ESTRUCTURA DE DATOS
PARTE TEÓRICA

Instrucciones:

- 17:00-18:10: Escribir la parte teórica a mano en letra clara y legible y luego subir en un solo archivo.pdf (fotografía o escaneado) al aula virtual. Luego de las 18:20 ya no podrán entregar el archivo.
- 18:10-19:10: La parte práctica subir al aula virtual. Luego de las 19:20 ya no podrán entregar el archivo.

Ejercicio 1 *(0.8P) Ordenar las siguientes funciones:*

$$f(n) = n^2 * log(8n^3), \qquad g(n) = \sum_{i=1}^{n} i^2$$

Ejercicio 2 *(0.8 P) Se desea dividir a $2n$ recintos electorales en dos distritos de n recintos cada uno. Cada recinto tiene asociado un índice p_i que representa el número de electores, es decir, $p_i < \alpha n$. La entidad electoral sugiere crer distritos de la manera más injusta posible, es decir, construir distritos A y B con el mayor desequilibrio respecto a número de electores. Mostrar cómo se puede hacer el trabajo en tiempo $\mathcal{O}(n)$.*

Ejercicio 3 *(0.8P) Sea T un árbol binario de búsqueda con números reales. Escribir un algoritmo para los siguientes casos:*

- *Encontrar el nodo $x \in T$ tal que $der(x), izq(x) \neq NULL$ y $key[der(x)] - key[izq(x)]$ sea máximo.*
- *Encontrar el nodo $x \in T$ tal que x sea una hoja, $x \neq$ raíz(T) y $|x - PADRE[x]|$ sea mínimo.*

PARTE PRÁCTICA

Ejercicio 4 *(2) Dado un vector de n puntos en el plano $P = (\ (x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\ )$ y un entero positivo $k > 1$, con n múltiplo de k. El problema del k-particionamiento consiste en dividir P en k grupos, tales que los puntos dentro de un grupo sean muy similares de acuerdo a algún tipo de medida. Usaremos la distancia euclidiana:*

$$d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1}$$

*Para resolver este problema se puede usar el siguiente algoritmo: se inicia seleccionando k puntos de P, mismos que representarán los centro de cada grupo $C_i, \forall i = 0, \ldots, k-1$. En un proceso iterativo, se asignan a los puntos **restante** al grupo con el centro más cercano. Al final de la iteración se calcula el nuevo centro del grupo y el proceso se repite hasta que los centros no cambien.*

Como simplificación del algoritmo anterior, se propone realizar las siguientes tareas:

- *Ordenar los puntos usando el criterio $P_i < P_j$ si $d(P_i, 0) < d(P_j, 0)$.*
- *Selecciona los puntos $A = \{\frac{n}{k} - 1, 2\frac{n}{k} - 1, \ldots, n-1\}$ y tomarlos como centros de cada grupo $G_0 = \{C_0 = P_{\frac{n}{k}-1}\}, \ldots, G_{n-1} = \{C_{k-1} = P_n\}$.*
- *Asignar los objetos $P \setminus \cup_{i=1}^{k} C_i$ al grupo con centro más cercano, es decir,*
  - *Para $i \notin A$:*
    - *Calcular el grupo ℓ tal que la distancia entre el punto i y el centro del grupo C_ℓ sea mínima.*
    - *$G_\ell = G_\ell \cup \{P_i\}$*
- *Actualizar el valor del centro del cluster (media en cada componente)*

Realizar las siguientes tareas:

1. *Escribir una función que reciba 2 pares y calcule su distancia de acuerdo a (1).*
2. *Leer el archivo "Puntos.txt" y almacenarlos en un vector $< pair < double, double >>$*
3. *Implementar la simplificación del algoritmo anterior usando las siguientes estructuras:*

```
class  grupo: private list<pair<double,double> >//lista de puntos en el grupo incluido el centro
{
private:
pair<double,double> centro;
public:
...
};

class particion:private vector<grupo>
{
private:
int k;
public:
...
};
```

4. *Sobrecargar los operadores $<<$ de las clases anteriores.*

```
Centro 1: Puntos en el grupo
Centro 2: Puntos en el grupo
....
```