UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2021-2022
Convocatoria ordinaria de Enero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

1. (0.75 puntos)
   A) Dadas las 3 siguientes afirmaciones:
      (a) En un esquema de hashing doble puede usarse como función hash secundaria
      $$h_0(k)=[\ M-(k\%M)\ -1\ ]\%M, \text{ con M primo.}$$
      (b) Es correcto hacer la siguiente definición **set <stack <int>> :: iterator q;**
      (c) Un **APO** puede reconstruirse de forma unívoca dado su recorrido en preorden.
      Indica
      indica la respuesta correcta (**Razonando la respuesta**):
      (a) Las tres son ciertas. **(b)** Dos son ciertas y una falsa **(c)** Dos son falsas y una cierta
      (d) Las tres son falsas.
   B) Si busco el elemento máximo en un APO con el mínimo en la raíz:
      (a) El elemento debe estar necesariamente en una hoja.
      (b) Encontrarlo lleva un tiempo O(n)
      indica la respuesta correcta (**Razonando la respuesta**):
      (a) Las dos afirmaciones son ciertas **(b)** Una es cierta y la otra falsa **(c)** Las dos son falsas.
   C) Supongamos que hacemos las 2 siguientes afirmaciones:
      (a) En un árbol binario el número de nodos con 2 hijos más uno es igual al número de hojas.
      (b) Para cualquier valor k, la estructura final de un AVL que se construye insertando los enteros {1,2,…, 2^k -1}, en ese orden, es la de un árbol completo.
      indica la respuesta correcta (**Razonando la respuesta**):
      (a) Las dos afirmaciones son ciertas **(b)** Una es cierta y la otra falsa **(c)** Las dos son falsas.

2. (1.25 puntos) Tenemos una clase **libro** que permite gestionar las palabras de un libro. Para ello usa un **map<string, list<pair<int,int> > >** de forma que para cada palabra hay una lista **list<pair<int,int> >** dónde cada par contiene un número de capítulo y la posición dentro del mismo en la que aparece dicha palabra.
   A) Implementa un método **convertir_vector** que devuelva **vector<list<string> >** de forma que en la posición **i** contenga todas las palabras del capítulo **i+1** ordenadas alfabéticamente y sin repeticiones. Por ejemplo, si tenemos el map:
      <informática, {<**1**,10>,<**2**,10>,<**3**,41>}>
      <telemática, {<**1**,2>,<**1**,21>,<**2**,50>,<**3**,31> }>
      <robótica, {<**3**,30>,<**4**,5>}>
   el vector contendría:
   v[0]={informática,telemática,telemática} ->**v[0]={informática,telemática}**
   v[1]={informática, telemática}
   v[2]={informática,robótica,telemática}
   v[3]={robótica}

   B) Implementa una clase **iterator** que itere sobre las palabras del libro que aparezcan en capítulos impares y en posiciones pares de ese capítulo. Implementa los métodos de la clase iterator y los métodos **begin()** y **end()** de la la clase libro.

UNIVERSIDAD DE GRANADA
Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2021-2022
Convocatoria ordinaria de Enero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

3. (1 punto) Implementa una función:

   void resumecola(list<int> &L,queue<int> &Q);

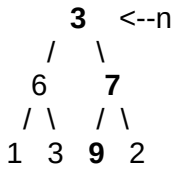
   que va tomando un elemento entero n de la cola Q y, si es estrictamente positivo, saca n elementos de L (si ya no quedan n elementos, saca todos los que queden) y los reemplaza por su producto. Si el elemento de la cola es negativo o cero, no hace nada. Esto ocurre con todos los elementos de Q hasta que se acaben, o bien se acaben los elementos de L. **No pueden usarse estructuras auxiliares**.
   Por ejemplo: Si L=(**1,3,2**,1,4,**5,3**,2,4,1) y Q=(**3**,2,-1,0,**2**,5,2,-3) entonces L debe quedar así: L=(**6**,4,**15**,8), y Q=(2,-3) (es decir, sobran elementos de Q).
   Otro ejemplo: Si L=(**1,3,2**,1,4,5,3,**2,4,1,3,2**,1,4,7) y Q=(**3**,2,-1,0,2,**5**) entonces L debe quedar así L=(**6**,4,15,**48**,1,4,7), y Q=() (es decir, sobran elementos de L que quedan como estaban en la lista)

4. (1 punto) Dado un árbol binario A y un nodo n de ese árbol, implementa una función:

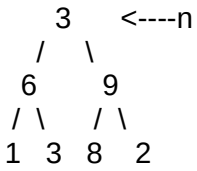
   bool secuencia_creciente (const bintree<int> &A, bintree<int>::node n);

   que devuelva true si existe algún camino desde n a una hoja en la que se cumpla que cada etiqueta de un nodo hijo tiene un valor mayor que la del nodo

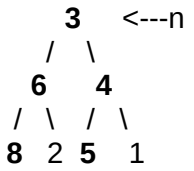
   Ejemplo:



true



false



true

5. (1 punto) Dado un **vector de conjuntos** vector<set<int> > V, implementa una función

   void esta_en_conjunto(vector<set<int> > &v, map<int,list<int> > &recuento);

   que devuelva a través del map **recuento** cada uno de los elementos que aparecen en algún conjunto y una lista de las posiciones del vector en las que se puede encontrar.

   Por ejemplo, si v=[{1,20,3}, {20,3,45}, {3,45,5,93}, {20,45,6,8}, {93}, {1,3,5}] entonces debe devolver

   recuento={<**1**,{0,5}>, <**20**, {0, 1,3}>, <**3**, {0,1,2,5}>, … , <**5**, {2,5}>, …. }

6. (1 punto)

   A) Inserta (detallando los pasos) las siguientes claves (insertadas en ese orden) en un **AVL**: {2, 1, 4, 5, 9, 3, 6, 7} especificando los pasos seguidos e indicando cuando sea necesario el **tipo de rotación** que se usa para equilibrar y mantener la estructura de AVL

   B) Inserta (detallando los pasos) las siguientes claves (en el orden indicado):

   {51, 35, 53, 70 ,54 ,56, 86, 42, 11, 67, 57}

   en una **tabla hash cerrada** de tamaño 13 con resolución de colisiones usando hashing doble.

Tiempo: 2.30 horas

Soluciones al examen de Estructuras de Datos de 21 de Enero de 2022

Pregunta 1

(A) (a) Falso. No puede usarse porque se anula

(b) Verdadero. No hay problema en usar iteradores sobre un set independientemente del tipo de dato interno del set

(c) Verdadero. Por la condición geométrica del APO que permite conocer la estructura interna que va a tener

Por tanto la respuesta correcta es 1.b

(B) La respuesta correcta es 2.b: No es cierto que los elementos de mayor valor en un APO-min se sitúan necesariamente en las hojas porque si hay elementos repetidos, pueden estar en un nodo interior. Si es cierto que localizarlos conlleva un rastreo lineal del árbol (por niveles p.ej) que conlleva a que sea un procedimiento O(n)

(C) La respuesta correcta es 3.a: Ambas son ciertas. la primera es una propiedad de todo árbol binario y la segunda por el proceso de rotación de nodos que en ese caso va colocando la mediana en cada nodo raíz.

Pregunta 2

```
#include <iostream>
#include <map>
#include <list>
#include <vector>
#include <algorithm>
#include <set>

using namespace std;

class Libro{
private:
   map<string,list<pair<int,int> > >datos;
public:
   //Construyo con los valores del ejemplo
   Libro(){
       list<pair<int,int> > l1={pair<int,int>(1,10),pair<int,int>(2,10),pair<int,int>(3,41)};
       list<pair<int,int> > l2={pair<int,int>(1,2),pair<int,int>(1,21),pair<int,int>(2,50),pair<int,int>(3,31)};
       list<pair<int,int> > l3={pair<int,int>(3,30),pair<int,int>(4,5)};

       datos.emplace("informatica",l1);
       datos.emplace("telematica",l2);
       datos.emplace("robotica",l3);
   }
   vector<list<string>> convertir_vector()const{
      map<int,set<string> > aux; //mapa con clave capitulo y palabras
      for (auto it=datos.begin();it!=datos.end();++it){
         for (auto itlist=it->second.begin();itlist!=it->second.end();++itlist){
            auto pos=aux.find((*itlist).first);
            if (pos!=aux.end())
               pos->second.insert(it->first);
             else{
               set<string>s; s.insert(it->first);
               aux.emplace((*itlist).first,s);
             }
         }
      }
      vector<list<string> > out;
      for (auto it=aux.begin();it!=aux.end();++it){
         list<string>laux;
         for (auto i= it->second.begin();i!=it->second.end();++i)
           laux.push_back(*i);
         out.push_back(laux);
      }
      return out;
   }
   class iterator{
   private:
       map<string,list<pair<int,int> > >::iterator it,final;
       list<pair<int,int> >::iterator it_l;

       bool condicion(pair<int,int> d){
          return (d.first%2==1 and d.second%2==0);
       }
   public:
      iterator(){}
      bool operator==(const iterator &i)const{
         if (i.it==final && it==final) return true;
         else if (i.it==it && i.it_l == it_l) return true;
         else return false;
      }
```

```
        bool operator!=(const iterator &i)const{
           return !(*this==i);
        }
        pair<const string,pair<int,int> >  operator*(){
           pair<const string,pair<int,int> > out(it->first,*it_l);
           return out;
        }
        iterator & operator ++(){
           do{
              ++it_l;
              if (it_l==it->second.end()){
                 ++it;
                 if (it!=final)
                   it_l=it->second.begin();

              }

           }while (it!=final && !condicion(*it_l));

          return *this;
        }
        friend class Libro;
    };

    iterator begin(){
     iterator i;
     i.it=datos.begin();
     i.final=datos.end();
     if (i.it!=datos.end()){
        i.it_l = i.it->second.begin();
        if (!i.condicion(*(i.it_l))) ++i;
     }
     return i;
    }

    iterator end(){
     iterator i;
     i.it=datos.end();
     i.final=datos.end();

     return i;
    }

};

int main(){
 map<string,list<pair<int,int> > >datos;
 Libro mibook;
 vector<list<string> > v=mibook.convertir_vector();
 cout<<"Comprobando funcion...."<<endl;
 for (int i=0;i<v.size();i++){
    cout<<"Chapter "<<i+1<<": ";
    for_each(v[i].begin(),v[i].end(), [](string &a){ cout<<a<<'\t';});
    cout<<endl;
 }
 cout<<"Comprobando iteradores...."<<endl;
 //Probando el iterator
 for (Libro::iterator i = mibook.begin(); i!=mibook.end();++i)
    cout<<(*i).first<<" "<<(*i).second.first<<" "<<(*i).second.second<<endl;

}
```

Pregunta 3

```
#include <list>
#include <iostream>
#include <queue>

using namespace std;

template <typename T>
void Imprimir(T comienzo,T fin){
  for (auto it =comienzo; it!=fin; ++it){
    cout<<*it<<" ";
  }
}
void resumecola(list<int> &L,queue<int> &Q){
  auto it=L.begin();
  while (!Q.empty() && it!=L.end()){
      int a=Q.front();
      Q.pop();
      if (a>0){
         auto it2=it;
         int cnt=0;
         int producto=1;
         while (it2!=L.end() && cnt<a){
            producto*=*it2;
            it2=L.erase(it2);
            ++cnt;
         }
         it2=L.insert(it2,producto);
         it=it2;
         ++it;
      }
  }
}

int main(){
  //Ejemplo 1
 /* list<int>L={1,3,2,1,4,5,3,2,4,1};
  queue<int>Q;
  Q.push(3);Q.push(2); Q.push(-1);
  Q.push(0);Q.push(2);Q.push(5);
  Q.push(2); Q.push(-3); */

  //Ejemplo 2
  list<int>L={1,3,2,1,4,5,3,2,4,1,3,2,1,4,7};
  queue<int>Q;
  Q.push(3);
  Q.push(2);Q.push(-1);Q.push(0);
  Q.push(2); Q.push(5);

  cout<<"Original "<<endl;
  Imprimir(L.begin(),L.end());
  resumecola(L,Q);
  cout<<endl;
  cout<<"La secuencia compacta : ";
  Imprimir(L.begin(),L.end());
}
```

Pregunta 4

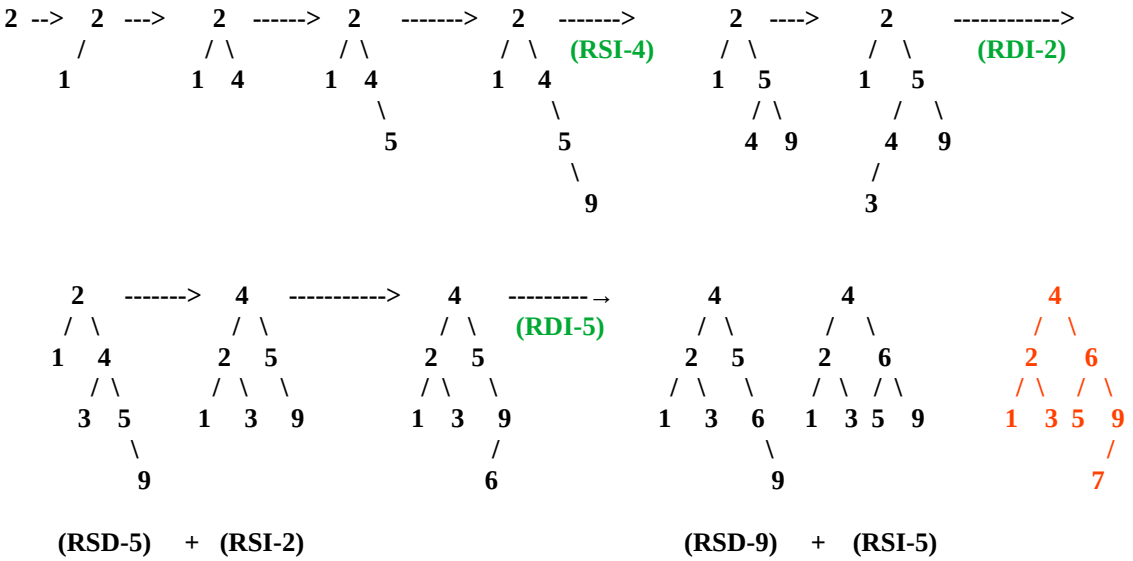
```
#include <iostream>
#include "bintree.h"

using namespace std;


bool secuencia_creciente (bintree<int> &a,bintree<int>::node n){
  if (n.null()){
    return true;
  }
  else{
      bool pi=false,pd=false;

      if (n.left().null() && n.right().null())
        return true;

      if (!n.left().null() && *(n.left())>*n)
        pi=secuencia_creciente (a,n.left());

      if (pi) return true;

      if (!n.right().null() && *(n.right())>*n)
        pd=secuencia_creciente (a,n.right());

      return pd;

  }
}

int main(){
 // Creamos el árbol:
 //          1
 //        /   \
 //       8     5
 //            / \
 //           14   7
 //            \
 //             22

 bintree<int> Arb(1);
 Arb.insert_left(Arb.root(), 8);
 Arb.insert_right(Arb.root(), 5);
 Arb.insert_right(Arb.root().right(), 7);
 Arb.insert_left(Arb.root().right(), 14);
 Arb.insert_right(Arb.root().right().left(), 22);

 if (secuencia_creciente(Arb,Arb.root()))
   cout<<"Se cumple la condición "<<endl;
 else
   cout<<"No se cumple la condición"<<endl;
}
```

Pregunta 5

```
#include <vector>
#include <set>
#include <list>
#include <map>
#include <iostream>
using namespace std;

void esta_en_conjunto(vector<set<int> > &v, map<int,list<int> > &recuento){
  set<int>diferentes_ele; //obtenemos los elementos de forma unica
  for (int i=0;i<v.size();i++)
   for (auto it = v[i].begin();it!=v[i].end();++it)
     diferentes_ele.insert(*it);

  //recorremos los diferentes elementos y lo contabilizamos
  for (auto it = diferentes_ele.begin();it!=diferentes_ele.end();++it){
     recuento[*it]=list<int>();
     for (int i=0;i<v.size();i++)
        if (v[i].count(*it)>0)
         recuento[*it].push_back(i);

  }

}

int main(){
 set<int>s1={1,20,3},s2={20,3,45},s3={3,45,5,93},
         s4={20,45,6,8},s5={93},s6={1,3,5};

 vector<set<int> > miv={s1,s2,s3,s4,s5,s6};

 map<int,list<int> > re;
 esta_en_conjunto(miv,re);

 for (auto it =re.begin(); it!=re.end();++it){
  cout<<"Elemento "<<it->first<<" Aparece en ";
  for (auto it2=it->second.begin();it2!=it->second.end();++it2)
     cout<<*it2<<"\t";
  cout<<endl;
 }
}
```

Pregunta 6

(a) AVL:{2,1,4,5,9,3,6,7}



(b) Tabla hash cerrada

k	51	35	53	70	54	56	86	42	11	67	57
h(k)	12	9	1	5	2	4	8	3	11	2	5
h_o(k)	8	3	10	5	11	2	10	10	1	2	3

h(k) =k%13
h_o(k)=1+k%11

h_i(k)=[h_i-1 (k) + h_o (k)]%13

Todas se insertan a la primera excepto:

h_2(67)=4
h_3(67)=6

h_2(57)= 8
h_3(57)= 11
h_4(57)= 1
h_5(57)= 4
h_6(57)= 7

Tabla resultante

0	1	2	3	4	5	6	7	8	9	10	11	12
	53	54	42	56	70	67	57	86	35		11	51