UNIVERSIDAD DEL BÍO-BÍO

"TAREA 2"

Asignatura: Algoritmo y estructura de datos avanzadas
Estudio: Magister en ciencias de la computación.
Docente: Gilberto Gutiérrez
Alumno: Fredy Moncada

Tabla de contenido

Introducción

En la actualidad es de vital importancia tener software que resuelva nuestros problemas de manera rápida y precisa, ya que disponemos de tecnologías cada vez mas rápidas y potentes, por lo tanto, en este trabajo se presentaran 2 problemas y sus respectivas soluciones mediante 2 tipos de algoritmos (determinísticos y probabilísticos), generando así una comparativa que ayudara a comprender cual de los 2 se comporta de mejor manera al solucionar el problema.

Descripción de los algoritmos

Problema 1: Validación de la multiplicación de matrices

Pseudocódigo del algoritmo determinístico para resolver el problema de la validación de la multiplicación de matrices.

```
procedure matmultiply(X, Y, Z, n);
comment multiplies n × n matrices X := YZ
  for i := 1 to n do
    for j := 1 to n do
      X[i, j] := 0;
      for k := 1 to n do
        X[i, j] := X[i, j] + Y[i, k] * Z[k, j];
```

Pseudocódigo del algoritmo probabilístico para resolver el problema de la validación de la multiplicación de matrices.

Algorithm 9 testProdMat(A, B, C, n)

```
Input: A, B, C, matrices de n×n.
Output: respuesta correcta con una probabilidad = 1/2.
for ( j=0; j<n; j++ ) do
   X_j = uniforme(0..1); /* uniforme(0..1) devuelve 0 o 1 */
end for
if ( (XA)B == XC ) then
   return true;
else
   return false;
end if
```

Algorithm 10 RepeatTestProdMat(A, B, C, n, k)

```
Input: A, B, C, matrices de n×n; k, número de iteraciones
Output: respuesta correcta con una probabilidad > 1/2.
for ( i=0; i<k; i++ ) do
   if ( testProdMat(A,B,C,n) == false ) then
      return false;
   end if
end for
return true;
```

Algorithm 11 testProdMatEpsilon(A, B, C, n, ε)

```
Input: A, B, C, matrices de n×n; ε, probabilidad máxima de error.
Output: respuesta correcta con una probabilidad > 1/2.
k = ⌈log 1/ε⌉
return RepeatTestProdMat(A,B,C,n,k);
```

Problema 2: Solución al problema de las N-reinas

Pseudocódigo del algoritmo determinístico para resolver el problema de las N-reinas.

```
reinas(k, col, diag45, diag135)
{sol[1 ... k] es k-prometedor
col = {sol[i]|1 ≤ i ≤ k}
diag45 = {sol[i] − i + 1|1 ≤ i ≤ k} y
diag135 = {sol[i] + i − 1|1 ≤ i ≤ k}}
if  k = 8  then
   Escribir sol {Un vector 8-prometedor es una solución}
else
   for  j ← 1 to 8  do
      if  j ∉ col y j − k ∉ diag45 y j + k ∉ diag135 then
         sol[k + 1] ← j {sol[1 ... k + 1] es (k + 1)-prometedor}
         reinas(k + 1, col ∪ j, diag45 ∪ {j − k}, diag135 ∪ {j + k})
      end if
   end for
end if
```

Pseudocódigo del algoritmo probabilístico para resolver el problema de las N-reinas.

```
ReinasP(N)
Col ← Diag45 ← Diag135 ← ∅
fila = 1
repeat
  libres ← ∅
  for columna 1 hasta N do
    if columna ∉ Col) and fila − columna ∉ Diag45 and fila + columna ∉ Diag135 then
      libres ← libres ∪ {columna}
    end if
  end for
  if libres ≠ ∅ then
    columna ← ElementoAleatorio(libres)
    Solucion[fila] ← columna
    Col ← Col ∪ {columna}
    Diag45 ← Diag45 ∪ {fila−columna}
    Diag135 ← Diag135 ∪ {fila + columna}
    fila ← fila + 1
  end if
until fila > N or libres = ∅
if libres = ∅ then
  return "SolucionVacia"
else
  return Solucion
end if
```

Resultados experimentales

<u>Problema 1</u>: Para este problema era necesario comprobar si la multiplicación de las matrices A*B era igual o diferente de la matriz C, aplicando 2 algoritmos que lo calculan de manera diferente (probabilístico y determinista)

En esta tabla se puede visualizar los tiempos de ejecución (milisegundos) al entregar los resultados A*B = C.

N	determinista	1/2	1/4	1/8	1/16	1/32	1/64	1/128
10	0	1	1	1	1	1	1	1
25	2	3	4	5	5	7	4	4
50	12	1	2	2	3	4	6	5
75	33	3	8	9	12	15	33	25
100	8	9	18	26	35	38	15	15
200	66	27	49	65	60	100	117	140

En esta tabla se puede visualizar los tiempos de ejecución (milisegundos) al entregar los resultados A*B != C.

N	determinista	1/2	1/4	1/8	1/16	1/32	1/64	1/128
10	0	0	0	1	0	0	0	1
25	1	0	1	0	1	1	0	0
50	12	1	1	2	1	2	1	1
75	4	5	8	3	6	4	4	19
100	8	4	3	11	5	4	2	11
200	81	47	108	126	46	129	13	66

Como se puede observar en las tablas, el algoritmo probabilístico tiende a comportarse de peor manera cuando el error es ínfimo, pero si el error tiende a 50%, este se comporta de mejor manera que el determinista.
Por otra parte, el determinista se comporta de manera regular, ya que a medida que los valores de las matrices crecen este se comporta de peor manera, siendo así necesario aplicar otros algoritmos, como por ejemplo el probabilístico, con un error relativamente seguro.

***Problema 2*:** Para este problema, es necesario resolver el problema de las N-reinas, utilizando 2 algoritmos vistos en clases (determinista y probabilista), en la siguiente tabla se visualizará los tiempos de ejecución (milisegundos) que tomo para encontrar la solución.

N	Determinista	Probabilística
5	0	1
10	5	2
15	9	1
20	191	1
25	38	2
30	41.643	3
35	230.978	3

Como se puede observar en la tabla, el algoritmo probabilístico es considerablemente mejor que el determinista, ya que con una tabla de 35x35 ya requiere demasiado tiempo para encontrar las soluciones. Por lo tanto, el probabilístico es muy útil en este caso, ya que su respuesta es instantánea.

Conclusión

Según los resultados obtenidos a lo largo de este informe, se puede comprender que en estas 2 problemáticas, los algoritmos probabilísticos tienden a comportarse de mejor manera respecto al determinístico, ya que en ambos casos los tiempos de ejecución son mas cortos, que no superan el segundo, en cambio los algoritmos determinísticos se encontraban muy acotados al segundo, incluso superando mas de 5 segundos en algunos casos, como se puede observar en las tablas con anterioridad.

Referencias bibliográficas

- Lecture notes on algorithm analysis and computational complexity (4 edition)
- Presentaciones entregadas por el docente Gilberto Gutiérrez

Anexo

Problema 1:

```
public class Problema1 {

    private int[][] A;
    private int[][] B;
    private int[][] C;
    private int tamanio;
    private double[] epsilon = {
            (double)1/2,
            (double)1/4,
            (double)1/8,
            (double)1/16,
            (double)1/32,
            (double)1/64,
            (double)1/128
    };

    public Problema1(int tamanio){...}

    private void loadABequalC(){...}

    private void loadABnotEqualC(){...}

    public void aplicar(){...}

    private void deterministico(){...}

    private void probabilistico(){...}


    private boolean testProdMat(){...}

    private boolean repeatTestProdMat(double k){...}

    private boolean testProdMatEpsilon(double epsilon){...}
}
```

Problema 2:

```java
public class Problema2 {

    private int[][] A;
    private int tamanio;

    public Problema2(int tamanio){...}

    public void aplicar(){...}

    private void deterministico(){...}

    private void printSolution(){...}

    private boolean isSafe(int row, int col){...}

    private boolean solveNQUtil(int col){...}


    private void probabilistico(){...}

}
```

Para una comprensión mejor de la solución, el proyecto se adjuntará con el informe, por si existe la necesidad de realizar las pruebas respectivas y comprobar los resultados obtenidos en el informe.