Facultad Politécnica - Departamento de Informática – Carrera: Ingeniería Informática
Algoritmos y Estructura de Datos III - Primer Examen Parcial – 2do Semestre – Sección TQ
Prof. Cristian Cappo Miércoles, *12/09/2012* Puntos: 60

Nombre y Apellido: _____Nro. CIC:_____

Tema 1 *(13p)*

A continuación se presenta un código Java para calcular la Transforma Rápida de Fourier (FFT).

```
public class FFT {
  public static Complex[] fft(Complex[] x) {
     int N = x.length;
     if (N == 1) return new Complex[] { x[0] };
     if (N % 2 != 0) {
        throw new RuntimeException("N no es potencia de 2");
     }

     Complex[] par = new Complex[N/2];
     for (int k = 0; k < N/2; k++) par[k] = x[2*k];

     Complex[] q = fft(par);

     Complex[] impar  = par;
     for (int k = 0; k < N/2; k++) impar[k] = x[2*k + 1];

     Complex[] r = fft(impar);

     Complex[] y = new Complex[N];
     for (int k = 0; k < N/2; k++) {
         double kth = -2 * k * Math.PI / N;
         Complex wk = new Complex(Math.cos(kth), Math.sin(kth));
         y[k]       = q[k].plus(wk.times(r[k]));
         y[k + N/2] = q[k].minus(wk.times(r[k]));
     }
     return y;
   }
}
```

```
/* Implementacion de Números Complejos */

public class Complex {
   private final double re;
   private final double im;

   public Complex ( double r, double i){
       this.re = r ; this.im = i;
   }

   public Complex plus (Complex b) {
      return
         new Complex( this.re + b.re,
                      this.im + b.im );
   }

   public Complex minus (Complex b) {
      return
         new Complex ( this.re - b.re,
                       this.im - b.im );
   }

   public Complex times (Complex b) {
     return
      new Complex (
       this.re * b.re  - this.im * b.im,
       this.re * b.im  + this.im * b.re );
   }
}
```

Determinar para el método fft() lo siguiente:

a) La *T(n)* ***(5p)***

b) El costo asintótico temporal en términos de Θ ***(4p)***

c) El costo asintótico espacial en términos de Θ ***(4p)***

Tema 2 *(6p)*

Suponga que usted ha colectado los siguientes datos de tiempos de ejecución de un programa como una función de la entrada de tamaño N.

N	Tiempo en segundos
125	0,03
1000	1,00
8000	32,00
64000	1034,00
512000	32768,00

Estime el tiempo de ejecución de este programa (en segundos) como una función de N (*T(N)*).

Tema 3 *(7p)*

Por cada una de las funciones mostradas a la izquierda, indicar la letra del mejor orden de crecimiento de su tiempo de ejecución. Puede utilizar una respuesta más de una vez o no usarla. Las posibilidades son:

$A)\ \log N \quad B)\ N \quad C)\ N \log N \quad D)\ N^2 \quad E)\ N^3 \quad F)\ 2^N \quad G)\ 3^N \quad H)\ N!$

__
```
public static int f1 ( int N ) {
      int x = 0;
      for ( int i=0; i < N; i++ ) x++;
      return x;
  }
```

__
```
public static int f2(int N) {
      int x = 0;
      for (int i = 0; i < N; i++)
        for (int j = 0; j < i; j++)
            x += f1(j);
      return x;
}
```

__
```
public static int f5(int N) {
      int x = 0;
      for (int i = N; i > 0; i = i/2)
          x += f1(i);
      return x;
}
```

__
```
public static int f3(int N) {
      if (N == 0) return 1;
      int x = 0;
      for (int i = 0; i < N; i++)
           x += f3(N-1);
      return x;
  }
```

__
```
public static int f4(int N) {
      if (N == 0) return 0;
      return f4(N/2) + f1(N) + f1(N) + f1(N) +
      f4(N/2);
  }
```

__
```
public static int f6(int N) {
      if (N == 0) return 1;
      return f6(N-1) + f6(N-1);
  }
```

__
```
public static int f7(int N) {
      if (N == 1) return 0;
      return 1 + f7(N/2);
  }
```

Tema 4 ***(14p)***

Suponga que en el API de Java, la clase `java.util.LinkedList` es implementada usando una lista doblemente enlazada, manteniendo una referencia al primer y último nodo en la lista, junto con el tamaño de la lista. El diseño de la clase se muestra abajo:

```
public class LinkedList<Item> {
    private Node first;    // el primer nodo en la lista
    private Node last;     // el último nodo en la lista
    private int N;         // cantidad de elementos
    private class Node {
       private Item item;       //  el dato
       private Node next, prev; // el nodo siguiente y anterior
    }
    ...
}
```

a) Considerando que un objeto tiene una sobrecarga de 16 bytes, que el uso de la memoria es rellenada de forma que sea múltiplo de 8 bytes (en una máquina de 64 bits) y que una referencia ocupa 8 bytes, indique la cantidad de memoria necesaria para (no se incluye la memoria ocupada por los datos en sí, solo las referencias a ellos): ***(4p)***

a.1) Para el objeto ***`Nodo`***.

a.2) Para el objeto ***`LinkedList`*** para N nodos.

b) ¿Cuál es el orden de crecimiento para el peor caso de cada una de las operaciones que se muestran abajo, usando las siguientes posibilidades ? $1 \quad \log N \quad \sqrt{N} \quad N \quad N \log N \quad N^2$

`agregarPrimero(item)`	Agrega el dato *item* al comienzo de la lista	
`obtener(i)`	Retorna el dato en la posición *i* de la lista	
`reemplazar(i, item)`	Reemplaza el dato de la posición *i* con *item*	
`removerUltimo()`	Borra y retorna el elemento posicionado al final de la lista	
`contiene(item)`	Retorna *true* o *false* de acuerdo a que *item* se encuentre o no en la lista	

Tema 5 ***(12p)***

Suponga que las siguientes claves son insertadas en algún orden dentro de una tabla de dispersión de exploración lineal de tamaño 7 (sin rehashing), usando los siguientes valores de dispersión:

Clave	A	B	C	D	E	F	G
Valor hash	5	2	5	1	4	1	3

a) Dar el contenido de la tabla de dispersión considerando que son insertadas en orden alfabético ***(6p)***

Posición	0	1	2	3	4	5	6
Clave							

b) Usando el método NO perezoso, borrar todas las claves de la tabla generada en a) en este orden G, F, E, D, C, B y A. ***(4p)***

c) ¿Cuál de las siguientes tablas de dispersión son válidas si las claves son insertadas en algún otro orden que no sea alfabético? ***(2p)***

Opción	0	1	2	3	4	5	6
I	A	F	D	B	G	E	C
II	F	A	D	B	G	E	C
III	C	A	B	G	F	E	D

Marque con una **X** a la izquierda de la mejor opción:

__Solo I __Solo I y II __Solo I y III __I, II y III __Ninguna

Tema 6 ***(8p)***

Encuentre el orden de crecimiento en término de O (*O(g(n)*) de las siguientes sumas (utilice la función *g(n)* más simple y precisa posible).

a) $\sum_{i=0}^{n-1} (i^2+1)^2$ b) $\sum_{i=1}^{n} (i+1)2^{i-1}$ c) $\sum_{i=2}^{n-1} \log_2 i^2$ d) $\sum_{i=0}^{n-1} \sum_{j=0}^{i-1} (i+j)$

Tema 7 ***(5p) - OPCIONAL***

Diseñe un método estático genérico (usando *Generic*) en Java que retorne un valor lógico que indique si un elemento X se encuentra dentro de un arreglo A del mismo tipo de X. Ambos elementos: A y X se pasan como parámetros del método.

Muestre al menos dos ejemplos de uso de dicho método.