UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2019-2020
Convocatoria extraordinaria de Febrero
Grado en Ingeniería Informática.

1. (1 punto) Razonar (en términos de eficiencia de las operaciones) qué TDA elegirías entre los que se indican para:

   (a) Implementar un **conjunto ordenado de enteros** (con funciones básicas de insertar, borrar, y buscar) de entre: un **ABB** (árbol binario de búsqueda), un **APO** (árbol parcialmente ordenado implementado mediante un **heap**) y un **Vector ordenado**.

   (b) Implementar un **Diccionario** (con funciones básicas de insertar, borrar y buscar) de entre: una **Tabla Hash abierta**, un **vector ordenado** y una **lista ordenada**

   (c) Implementar un **conjunto no ordenado de enteros** (con funciones básicas de insertar, borrar y buscar) de entre: un **AVL**, un **vector no ordenado**, y una **cola con prioridad (**implementada con un heap**)**.

2. (1 punto) Dadas 2 listas de enteros L1 y L2 implementar una función:

   bool check_sum (const list<int> & L1, const list<int> & L2)

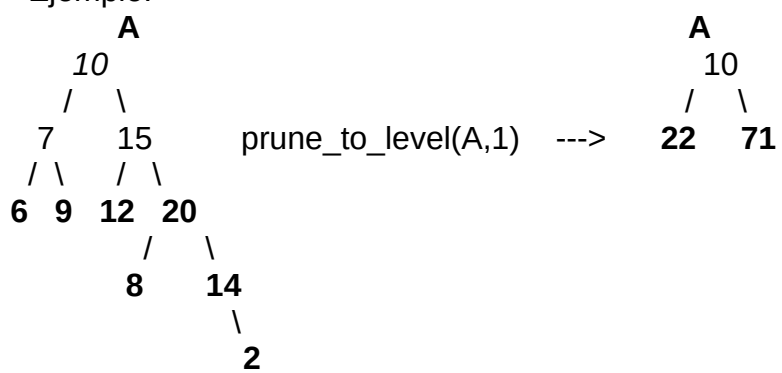
   que devuelva true si los elementos de L1 pueden agruparse sumando de forma que se puedan obtener los de L2 sin alterar el orden de los elementos.
   *P.ej: Si L1 = {**1,2,3**,4,1,3,2,**5**,6,**8,3**} y L2={ **6**, 10, **5**, 6, **11**} devolvería true*

3. (1 punto) Implementar una función

   void prune_to_level(bintree<int> & A, int level);

   que pode todos los nodos de un árbol binario A por debajo del nivel ***l*evel** sustituyendo la etiqueta de los nodos del nivel más profundo que quede tras podar por la suma de sus descendientes (incluyendo la etiqueta del propio nodo)

   Ejemplo:



4. (1 punto) Dado un AVL **A**, implementar una función, que, dado un nodo **n** en el árbol, devuelva el nodo siguiente a **n** en el recorrido en inorden

5. (1 punto) Detalla cada una de las operaciones siguientes:
   a) Insertar las claves {8, 16, 12, 41, 10, 62, 27, 65, 13} en una **Tabla Hash cerrada** de tamaño 13. A continuación borrar el 10 y finalmente insertar el valor 51. Resolver las colisiones usando **hashing doble**.
   b) Construir un **AVL** insertando, en ese orden, las claves anteriores especificando los pasos seguidos e indicando cuando sea necesario el tipo de rotación que se usa.
   c) Construir un **APO** con las claves anteriores. Borrar dos elementos en el APO resultante.

6. (1 punto) Implementar un **iterador** que itere sobre las claves que tengan asociada una lista cuyos elementos sean todos números primos en una clase Diccionario definida como:

```
class Diccionario{
   private:
     map<int, list<int> > datos;
   ..............
    ..............
};
```

Han de implementarse (aparte de las de la clase iterador) las funciones begin() y end(). Se supone implementada una función bool primo (int x) que devuelve true si el entero x es primo.

Tiempo: 3 horas

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2019-2020
Convocatoria extraordinaria de Febrero
Grado en Ingeniería Informática.

Preguntas específicas de prácticas (4 puntos) Tiempo: 1 hora

1.- (1.5 puntos) Se desea diseñar un TDA, que llamaremos **alfabeto** que representa a un conjunto de caracteres que admite repeticiones y donde las únicas operaciones son **insertar_caracter**, **borrar_caracter** y **cuantos_son_menores_que_m.** La operación insertar_caracter añade un carácter al conjunto, la operación borrar_caracter borra todas las ocurrencias del carácter que se le pase como parámetro y la operación cuantos_son_menores_que_m devuelve un valor entero indicando cuantos de los caracteres del conjunto son menores o iguales que m. Implementar el TDA alfabeto usando como base una tabla hash cerrada.

2.- (2.5 puntos) Suponer que queremos crear un **TDA razas_gatos** que nos permita obtener de forma eficiente las razas de gatos usando un conjunto de características. Un ejemplo de conjunto de características podría ser:

* Fuerte. Son gatos que no enferman fácilmente. Por ejemplo la raza Maine coon.
* Fino. Son gatos delgados. Por ejemplo Gato devon rex.
* Cola gruesa. Son gatos que tienen una cola muy gruesa. Por ejemplo Gato Curl americano.
* Dos colores son gatos que tienen dos o más colores.
* Orejas Grandes. Son gatos con las orejas grandes. Por ejemplo Gato bosque de Noruega.
*Pelo Largo. Son gatos con el pelo largo. Por ejemplo Gato Siberiano.

El ***TDA razas_gatos*** se construye a partir de un conjunto de características y un conjunto de pares (nombre raza y una secuencia de 0 y 1). Cada uno de estos valores, 0 o 1, representa si la característica es dada en la raza 1 o si no es dada en la raza 0. Por ejemplo el conjunto de características podría ser:
(Fuerte,Fino,Cola_Gruesa,dos_colores,Orejas_Grandes, Pelo Largo).
Y un ejemplo de par podría ser: ***(Siamés, 1,1,0,1,0,0)***. Esta secuencia codifica la raza Siamés como que es fuerte, es de corpulencia fina, no tiene la cola gruesa,si tiene dos colores,no tiene las orejas grandes ni el pelo largo.

Para representar las características del conjunto de razas se usará un árbol binario.
Se pide:
1.- Implementar el constructor para razas_gatos dado el conjunto de características y conjunto de pares (nombre raza, secuencia de 0 y 1).
2.- Dadas una secuencia de características obtener la raza o conjunto de razas que responden a esas secuencia de características.
3.- Construir un método que te informe si el conjunto de características determina de forma univoca a cada raza.
4.- Construir un método que te informe si el conjunto de características es minimal. Un conjunto de características es minimal si al eliminar una característica, el conjunto de razas no se determina de forma unívoca.
5.- Suponer que queremos crear una nueva raza dadas dos razas que tenemos almacenadas. Crear un método que dadas las dos razas obtenga el vector de características de la raza cruce. Se supone que existe una probabilidad entre 0 y 1 de transmitir un progenitor su característica a su descendiente. Para ello se genera un numero entre 0 y 1 para cada raza y su vector de características se multiplica por esta probabilidad. Finalmente se suman los dos vectores teniendo en cuenta no superar el valor de 1. Si el valor es mayor que 0.5 se redondea a 1 y si es menor o igual que 0.5 a 0.