Laboratorio Nro. 1
Recursión

Simón Marín Giraldo
Universidad Eafit
Medellín, Colombia
smaring1@eafit.edu.co

Miguel Fernando Ramos García
Universidad Eafit
Medellín, Colombia
mframosg@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 T(n) = T(n-2) + T(n-1) + c_2 = (T(n) = (C_1*(2^n-1))) = O(C_1*(2^n-1))= O(2^n)

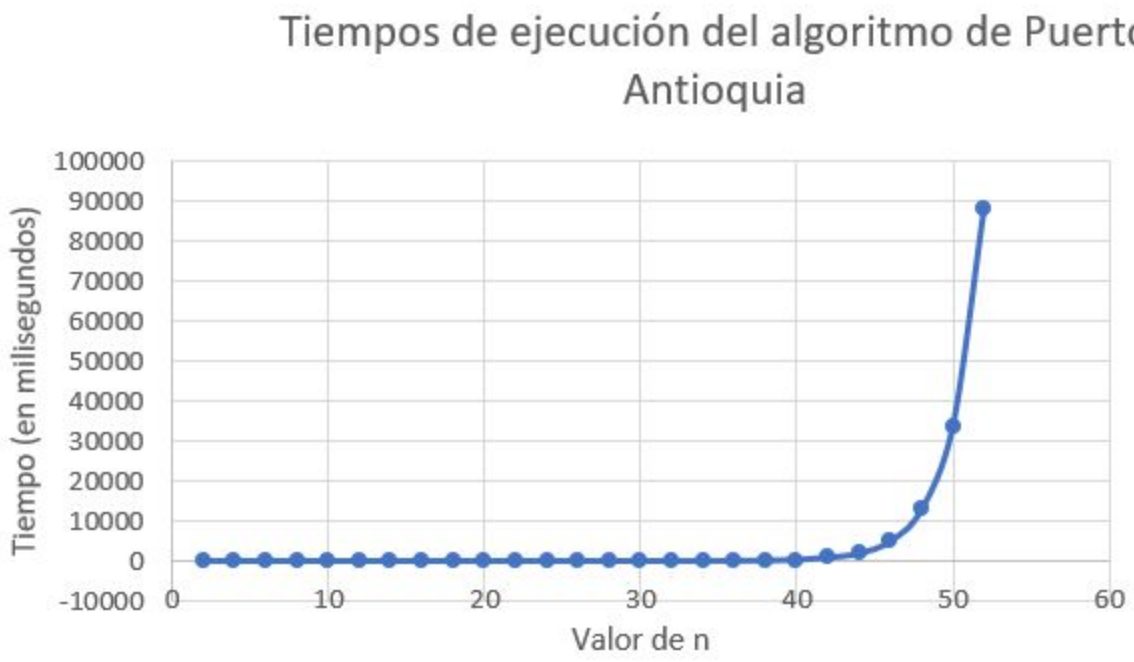
3.2 El código usado para tomar los tiempos fue:

```
public class PruebasLab {
   public static void main(String[] args) {
      urabaTest();
   }

   public static int nRectangulos(int n)
   {
      if (n <= 2) return n;   // C_1
      else return nRectangulos(n - 1) + nRectangulos(n - 2); //T(n) = T(n-1) + T(n-2)
   }

   public static void urabaTest() {
        int pruebas[] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44,
          46, 48, 50, 52};
      System.out.println(pruebas.length);
      for (int i = 0; i < pruebas.length; i++) {
         long inicio = System.currentTimeMillis();

         nRectangulos(pruebas[i]);
         long fin = System.currentTimeMillis();
         double dif = (double) (fin - inicio);
         System.out.println("La duración para n = " + pruebas[i] + " fue de " + dif + " milisegundos.");
      }
   }
}
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

La duración para n = 2 fue de 0.0 milisegundos.
La duración para n = 4 fue de 0.0 milisegundos.
La duración para n = 6 fue de 0.0 milisegundos.
La duración para n = 8 fue de 0.0 milisegundos.
La duración para n = 10 fue de 0.0 milisegundos.
La duración para n = 12 fue de 0.0 milisegundos.
La duración para n = 14 fue de 0.0 milisegundos.
La duración para n = 16 fue de 0.0 milisegundos.
La duración para n = 18 fue de 1.0 milisegundos.
La duración para n = 20 fue de 0.0 milisegundos.
La duración para n = 22 fue de 0.0 milisegundos.
La duración para n = 24 fue de 0.0 milisegundos.
La duración para n = 26 fue de 1.0 milisegundos.
La duración para n = 28 fue de 1.0 milisegundos.
La duración para n = 30 fue de 2.0 milisegundos.
La duración para n = 32 fue de 6.0 milisegundos.
La duración para n = 34 fue de 16.0 milisegundos.
La duración para n = 36 fue de 43.0 milisegundos.
La duración para n = 38 fue de 110.0 milisegundos.
La duración para n = 40 fue de 294.0 milisegundos.
La duración para n = 42 fue de 829.0 milisegundos.
La duración para n = 44 fue de 1897.0 milisegundos.
La duración para n = 46 fue de 4905.0 milisegundos.
La duración para n = 48 fue de 12831.0 milisegundos.
La duración para n = 50 fue de 33576.0 milisegundos.
La duración para n = 52 fue de 88013.0 milisegundos.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473



3.3 O(2^n): exponencial. Se trata de funciones que duplican su complejidad con cada elemento añadido al procesamiento. Por este motivo, este algoritmo no es viable para ser usado en Puerto Antioquia, dado que con solo n = 1000 se requieren 10 a la 301 procesos aproximadamente. Si solo mil millones, para un procesador normal son 10 a la 9 procesos, ahora, con 10 a la 301 es totalmente inviable.

3.4 En el método groupSum5 se verifica, verdadero falso si, dado un arreglo de enteros, se puede tomar un grupo de algunos de los enteros, comenzando por el índice o posición de inicio (start), de tal manera que, la suma de los elementos en el grupo sea igual al entero dado como objetivo (target). Con la condición adicional de que todos los múltiplos de 5 que estén en el arreglo se deben incluir en el grupo y si el valor inmediatamente después de un múltiplo de 5 es 1, este valor no debe ser incluido en el grupo. El código para este ejercicio es el siguiente:

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```java
public boolean groupSum5(int start, int[] nums, int target) {
  if (start >= nums.length) {
    if (target == 0) {
      return true;
    }
    return false;
  }
  if (nums[start] % 5 == 0) {
    if (start < nums.length - 1 && nums[start+1] == 1) {
      return groupSum5(start + 2, nums, target - nums[start]);
    }
    return groupSum5(start + 1, nums, target - nums[start]);
  }
  if (groupSum5(start + 1, nums, target - nums[start])) {
    return true;
  }
  return groupSum5(start + 1, nums, target);
}
```

3.5 factorial = T(n) = C_2 + T(n-1)
powerN = T(n) = C_2 + T(n-1)
fibonacci = T(n) = T(n-2) + T(n-1)
bunnyEars2 = T(n-1)+C_1
triangle = C_2 + T(n-1)
groupSum6 = T(n) = 2 T(n-1) + C2
groupNoAdj = T(n) = 2 T(n-1) + C2
groupSumClump = T(n)=2 T(n-1) + C2
split53=T(n) = 2 T(n-1) + C2
splitArrayHelper = T(n)=2 T(n-1) + C2

3.6 Para factorial, n representa el valor al cual encontrar el factorial.
Para powerN: n representa la potencia a la cual se eleva una base.
Para fibonacci: n representa el n-ésimo número de la sucesión de fibonacci
Para bunnyEars2: n representa la cantidad de conejos.
Para triangle: n representa las filas de la pirámide a construir
Para groupSum6: n representa el tamaño del arreglo de enteros.
Para groupSumNoAdj: n representa el tamaño del arreglo de enteros.
Para groupSumClump: n representa el tamaño del arreglo de enteros.
para split53: n representa el tamaño del arreglo de enteros.
Para splitArrayHelper: n representa el tamaño del arreglo de enteros.

4) Simulacro de Parcial

4.1 *start + 1, nums, target*

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.2 *d*
4.3 ***4.3.1:*** *n-1,a,c,b* ***4.3.2:*** *n, a* ***4.3.3:*** *b, c*
4.4 *e*
4.5 *public int formas(int n){*
if(n <= 2) n;
return formas(n-1) +
formas(n-2);
}
T(n)=T(n−1)+T(n−2)+C

4.6 ***4.6.1:*** *sumaAux(n, i+1)* ***4.6.2:*** *sumaAux(n, i+1)*
4.7 *opcional*
4.8 ***4.8.1:*** *return 0* ***4.8.2:*** *ni + nj*
4.9 *c*
4.10 *b*
4.11 *lucas(n-1)+lucas(n-2). T(n)=T(n−1)+T(n−2)+c, que es O(2^n)*
4.12 ***4.12.1:*** *sat* ***4.12.2:*** *Math.max(fi, fj)* ***4.12.3:*** *sat*

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473