Review

Algoritmos y Estructuras de Datos - Curso 2011
2do Parcial – Sábado 25 de Junio

Ejercicio 1.

Dado el siguiente segmento de código,

a.- Calcular analíticamente el T(n), detalle los pasos seguidos para llegar al resultado.

b.- Calcular el O(n) justificando usando la definición de big-OH

```
j = 1;
while (j <= n) {
    for (i = n*n; i >=1; i = i-3)
        x=x+1;
    j = j*2;
}
```

Ejercicio 1. Solución(a):

- El loop más interno es de orden n^2
  - $i = n^2$ (paso inicial)
  - $i = n^2 - 3$ (paso 1)
  - $i = n^2 - 6$ (paso 2)
  - ...
  - $i = n^2 - 3*k$ (paso k-ésimo)
  - cuando $i = 0$ sale del for → $n^2 - 3* k = 0 \rightarrow k = n^2/3$ es la cantidad de veces que se ejecuta el for.

Valor de J	iteraciones	Nuevo valor de J
J = 1	1	J = 2
J = 2	2	J = 4
J = 4	3	J = 8
J = 8	4	J = 16
J = 16	5	J = 32
...	...	...
$2^{(k-1)}$	k	2^k

- El loop del while es $\log_2 n$
  - Cuando el while da k vueltas, sale xq no se cumple que $2^k > n$
  - $k > \log(n)$
- $T(n) = cte + (otraCte/3) * n^2 * \log_2 n$

Ejercicio 1. Solución(b):

$T(n) = cte + (otraCte/3) * n^2 * \log_2 n$

$\exists c, n_0$ con c > 0 y n_0 tales que $T(n) \leq cn^2 \log_2(n) \; \forall \; n \geq n_0$

Ejercicio 2

Había una vez un chimpancé llamado *Luchu Bandor,* cuyo significado era "Mono Playboy". *Luchu* estaba infelizmente casado con *Bunty Mona,* una chimpancé muy bonita pero de baja estatura. *Luchu* era alto y guapo, se sentía incómodo cuando estaba con *Bunty* en lugares públicos, ya que la gente los miraba a ellos continuamente. En un momento dado, *Luchu* no pudo soportar más esta situación y decidió hacer justicia a su nombre. Él comenzó a buscar una nueva esposa en el "Colegio Nacional de Señoritas Chimpancés". Cada día *Luchu* se subía a unas cañas de bambú y esperaba a que el ejercicio matutino empezara. Desde allí podía ver a todas las chimpancés haciendo su rutina de ejercicio diario. Ahora, *Luchu* estaba buscando a una chimpancé más alta pero que sea más baja que él, y también estaba interesado en aquella chimpancé un poco más alta que él. Sin embargo, alguien de su misma altura no la consideraba.

Luchu pudo modelar la situación descripta a través de un árbol AVL, el cuál contenía todas las alturas de las señoritas chimpancés que él había observado durante un cierto período de tiempo.

Su trabajo consiste en ayudar a *Luchu* para encontrar a las dos mejores chimpancés de acuerdo al criterio de selección establecido: la chimpancé más alta de las más bajas que él y la más baja entre las más altas que él.

Usted debe implementar un método en la clase árbol AVL, considerando que recibe como parámetro la altura de *Luchu* y debe devolver las alturas de las dos chimpancés buscadas ordenados de manera creciente. En el caso que sea imposible encontrar alguna de estas dos alturas devuelva un valor igual a 0 para la menor y 999 para la mayor. (0 y 999 no son alturas válidas, no están en el árbol)

Importante: considere que en el árbol existe una altura igual a la altura de *Luchu*.

Ejercicio 2. Solución:

```
ListaGenerica<int> buscar (int alt){
   ListaGenerica<int> l = new ListaEnlazadaGenerica<int>();
   this.altMayor = 999;
   this.altMenor = 0;
   this.find(alt);
   l.agregar(altMenor);
   l.agregar(altMayor);
   return l;
}
```

Ejercicio 2. Solución:

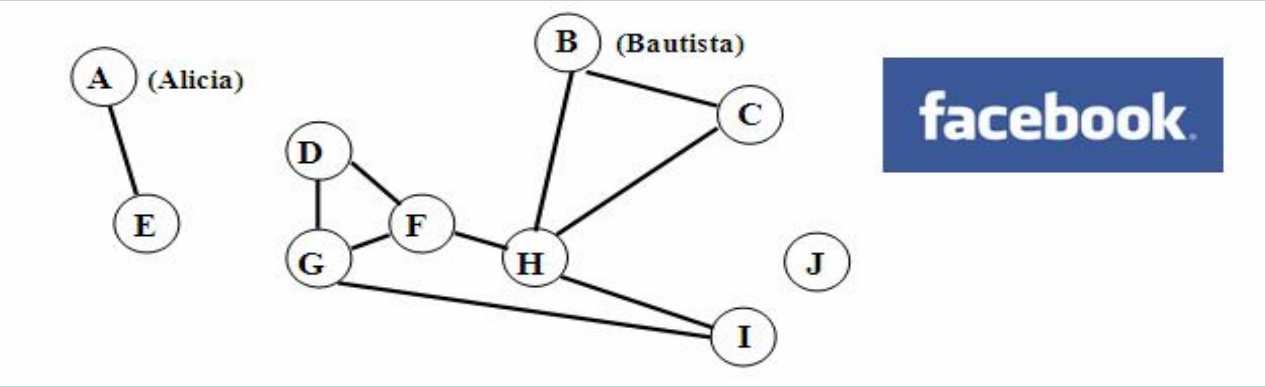
```
void  find (int altLuchu) {
  if (!this.esVacio()) {
    if (altLuchu.compareTo(this.getDatoRaiz()) == 0)  {  // encontré la altura de Luchu !
      if( ! this.getHijoIzquierdo().esVacio())
        altMenor  = this.getHijoIzquierdo().findMax();    // busco la altura mayor del sub izq
      if( ! this.getHijoDerecho().esVacio())
        altMayor = this.getHijoDerecho().findMin();  // busco la altura menor del sub der
    } else {
      if (altLuchu.compareTo(this.getDatoRaiz())  > 0) {
          altMenor  = this.getDatoRaiz();
          this.getHijoDerecho().find(altLuchu);                 // busco en la rama derecha
      } else {
          altMayor  = this.getDatoRaiz();
          this.getHijoIzquierdo().find(altLuchu);               // busco en la rama izquierda
             }
          }
                    }
}
```

Ejercicio 2. Solución(cont.):

```
int findMax() {
   int alt;
   if( ! this.getHijoDerecho().esVacio())
     alt = this.getHijoDerecho().findMax();
   else {
     alt = this.getDatoRaiz();
   }
   return alt;
}
```

Ejercicio 3.

Facebook, Friendster, etc son en la actualidad populares sitios web de redes sociales En estos sitios, una persona puede crear un perfil virtual y construir una relación de amistad con cualquier otra persona que esté registrada en la red, siempre y cuando ésta acepte su invitación. Esta relación de amistad es muy bien modelada usando un grafo, por ejemplo como el que se muestra en la figura siguiente (Por simplicidad los nombres de las personas fueron abreviados a un carácter).



Usted debe determinar cuáles son los grupos de amigos que hay en la red social Facebook

En el ejemplo :

Hay 3 grupos de amigos : {{BCDFGHI} con 7 miembros, {AE} con 2 miembros y {J} con 1 miembro}

Ejercicio 3. Solución:

Solución:

```
ListaGenerica<ListaGenerica<T>> componentes() {
   ListaGenerica<T> l = new ListaEnlazadaGenerica<T>();
   Boolean [] visitados = new Boolean[this.vertices.tamanio()]
   for(int i = 0; i < visitados.length; i++)  {visitados[i]=false;}
   vertices.comenzar();
   while (!vertices.fin()){
      Vertice act = vertices.elemento();
      if (!visitados[act.posicion()]) {
         ListaGenerica<T> comp = new ListaEnlazadaGenerica<T>();
         dfs(vertices.elemento(), comp, visitados); //genera una componente conexa
         l.agregar(comp);
      }
      vertices.proximo();
   }
   return  l
}
```

Ejercicio 3. Solución:

```
void dfs(Vertice v, ListaGenerica<T> comp, Boolean[] visit){
   comp.agregar(v);
   visit[v.posicion()]=true;
   ListaGenerica<Arista> adjs = v.obtenerAdyacentes();
   adjs.comenzar();
   while(!adjs.fin()){
      Vertice ady = adjs.elemento().verticeDestino();
      if (!visit[ady.posicion()]) {
         dfs(ady, comp, visit);
      }
   adjs.proximo();
 }
```