ESTRUCTURA DE DATOS
PARTE TEÓRICA

INDICACIONES: Disponen de 2:30 para terminar la evaluación y subirla al sistema. Planifiquen su tiempo ya que si lo entregan fuera de tiempo tendrán una reducción en la nota. Ninguna evaluación será recibida fuera de horario

- La parte teórica debe ser entregada hasta las 15:30. Entrega atrasada hasta las 15:40.
- La parte práctica debe ser entregada hasta las 16:30. Entrega atrasada hasta las 16:40.

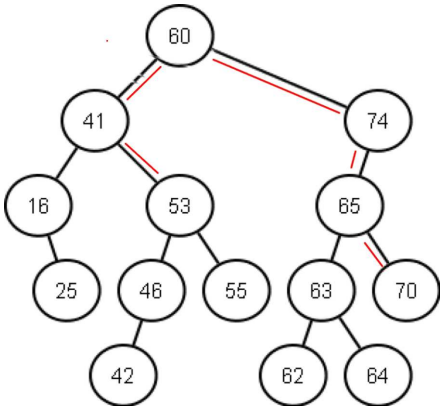
Ejercicio 1 *(2 P) Pruebe o desapruebe lo siguiente:*

- *Si* $f(n) = an^2 + bn + c$, *entonces* $f(n^2) = \mathcal{O}(n^2)$
- $log(n!) \in O(log(n))$

Ejercicio 2 *(2 P) La moda de un conjunto de números es el número de ocurrencias más frecuente en el conjunto. Dar un algoritmo eficiente que calcule la moda de un conjunto de n números. Ej: Dado el conjunto* $S = \{4, 6, 2, 4, 1, 4, 3\}$, *se tiene una moda de* 4.

Ejercicio 3 *(2 P) Dado un árbol binario de búsqueda* T *y dos nodos* $u, v \in T$. *Sea* $\pi(u, v)$ *el peso del camino entre el nodo* u *y el nodo* v *y definido como la suma de las claves asociadas a cada nodo en el camino que une el nodo* u *con el nodo* v *incluyendo dichos nodos. Cuando es* $v = u$, *entonces* $\pi(u, u) = key[u]$. *Considerando el árbol del gráfico y* $key[u] = 53$, $key[v] = 70$, *entonces el peso del camino es la suma de nodos en rojo:*

$$\pi(u, v) = 53 + 41 + 60 + 74 + 65 + 70$$



Diseñar un algoritmo en pseudocódigo que reciba un árbol T *y dos nodos* $u, v \in T$ *con* $key[u] < key[raiz[T]]$, $key[v] > key[raiz[T]]$ *y retorne el valor* $\pi(u, v)$. *Determine el orden del algoritmo diseñado.*

PARTE PRÁCTICA

Ejercicio 4 *(5P) Dado un conjunto* S *con* n *números enteros, se desea determinar si existe alguna forma de particionar* S *en dos subconjuntos* $S1$ *y* $S2 = S \setminus S1$, *tal que:*

$$minimizar \ \{|\sum_{S1} s_k - \sum_{S2} s_k| : | \ |S1| - |S2| \ | \leq 1\} \tag{1}$$

Por ejemplo, dado el conjunto $S = \{-7, -3, -4, -2, 8, 3, 1\}$, *se puede construir el subconjunto* $S1 = \{-7, -3, 8\}$ *y* $S2 = \{-4, -2, 3, 1\}$ *con suma* $\sum_{S1} s_k = \sum_{S2} s_k = -2$, $|S1| = 3$, $|S2| = 4$ *y* $| \ |S1| - |S2| \ | = 1$. *Algoritmo 1 permite resolver el presente problema. Realizar las siguientes tareas*

Algorithm 1 Particionamiento

ENTRADA:Una instancia $S = \{s_1, \ldots, s_n\}$.
SALIDA: Una solución de problema de particionamiento $(S1, S2)$
Construir una solución inicial:
* $S1$ dispone los elementos de S ubicados en las posiciones pares
* $S2$ dispone los elementos de S ubicados en las posiciones impares
$ObjBest = abs(\sum_{S1} s_k - \sum_{S2} s_k)$
for i=1 hasta $|S1|$ **do**
  for j=1 hasta $|S2|$ **do**
    $Obj = abs((\sum_{S1} s_k - S1_i + S2_j) - (\sum_{S2} s_k + S1_i - S2_j))$
    if $Obj < ObjBest$ **then**
      $ObjBest = Obj$
      Intercambiar el elemento i de $S1$ con el elemento j de $S2$
    end if
  end for
end for
Retornar $S1$ y $S2$

1. *Escribir una clase que permita almacenar un conjunto de números:*

```
class conjunto:private vector<int>
{
private:
    int suma;
public:
conjunto();//constructor
 ~conjunto();//destructor
 void CalcularSuma(); //Calcula la suma de los elementos del conjunto
 int Rango();//Retorna el rango=max-min de los datos.
 Sobrecargar <<. //Retorna todos los números y la suma.
 Sobrecargar >>. //Usar el formato que usted considere adecuado.
 };
```

2. *Leer los datos desde un archivo usando >>*

3. *Implementar el algoritmo 1 bajo la siguiente declaración:*

```
void Particionamiento(conjunto S, conjunto &S1, conjunto& S2);
```

donde se recibe un conjunto de números S y se retorna dos conjuntos con la solución del problema $(S1, S2)$