Estructuras de datos (2016-1-2): segundo parcial

Nombre y código: _____

Pregunta:	1(1)	1(2)	1(3)	1(4)*	2(1)	2(2)	3(1)	3(2)	3(3)
Puntos:	0.5	0.5	1.0	1.0	1.0	0.5	0.5	0.5	0.5
Calificación:									

Pregunta 1: Valet parking. Level M

En un parqueadero de un hotel muy costoso han decidido implementar un sistema para la gestión de los vehículos. La idea es tener organizados en todo momento los datos sobre los clientes y sus carros. Para esto, el diseñador del sistema de información decide que la mejor estructura de datos debe ser asociativa y propone un mapa (`map`). Él opina que la llave del mapa debe estar formada por la información de la marca y el color del carro. El valor asociado a cada llave tendrá el nombre del cliente, la placa del carro y el número del lugar donde este se encuentra estacionado.

```
class Car {
private:
  string brand;
  string color;
public:
  Car(string b, string c) : brand(b), color(c) {}
  /* ... More operations ... */
};
class Customer {
private:
  string name;
  string plateNumber;
  int pos;
public:
  Customer(string n, string num, int p) : name(n), plateNumber(num), pos(p) {}
};

int main() {
  map<Car,Customer> parkingInfo;
}
```

1. En la primera noche de prueba del sistema ocurre algo catastrófico. Qué problema tiene el uso de la estructura como lo propuso el diseñador?. Justifique su respuesta.

2. Qué modificaciones sobre la propuesta haría usted para poder almacenar correctamente la información?.

Gracias a sus respuestas, el diseñador acaba de salvar su empleo. Pero ahora se enfrenta a un reto mayor. El administrador del hotel es un gran economista visionario y teme que la nueva propuesta solo sirva temporalmente. Durante el primer mes han tenido 100 clientes y todo va bien. El problema es que se espera que el número de clientes aumente *cuadraticamente* mes a mes durante los próximos 10 meses. Cómo todo buen

economista debe ser un gran matemático, esa es la única forma que usted tiene de convencerlo que el sistema de parqueo del hotel no va a colapsar nuevamente.

3. Qué garantía matemática le puede dar usted sobre la solución propuesta anteriormente que le permita al administrador estar tranquilo en los próximos 10 meses?. Justifique su respuesta.

4. Mejor aún, durante cuántos meses puede estar el administrador tranquilo sobre el funcionamiento del sistema de parqueo?. Para esto usted puede asumir que el equipo donde funciona el sistema de parqueo cuenta con una CPU estándar de 2.0 GHz. Justifique su respuesta[a].

[a] 2.0 GHz equivale a 2×10^6 operaciones cada segundo.

Pregunta 2: Interval visitor. Level H

Los recorridos clásicos sobre árboles binarios incluyen *inorder*, *preorder*, *postorder* y *levelorder*. El patrón de diseño visitante (*visitor*) aplica una función arbitraria sobre cada elemento almacenado en el árbol. La combinación de ambos da como resultado la aplicación, en un orden determinado de una función sobre los elementos del árbol. Dado que los árboles de búsqueda binarios preservan el orden de la información en su estructura, debería ser posible un nuevo tipo de recorrido. A este recorrido le llamaremos `interval` y aplicará una función arbitraria `f` a un subconjunto de los elementos del árbol.

```
template<typename Key, typename Value>
class RBT {
  /* All the attributes and private definitions */
  using Function = function<void(Key,Value)>;
  public:
  void interval(const Key& lower, const key& upper, Function f) {/*...*/}
}
```

La idea es que `interval` debe recorrer todos los elementos en el árbol con llave `k` tal que `lower` $\leq k \leq$ `upper`.

1. Proponga una implementación de la operación `interval` que cumpla su objetivo y solo recorra los nodos absolutamente necesarios.

2. Qué complejidad tiene su solución?.

Pregunta 3: Heapsort v0.0. Level M

Existe un método de ordenamiento clásico llamado *heapsort*. Como su nombre lo indica, usa el concepto de un `Heap` para ordenar los elementos de una secuencia. Como requiere de dicha estructura de datos primero veamos que tal se encuentra en dicho tema.

1. Inserte los siguientes elementos secuencialmente en un `MaxHeap` y presente tanto su representación jerárquica (el árbol) como el vector que almacena sus datos.

$$\langle 10, 5, 12, 3, 2, 1, 8, 7, 9, 4 \rangle$$

El algoritmo de ordenamiento en su versión más simple y no tan óptima consiste en los siguientes pasos:

- insertar todos los elementos de la secuencia de entrada en un heap.
- Mientras el heap tenga elementos, sacar la raíz y adicionarla al final de la secuencia resultante.

2. Proponga una implementación del ordenamiento descrito anteriormente.

3. Proponga una cota para su complejidad.