UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2022-2023
Convocatoria ordinaria de Enero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

1. (1 punto) **(a)** Si insertamos un conjunto de enteros **ordenado** ascendentemente en un ABB, un APO-min y un AVL: ¿En cual de los 3 es más eficiente la operación de inserción? **Razonarlo.**

   (b) (b1) ¿Cuántos elementos, en promedio, hay en cada lista en una tabla hash abierta de tamaño B con N elementos ? **(b1)** (N/B) **(b2)** (N + B) **(b3)** (B/N) **(b4)** 1. **Razonarlo.**

   (b2) ¿Es correcto en un esquema de hashing cerrado con un tamaño de la tabla **B** primo con resolución de colisiones usando hashing doble, el uso como función hash:

   h(x)=[(x % C) % B] ? ¿Y como función hash secundaria $\mathbf{h_0(x)=[(x*C) \% (B-2)]}$ **?** B, C primos entre si. **Razonarlo**

   (c) Dadas las siguientes 3 afirmaciones:

   - Dados A y B dos árboles binarios (con más de un nodo) distintos con etiquetas diferentes, nunca puede ocurrir simultáneamente: Pre (A) = Post (B) y Post (A)=Pre (B)
   - Un APO puede reconstruirse de forma unívoca dado su recorrido en postorden
   - Solo hay un APO que tiene como preorden={4,9,24,33,21,74,63}

   (c1) Todas son falsas **(c2)** Hay 2 ciertas y 1 falsa **(c3)** Hay 1 cierta y dos falsas **(c4)** Todas son ciertas. **Razonar la respuesta.**

   (d) Dados los siguientes recorridos en **preorden** = (A,Z,W,R,X,Q,T,Y,L,V), y **postorden** = (R,W,Q,X,Z,Y,V,L,T,A) de un árbol binario **(d1):** No hay ningún árbol binario con esos recorridos asociados; **(d2):** Hay 1 solo árbol binario con esos recorridos asociados; **(d3):** Hay exactamente dos árboles binarios con esos recorridos asociados; **(d4):** Todo lo anterior es falso. **Razonar la respuesta.**

2. (1 punto) Se desea construir un **traductor** de un idioma origen a un idioma destino. Una palabra en el idioma origen puede tener más de una traducción en el idioma destino.

   Usando como **representación** para el **TDA Traductor** un **map<string,set<string> >:**

   • Implementar una **clase iteradora** dentro de la clase Traductor para **dada una palabra** en el idioma de destino, iterar por todas las palabras del idioma de origen que tengan como traducción esa palabra. Han de implementarse (aparte de las de la clase iteradora) las funciones **begin** y **end**

3. (1 punto) Implementar una función

   void intercambia_sec (list<int>& L);

   que dada una lista L, intercambie el grupo de los primeros elementos consecutivos impares por el siguiente grupo de elementos consecutivos pares de principio a final de la lista. **No pueden usarse estructuras de datos auxiliares.**

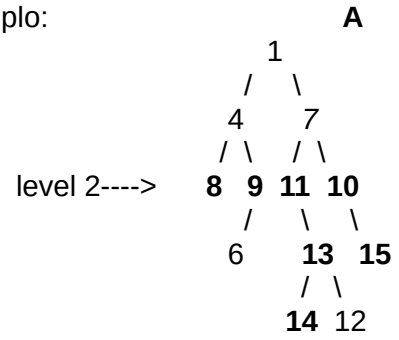
   Por ejemplo si L={**1**,2,4,**5**,6,8,**7,9,13**,2,**9**} después de llamar a intercambia_sec (L) debe quedar L={**2, 4**, 1, **6, 8**, 5, **2**, 7, 9, 13,9}

4. (1 punto) Dado un árbol binario de enteros A y un nivel en el árbol **level**, implementar una función:

   vector<list<int>> nivel_caminos_creciente (bintree<int> A, int level);

   que encuentre los caminos de longitud mayor o igual que uno **desde cualquier nodo** de ese nivel a una hoja en el árbol **que tengan una secuencia creciente de valores de etiquetas**. Devolver la solución en un vector de listas.

   Ejemplo:

   V=nivel_caminos_creciente **(**A, 2)

   V=**{** {11,13,14 }, {10, 15} **}**

5. (1 punto) Implementar una función:

   void solo_en_2(vector<set<int> > &VS,set<int> &S1);

   que dado un vector de conjuntos enteros VS, encuentre el conjunto S1 de todos aquellos elementos que están exactamente en dos de ellos.

   Por ejemplo,si VS=[{**0**,1,**2**,3}, {1,3,**4,**5}, {1,3,6,7}, {**2**,**4**,7,**9**}, {**0**,7,8,**9**}], entonces S1={**0,2,4,9**}

6. (1 punto) (a) Insertar (en ese orden) las claves {3, 11, 15, 37, 8, 6} en una **Tabla Hash cerrada** de tamaño 13 usando como función hash **h(k) = k % 13**. A continuación borrar el 15 y finalmente insertar el valor 21. Resolver las colisiones usando **hashing doble.**
   (b) Construir un **AVL** con las claves {7, 6, 9, 10, 14, 8, 11}, especificando los pasos seguidos e indicando cuando sea necesario el **tipo de rotación** que se usa para equilibrar y mantener la estructura de AVL.

Tiempo: 2.30 horas

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2022-2023
Convocatoria ordinaria de Enero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

SOLUCIONES EXAMEN CONVOCATORIA ENERO 2023

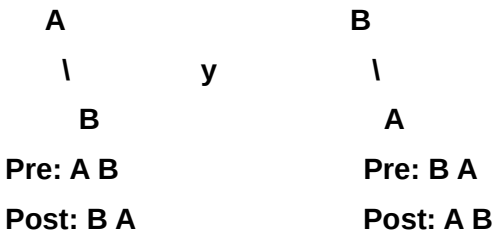
Pregunta 1

1.a Es más eficiente en el APO en que sería puramente O(1) al no tener que hacer ningún intercambio padre-hijo al quedar cada nuevo nodo insertado en el lugar que le corresponde por hacerse una secuencia de inserción creciente.

1.b1. La respuesta correcta es la b.1. En promedio los datos que habrá en cada cubeta será el número total de datos dividido por el número total de cubetas

1.b2. Sería correcto usar esa función hash primaria si C>B, pero no sería correcto usar esa secundaria porque podría anularse y en ese caso no serviría de nada.

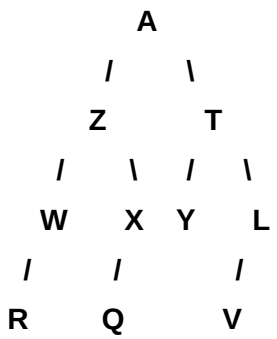
1.c La respuesta correcta es c2. Dos son son ciertas y una falsa. La primera es falsa porque en los árboles



se cumple la afirmación.

La segunda y la tercera son ciertas, consecuencia de la estructura geométrica del APO. Al tener las hojas empujadas a la izquierda sabemos a priori su estructura interna para n datos cualesquiera de forma que un solo recorrido sea preorden o postorden lo podemos reconstruir.

1.d. La respuesta correcta es la d4. Todas son falsas. La estructura jerárquica:



tiene esos recorridos y cambiando (en diferentes combinaciones posibles) R, Q, V a la derecha salen múltiples árboles binarios distintos todos con esos dos recorridos preorden y postorden

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2022-2023
Convocatoria ordinaria de Enero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

Pregunta 2

```
#include <iostream>
#include <set>
#include <map>
#include <vector>
using namespace std;
class Traductor{
private:
    map<string,set<string> > datos;
public:
    Traductor();
    Traductor(const vector<pair<string,string> >& d){  //No es necesario. Solo está hecho para poder probar el
                                                        //iterador en una función main()
      for (int i=0;i<d.size();i++){
        auto it =datos.find(d[i].first);
        if (it!=datos.end()){
          it->second.insert(d[i].second);
        }
        else {
          set<string> s1;
          s1.insert(d[i].second);
          datos.emplace(d[i].first,s1);
        }
      }
    }
    class iterator{
    private:
      map<string,set<string>>::iterator it,final;
      string word;
      bool is_translated()const{
        return (it->second.find(word)!=it->second.end());
      }
    public:
      iterator(){
        word="";
      }
      bool operator==(const iterator & i)const {
        return i.it==it && i.word==word;
      }
```

```
    bool operator!=(const iterator & i)const {
       return i.it!=it || i.word!=word;
    }
    pair< const string,set<string> > &operator*(){
       return *it;
    }
    pair< const string,set<string> > *operator->(){
       return &(*it);
    }
    iterator & operator++(){
       do{
         ++it;
       }while (it!=final && !is_translated());
       return *this;
    }
    friend class Traductor;
};

iterator begin(const string &w){  //se itera sobre las palabras para las que coincida su traducción w
    iterator i;
    i.it=datos.begin();
    i.word = w;
    i.final =datos.end();
    if (i.it!=i.final && !i.is_translated()) ++i;
    return i;
}
iterator end(const string &w){
    iterator i;
    i.it=datos.end();
    i.final =datos.end();
    i.word = w;
    return i;
}
};
```

```
int main(){
   vector<pair<string,string> > d1;
   d1.emplace(d1.end(),"hello","hola");
   d1.emplace(d1.end(),"hi","hola");
   d1.emplace(d1.end(),"hey","hola");
   d1.emplace(d1.end(),"flower","flor");
   d1.emplace(d1.end(),"blossom","flor");
   d1.emplace(d1.end(),"blossom","florecer");
   d1.emplace(d1.end(),"posy","flor");
   d1.emplace(d1.end(),"posy","ramillete de flores");
   d1.emplace(d1.end(),"orange","naranjo");
   d1.emplace(d1.end(),"orange","naranja");

   Traductor T(d1);

   Traductor::iterator it;
   for (it=T.begin("flor"); it!=T.end("flor");++it){
      cout<<"Palabra origen "<<it->first<<"  Palabras destino: " ;
      for (auto it2=it->second.begin(); it2!=it->second.end();++it2){
         cout<<*it2<<" ";
      }
      cout<<endl;
   }

}
```

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2022-2023
Convocatoria ordinaria de Enero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

Pregunta 3

```
#include <iostream>
#include <list>
#include <algorithm>
using namespace std;

void intercambia_sec(list<int> &L){
   auto it_i_start=L.begin();
   while (it_i_start!=L.end()){
      if ((*it_i_start)%2==1){
         auto it_i_end =it_i_start; ++it_i_end;
         while (it_i_end !=L.end() && (*it_i_end)%2==1) ++it_i_end;
         if (it_i_end!=L.end()){          //si no hay siguiente secuencia par
            auto it_p_start = it_i_end;
            auto it_p_end = it_p_start; ++it_p_end;
            while (it_p_end !=L.end() && (*it_p_end)%2==0) ++it_p_end;
                  //Se podria hacer sin splice con erase e insert
            L.splice(it_i_start,L,it_p_start,it_p_end);
            it_i_start = it_p_end;
         }
         else return ;
      }
      else ++it_i_start;
   }
}

int main(){
   list<int> L = {1,2,4,5,6,8,7,9,13,2,9};
   intercambia_sec(L);
   for_each(L.begin(),L.end(), [] (int a) {cout<<a<<" ";} );
}
```

Estructuras de datos. Curso 2022-2023
Convocatoria ordinaria de Enero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

Pregunta 4

```
#include<iostream>
#include "bintree.h"
#include <queue>
#include <list>
#include <vector>
using namespace std;

vector<list<int> > secuencias_crecientes (bintree<int>::node n){
  if (n.null()){
    return vector<list<int>>();
  }
  else  //es hoja
   if(n.left().null() && n.right().null()){
    list<int> vres={*n};
    vector<list<int>> vout;
    vout.push_back(vres);
    return vout;
   }
   else{
      vector<list<int> > vhi,vhd,vres;
      if (!n.left().null() && *(n.left())>*n)
        vhi=secuencias_crecientes (n.left());
      if (!n.right().null() && *(n.right())>*n)
        vhd=secuencias_crecientes (n.right());

      for (int i=0;i<vhi.size();i++){
         vhi[i].push_back(*n);
         vres.push_back(vhi[i]);
      }
      for (int i=0;i<vhd.size();i++){
         vhd[i].push_back(*n);
         vres.push_back(vhd[i]);
      }
      return vres;
 }
}
```

Estructuras de datos. Curso 2022-2023
Convocatoria ordinaria de Enero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

```
vector<list<int>> nivel_caminos_creciente (bintree<int> & A, int level){
 using mipair= pair< bintree<int>::node,int>;

 queue<mipair>micola;
 mipair a(A.root(),0);
 micola.push(a);

 vector<list<int>>vout;
 bool seguir=true;

 while (!micola.empty() && seguir){
  a=micola.front();
  micola.pop();
  if (a.second==level){
     auto v=secuencias_crecientes (a.first);
     for (int i=0;i<v.size();i++){
        if (v[i].size()>1){
         v[i].reverse();
         vout.push_back(v[i]);
        }
     }
  }
  if (a.second>level) seguir=false;
  else{
     mipair h;
     if (!a.first.left().null()){
        h.first=a.first.left();
        h.second=a.second+1;
        micola.push(h);
     }
     if (!a.first.right().null()){
        h.first=a.first.right();
        h.second=a.second+1;
        micola.push(h);
     }
  }
 }
 return vout;
}
```

```
int main(){
//
//
//          Ejemplo:                          A
//                                      1
//                                     /    \
//                level 1---->     4       7          V= nivel_caminos_creciente (A, 1)
//                                  /   \
//                                 8    3               V={ {4,8} }
//


 bintree<int> a(1);

 a.insert_left(a.root(), 4);
 a.insert_right(a.root(), 7);
 a.insert_left(a.root().right(), 3);
 a.insert_left(a.root().left(), 8);

auto v=nivel_caminos_creciente(a,1);
for (int i=0;i<v.size();i++){
   cout<<endl;
   for (auto it = v[i].begin();it!=v[i].end();++it)
      cout<<*it<<" ";

 }
}
```

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Pregunta 5

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

void solo_en_2(vector<set<int> > &VS,set<int> &S1){
   //Creamos un mapa con los elementos diferentes y sus frecuencias
   map<int,int> freq;
   for (int i=0;i<VS.size();i++){
      for (auto it=VS[i].begin();it!=VS[i].end();++it){
         auto pos= freq.find(*it);
         if (pos!=freq.end()){
            pos->second++;
         }
         else {
            freq.emplace(*it,1);
         }
      }
   }
   //cogemos solamente aquellos que tengan frequencia 2
   for (auto it = freq.begin();it!=freq.end();++it){
      if (it->second==2)
         S1.insert(it->first);
   }
}
```

```
int main(){
   //[{0,1,2,3}, {1,3,4,5}, {1,3,6,7}, {2,4,7,9}, {0,7,8,9}],
   vector<set<int> > VS;
   set<int> s1={0,1,2,3};
   VS.push_back(s1);
   set<int> s2={1,3,4,5};
   VS.push_back(s2);
   set<int> s3={1,3,6,7};
   VS.push_back(s3);
   set<int> s4={2,4,7,9};
   VS.push_back(s4);
   set<int> s5={0,7,8,9};
   VS.push_back(s5);

   set<int> Sol;
   solo_en_2(VS,Sol);
   for (auto it = Sol.begin();it!=Sol.end();++it){
       cout<<*it<<" ";
   }
}
```

Pregunta 6

6.a

k	3	11	15	37	8	6						21
h(k)	3	11	2	11	8	6						8
h_o(k)	4	1	5	5	9	7						11

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 las claves 37 y 8:

h_2(37) = (11 + 5) % 13 = 3 colisión
h_3(37) = (3 + 5) % 13 = **8**
h_2(8) = (8 + 9) % 13 = 4

Tabla resultante

0	1	2	3	4	5	6	7	8	9	10	11	12
		15	3	8		6		37			11	

Ahora borramos el 15:

Tabla resultante

0	1	2	3	4	5	6	7	8	9	10	11	12
		B	3	8		6		37			11	

Ahora insertamos el 21:

h(21)= 21%13= 8 colisión.
h_2(21)= (8+11)% 13 = 6 colisión
h_3(21) = (6+11)%13= 4
h_4(21)= (4+11)%13= 2

Tabla resultante:

0	1	2	3	4	5	6	7	8	9	10	11	12
		21	3	8		6		37			11	

(6.b) AVL:{7,6,9,10,14,8,11}