el siguiente S +1, además que la recursión permite de manera muy sencilla descartar un camino al cual no se llega a una solución, simplemente haciendo un llamado al anterior nodo que se expresa como validateQueens(N,M+1) donde N y M son las columnas anteriores a la solución no válida S +1.

A pesar de sus ventajas, y de su facilidad de uso al momento de implementar el algoritmo las vegas de manera recursiva, encontramos limitaciones en la profundidad de la recursión, por lo que se amplió hasta 5000 con el riesgo de generar un error en la evaluación de la función que no es muy frecuente.

Aun así el método determinístico basado en la recursividad fue un modelo clave para la implementación del modelo las vegas, donde únicamente se dejó a la aleatoriedad la elección de la columna y se descartó la exploración de todas las posibilidades para una columna N, pasando a priori a generar una nueva columna para el nodo que se está evaluando del problema.

La implementación de el problema de las N reinas se dio un plazo muy breve dando más dificultad generar un modelo que pudiera simular las llegadas, salidas, tiempo de atención y de espera de los contrincantes del maestro, se evaluaron posibilidades como hilos que frecuentemente evaluaran el número de iteraciones de las recursiones y a su vez basado en el número de recursiones se iban generando las llegadas de los contrincantes; se llegó a la conclusión de que el modelo era muy complejo para ser implementado y no era muy preciso.

Cambiando el enfoque de la solución se encontró un camino más simple en donde las iteraciones eran la unidad definitiva de tiempo y a partir de un ciclo donde el número de iteraciones son sus oponentes se generan los datos estadísticos de cómo rinden los algoritmos para la solución del problema de las N reinas

El modelo planteado describe que la persona atendida en una iteración es P - 1 pero la persona la cual se despacha es P, los tiempos de las personas se calculan a través de sumas, a continuación describimos la manera en que estos se calculan

Se definen las siguientes variables

- Tiempo de llegada - person_arrival_time
- Tiempo de atención - person_attended_time
- Tiempo de espera - person_waiting_time
- Tiempo de juego - person_game_time

- Tiempo de partida - person_leave_time
- Tiempo de ocio - idle_time

El Tiempo de llegada se calcula generando aleatoriamente un número entre 0 a Interval Time (Variable que define el tiempo entre llegada de personas) y sumandolo al Tiempo de llegada de la persona anterior

El Tiempo de atención se calcula evaluando el tiempo de partida de la persona anterior, si tiempo de partida de la persona anterior es menor al tiempo de llegada de la persona actual entonces el tiempo de atención es el mismo tiempo de llegada, pero si el tiempo de llegada es menor que el tiempo de partida de la persona anterior, el tiempo de atención de la persona actual es el tiempo de partida de la persona anterior.

El Tiempo de espera se calcula restando el tiempo de atención con el tiempo de llegada.

El Tiempo de juego se calcula sumando el número de iteraciones tomada por uno de los algoritmos para dar solución al problema de las N reinas

El Tiempo de partida se calcula sumando el tiempo de atención con el tiempo de juego

El Tiempo de ocio se calcula restando el tiempo de atención de la persona actual menos el tiempo de atención de la persona anterior y su tiempo de juego, esta variable se guarda con el fin de mirar si el maestro pasa mucho tiempo sin recibir algún retador.

Las variables anteriormente definidas buscan poder mostrar información con el fin de dar respuestas y conclusiones a las preguntas planteadas en el siguiente enunciado.

HIPÓTESIS

***Tesis**: El algoritmo determinístico es más rápido, debido a que su tiempo permanecerá estable si la unidad de tiempo es número de iteraciones, al contrario del algoritmo vegas que puede tardar mucho y generar muchas iteraciones cuando los números aleatorios no son generosos con la solución buscada*

Experimento

Con el fin de probar la tesis anteriormente mencionada, se harán una serie de pruebas donde se va evaluar el caso si el maestro resolviera todo de manera determinista, si el maestro resuelve con algoritmo las vegas o si el maestro usa ambos con la misma probabilidad, la variable que determinará que tan efectivo son las diferentes maneras de abordar el problema es el tiempo de espera promedio.

Elaborando 5 ejecuciones solo usando el método determinista para 50 retadores con un intervalo de llegada de 700 iteraciones con una desviación de 87 nos arrojó los siguientes tiempos de espera

Prueba Numero	Tiempo Espera Promedio
1	6433 Iteraciones
2	6290 Iteraciones
3	6659 Iteraciones
4	6149 Iteraciones
5	6710 Iteraciones

Ahora elaboramos la misma prueba para el algoritmo las vegas

Prueba Numero	Tiempo Espera Promedio
1	0 Iteraciones
2	0 Iteraciones
3	0 Iteraciones
4	0 Iteraciones
5	0 Iteraciones

Los resultados para el algoritmo las vegas resultaron no dar mucha información acerca de la espera con estas variables, por lo que se decide reducir el tiempo de intervalo de llegada 200 con una desviación de 25 y evaluar de nuevo

Prueba Numero	Tiempo Espera Promedio
1	3252 Iteraciones
2	3884 Iteraciones
3	4003 Iteraciones
4	3212 Iteraciones
5	5595 Iteraciones

Dando un tiempo de llegada muchísimo menor podemos percibir que el algoritmo las vegas es mucho más rápido que el algoritmo determinista convencional, el algoritmo convencional tardó 49100 en derrotar a los retadores mientras que las vegas tardó la suma de 16501 iteraciones en derrotarlos.

Como último escenario se probará ambos algoritmos con una probabilidad del 50% de ser usados.

Prueba Numero	Tiempo Espera Promedio
1	773 Iteraciones
2	348 Iteraciones
3	40 Iteraciones
4	21 Iteraciones
5	43 Iteraciones

Como podemos evidenciar aunque los tiempos reducen evidentemente sobre el uso del algoritmo convencional, el uso de únicamente de el algoritmo las vegas es el claro vencedor en este experimento por lo que la tesis planteada es completamente desechada.

CONCLUSIONES

A continuación listamos las conclusiones obtenidas a lo largo de la implementación, análisis y experimentación de los algoritmos en el problema de las N reinas.

- El algoritmo las vegas es el más rápido de los tres métodos evaluados
- El algoritmo determinista es el más lento de los métodos evaluados
- El uso de ambos métodos puede hacer el sistema más estable pero lleva a una pérdida de rendimiento
- Al disminuir el intervalo de llegada de las personas, el tiempo de espera incrementa considerablemente, incrementa de forma más acelerada al usar el método determinista
- Al definir un intervalo de llegada de 800 iteraciones con una desviación de 100 obtuvimos un resultado muy bueno, donde el tiempo de espera es 11 iteraciones y el tiempo de ocio promedio es muy bajo con solo 259 iteraciones.
- Python no es el mejor lenguaje para implementar funciones recursivas