Laboratorio Nro. 2: Fuerza bruta

Luisa María Vásquez Gómez
Universidad Eafit
Medellín, Colombia
lmvasquezg@eafit.edu.co

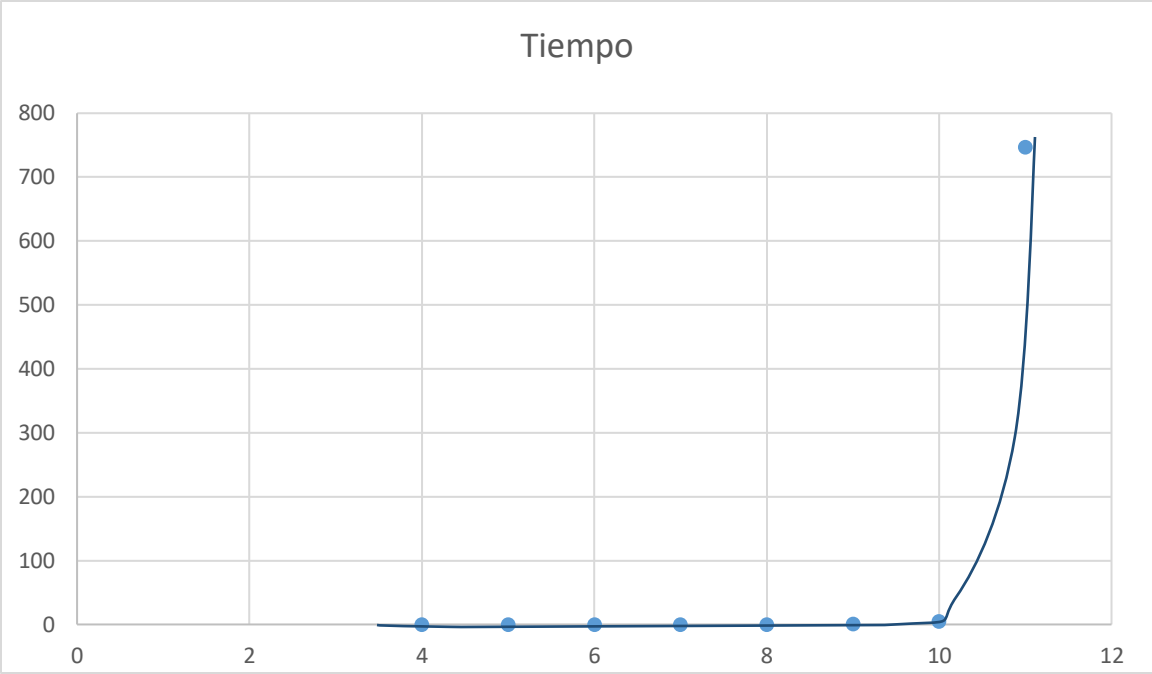
Juan José Parra Díaz
Universidad Eafit
Medellín, Colombia
jjparrad@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1) Aparte de la fuerza bruta, la cual revisa absolutamente todas las soluciones posibles a un problema, existe el backtracking, técnica cuyo objetivo es acortar los tiempos de ejecución al usar fuerza bruta. El bactracking es un tipo de fuerza bruta que descarta los caminos erróneos antes de entrar en ellos, ahorrándose así explorar dentro de los caminos sin salida.

3.2)

Valor de N	Tiempo de ejecución
4	0.000s
5	0.000s
6	0.000s
7	0.016s
8	0.062s
9	0.487s
10	5.055s
11	747.3000s
12	900+
13	900+
14	900+
15	900+
…	…
32	900+
N	**O(n!)**



3.3) El ejercicio 2.1 funciona leyendo la entrada línea por línea y, a partir de esta, guarda la información de los tableros y dónde se encuentra cada uno de los huecos que este tiene en una matriz. Se repite esto hasta que se encuentre el número “0”, que significa el fin del archivo y guarda las matrices en un arreglo. Con esa información, recurre al algoritmo de las reinas en donde se buscan todas las posibilidades, y, para verificar la validez de la posición de la reina, revisa también que ésta no se haya puesto en un hueco. Se hace esto con cada elemento de arreglo (cada tablero). Al finalizar, se imprime el número de soluciones posibles para el cada caso.

3.4) La estructura de datos usada para el problema es un arreglo de matrices. Este se eligió, porque fue la manera más eficiente de guardar los huecos en los tableros sin sobre escribir información. La matriz guarda la columna y las posiciones que contienen huecos en estas, mientras que el arreglo guarda las diferentes matrices que se encontraron en la entrada. Se usan matrices y no arreglos porque de otra manera, solamente se podría guardar un hueco por columna, y cada columna podría contener más de 1 hueco.

3.5)

```
public ArrayList<int[][]> inputManager() throws IOException{                    C*n*m²

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));   C
    String entrada = br.readLine();                                             C
    int largo = Integer.parseInt(entrada);                                      C
    ArrayList<int[][]> tableros = new ArrayList<>();                            C
    while(largo != 0){                                                          C*n
        int[][] huecos = new int[largo][largo];                                 C*n
        for(int f = 0; f < largo; f++){                                         C*n*m
            entrada = br.readLine();                                            C*n*m
            for(int c = 0; c < largo; c++){                                     C*n*m²
                if(entrada.charAt(c) == '*'){                                   C*n*m²
                    huecos[f][c] = 1;                                           C*n*m²
                }
            }
        }
        tableros.add(huecos);                                                   C*n
        entrada = br.readLine();                                                C*n
        largo = Integer.parseInt(entrada);                                      C*n
    }
    return tableros;                                                            C
}


public static void main(String [] args){                                        O(nm!)
    System.out.println();                                                       C
    nreinasTableroRoto nreinas = new nreinasTableroRoto();                      C*n*m²
    ArrayList<int[][]> huecos = null;                                           C
    try{                                                                        C
        huecos = nreinas.inputManager();                                        C
    } catch (IOException e){}                                                   C

    for(int[][] i: huecos){                                                     C*n
        int n;                                                                  C*n
        for(n = 0; n < i.length; n++){}                                         C*n*m
```

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

```
            int[] tablero = new int[n];                                          C*n
            nreinas.sol=0;                                                       C*n
            int res = nreinas.nReinas(0,n,tablero,i);                            O(m!)*n
            System.out.println("Para caso de prueba " + n + ", puedo poner en " +  C*n
res + " posiciones diferentes las reinas.");
        }
    }

    private boolean puedoPonerReina(int r, int c, int[] tablero, int[][] huecos) {  O(m)
        if(huecos[r][c] == 1){                                                   C
            return false;                                                        C
        }
        for (int i=0; i<r;i++) {                                                 C*m
            if ((i-r)==(tablero[i]-c)||(i-r)==(c-tablero[i]) || tablero[i]==c){   C*m
                return false;                                                    C*m
            }
        }
        return true;                                                             C
    }

    private int nReinas(int r, int c, int[] queens, int[][] huecos){             O(m!)
        for (int i = 0;i<c;i++){                                                 C*m
          if(puedoPonerReina(r, i, queens, huecos)){                             O(m)*m
              queens[r] = i;                                                     C*m
              if(r ==c-1){                                                       C*m
                    //imprimirTablero(queens);                                   C*m
                    sol++;                                                       C*m
              }else{                                                             C*m
                    nReinas(r+1,c,queens, huecos);                               T(m-1)*m
              }
          }
        }
        return sol;                                                              C
    }
}
```

O(nm!)

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

3.6) En el cálculo de la complejidad, la variable 'n' representa el número de tableros que serán analizados como casos apartes, es decir, los elementos que se ingresan en la entrada, y 'm' es el tamaño de cada uno de los tableros, el número de filas y columnas que tiene.

4) Simulacro de parcial

1.

a) actual > máximo
b) $O(n^2)$

2.

a) arr,k+1
b) $O(n^3)$

3.

a) i-pat.length();
b) txt.length()
c) $O(n^2)$

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co