Práctica Final: WordDance

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

DECSAI

Estructuras de Datos

Grado en Ingeniería Informática. Grupo D

Índice de contenido

1. Introducción

Los objetivos de este guión de prácticas son los siguientes:

- Resolver un problema eligiendo la mejor estructura de datos para las operaciones que se solicitan

Los requisitos para poder realizar esta práctica son:

1. Haber estudiado el Tema 1: Introducción a la eficiencia de los algoritmos
2. Haber estudiado el Tema 2: Abstracción de datos. Templates.
3. Haber estudiado el Tema 3: T.D.A. Lineales.
4. Haber estudiado el Tema 4: STL e Iteradores.
5. Haber estudiado estructuras de datos jerárquicas: Árboles

2. Objetivo.

El objetivo de esta práctica es llevar a cabo el análisis, diseño e implementación de un proyecto. Con tal fin el alumno abordará un problema donde se requiere estructuras de datos que permiten almacenar grandes volúmenes de datos y poder acceder a ellos de la forma más eficiente.

3. Ejercicio

El ejercicio consiste en obtener un programa que implemente el juego ***WordDance.***

WordDance es un juego en el que se enfrentan dos jugadores. Partiendo de un conjunto de palabras el programa escoge una palabra del conjunto. El jugador que tiene el turno escoge una palabra de entre las restantes, que se diferencia de la anterior palabra en una letra. A continuación se pasa el turno al otro jugador. El jugador, con el turno, que no es capaz de obtener una nueva palabra, que se diferencie a la anterior en una letra pierde. Un ejemplo del proceso del juego se muestra en el recuadro a la derecha.

```
1.- Se escoge una palabra inicial
       MORA
2.- Turno para el jugador 1. Propone como
palabra siguiente:
       HORA
3.- Turno para el jugador 2. Propone como
palabra siguiente:
       HOLA
4.- Turno para el jugador 1. Propone como
palabra siguiente:
       COLA
...
```

4. Programa WordDance

Un ejemplo de llamada a este programa desde la línea de órdenes sería:

```
prompt% bin/WordDance datos/dic_spanish.txt 0 Rosa 1 Machine 4 datos/ranking.txt
```

Los parámetros de entrada son los siguientes:

1. El nombre del fichero con el conjunto de palabras. El formato del fichero se ha descrito en las prácticas 2 y 3.

2. El tipo de jugador 1. El valor introducido será:
    - 0: si es un jugador humano
    - 1: si es un jugador máquina. Para escoger la siguiente palabra sigue una lógica simple.
    - 2: jugador máquina. Para escoger la siguiente palabra sigue un proceso inteligente.
3. El nombre del jugador 1.
4. El tipo de jugador 2 (las mismas posibilidades descritas para el tipo de jugador 1)
5. El nombre del jugador 2
6. La longitud de las palabras. Las palabras con dicha longitud serán las que se usarán para jugar. El resto se puede obviar ya que por la propia mecánica del juego si por ejemplo se escoge longitud tres todas las palabras que suceden en el juego serán de longitud 3.
7. El nombre del fichero donde se lee/escibe el ranking global del juego. En este fichero aparecen los nombres de los jugadores y las partidas que han ganada.

En primer lugar la ejecución de este comando muestra mensajes de inicialización, para a continuación mostrar la primera palabra, en el ejemplo del recuadro sería “mora”. Además lista el conjunto de palabras que quedan (“mora” ya no debe aparecer en este conjunto). El jugador con el turno escogerá una palabra de este conjunto que se diferencie con “mora” en sólo un carácter. Si el jugador es HUMANO el programa le pedirá que introduzca la siguiente palabra. En el ejemplo la palabra introducida es “hora”. El programa chequea que es una palabra válida y pasa el turno al siguiente jugador. En caso de que no sea válida le pedirá al jugador HUMANO que introduzca una palabra.

```
Creados jugadores
Leidas las palabras
Creado juego
Las palabras dichas son:
mora

Escoge una palabra entre el siguiente conjunto:
acto    afan    agil    agua    aire    alto    alza    amar
amor    aqui    arar    arco    area    arte    auge    auto
azar    azul    bajo    base    beso    bien    boca    boda
bola    bota    bote    cabo    cada    caer    cafe    caja
.       .       .
La ultima palabra fue mora
/************************************/
Rosa>>
Cual es tu palabra? hora
```

Si el

jugador es MAQUINA escoge su palabra usando su lógica (o estrategia).

El programa de nuevo muestra la secuencia de palabras que han ocurrido en el juego y las palabras restantes, y pasa el turno al jugador "Rosa". En el caso de que el jugador no pueda obtener una siguiente palabra insertará 0. En este caso el programa indicará quien ha ganado. Y previamente muestra también todas las palabras que han ocurrido en la partida como se muestra en el siguiente recuadro.

Una posibilidad en el juego es que si el jugador es HUMANO e inserta 0 (porque no obtiene una siguiente jugada), el juego puede ayudarle indicándole un conjunto de posibles siguientes palabras si hubiese. En el caso de que no haya siguientes palabras entonces el jugador HUMANO perdería al insertar 0.

```
machine>>

Las palabras dichas son:
mora    hora    hola

Escoge una palabra entre el siguiente conjunto:
acto    afan    agil    agua    aire    alto    alza    amar
amor    aqui    arar    arco    area    arte    auge    auto
azar    azul    bajo    base    beso    bien    boca    boda
bola    bota    bote    cabo    cada    caer    cafe    caja
cama    cara    casa    casi    caso    cero    cien    cima
cine    cita    club    cola    como    copa    cosa    cual
cuyo    dedo    diez    duda    duro    edad    ella    esas
esta    este    fase    fijo    flor    foto    frio    gato
golf    goma    gota    gran    gris    hace    hija    hijo
hoja    humo    idea    isla    joya    juez    lado    lago
lana    lata    lazo    leal    leer    leon    leve    liso
luna    malo    mano    mapa    masa    mayo    maza    mazo
mesa    mina    modo    mono    mozo    muro    nada    nave
nota    nube    nudo    nuez    ocho    odio    oido    oler
olor    once    otro    pago    pais    palo    papa    para
pase    paso    pelo    pera    peso    pico    piel    pila
piso    plan    poco    pozo    raiz    rama    raya    rayo
real    reir    reja    rico    riel    roca    rojo    ropa
rosa    roto    ruso    sano    seco    seda    seis    sexo
solo    sopa    suma    tajo    tasa    taza    tela    tele
tema    test    tipo    todo    tono    tras    tren    tres
unir    usar    util    vaca    vara    vaso    vela    vera
vida    vino    vivo
La ultima palabra fue hola


.       .       .

Rosa>>
Cual es tu palabra? 0

FIN PARTIDA_____________________________
Las palabras que han salido son:
mora    hora    hola    cola    cosa    casa    tasa    taza
maza    mapa    papa    para    vara    vera    vela    tela
tema
_____________________________________________

Enhorabuena jugador machine Total ganadas 1
Quieres jugar otra (s/n)?
```

Finalmente el programa preguntará si quiere volver a jugar. Si es así escogerá de forma aleatoria la longitud de las palabras y la palabra con la que se inicia la siguiente partida. Es recomendable que la longitud máxima no sea superior a 7 letras.

Si se decide no volver a jugar más el programa almacenará en el fichero de ranking los resultados obtenidos y mostrará la ordenación global de los jugadores. Tal como se muestra a la derecha

```
Resultado Final
Rosa puntos 0
machine puntos 1
Global Ranking  ..............
1.-Carmen       19
2.-machine1     4
3.-Tomas        3
4.-machine2     1
5.-Rosa 1
```

5. La heurística de la Máquina

La forma en la que elige la máquina su siguiente jugada puede ser poco elaborada, y por lo tanto será más rápida en tomar sus decisiones o más elaborada. Independiente de que sea más o menos elaborada para implementar la estrategia el alumno/a usará un ArbolGeneral. La raíz de este ArbolGeneral tendrá la palabra de inicio y como descendientes todas aquellas palabras que dista a esta primera en un carácter. En la siguiente imagen se muestra un ejemplo:

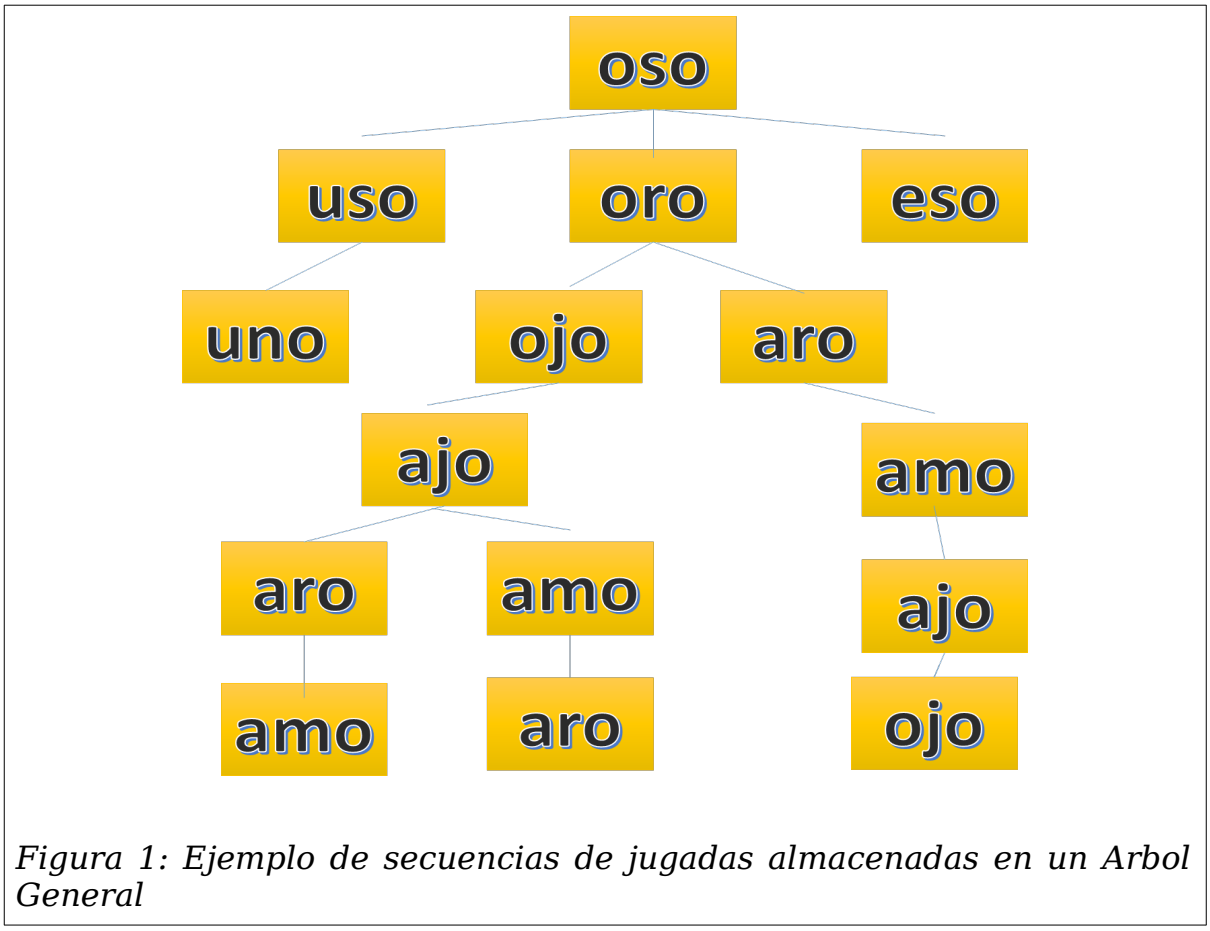


Figura 1: Ejemplo de secuencias de jugadas almacenadas en un Arbol General

Así en la imagen anterior teniendo como palabra de partida “oso” el jugador máquina construye el ArbolGeneral como se muestra. Así los hijos de “oso” son todas aquellas palabras de entre el conjunto de palabras que se diferencian en un sólo carácter. Así tenemos “uso”,

"oro" y "eso". A su vez "uso" tiene como único hijo "uno". Y "oro" por su parte tiene como hijos "ojo" y "aro". Cuando se inserta un hijo se asegura previamente que el camino que va desde la raíz a el nodo no haya aparecido repetida la palabra para poder insertarla esta como hijo. Esto quiere decir que "oro" no tiene como descendiente "oso" ya que "oso" es un ancestro de "oro". Como se puede observar existen palabras repetidas en el ArbolGeneral pero esas palabras no pueden aparecer repetidas en un mismo camino que vaya desde la raíz a un nodo.

En cuanto a la siguiente jugada mejor de la máquina, en nuestro ejemplo, si después de aparecer "oso" fuese el turno de la máquina, lo mejor para ella sería decir "eso" ya que el jugador contrario no tendría a continuación ninguna palabra alternativa (con el conjunto de palabras con el que estamos jugando). Por otra parte si fuese el turno del jugador humano y este dice "uso" lo mejor para la máquina es decir "uno" y ganaría. Pero esto requiere una lógica más inteligente. Por ello se propone al alumno/a que intente desarrollar algún tipo de lógica más compleja. La lógica más simple sería escoger por ejemplo el hijo más a la izquierda (o entre los hijos uno de forma aleatoria). Así en el ejemplo con esta lógica simple escogería "uso".

NOTA: El alumno/a tiene que implementar la lógica simple y una lógica más inteligente. Con estas dos estrategias el programa podría hacer competir los dos tipos de máquinas.

6. Tareas a realizar

Antes de abordar el programa descrito en la sección 4 el alumno/a deberá desarrollar diferentes tipos de datos y realizar los tests correspondientes. Así las tareas a realizar son:

1. Recuperar (o desarrollar completamente si no hizo la práctica 2 o 3) el T.D.As Palabras y Palabra.
2. Desarrollar y probar el ***T.D.A Jugador***
3. Desarrollar y probar el ***T.D.A Strategy***
4. Desarrollar y probar el ***T.D.A Juego***
5. Desarrollar y probar el ***T.D.A Ranking***
6. Generar un **módulo Utilidades** donde se almacena diferentes funciones asociadas a *string, set, list* y *ArbolesGenera*l que serán necesarias.
7. Construir el programa **WordDance** descrito en la sección 4

6.1. TDA Jugador

Un objeto de tipo **Jugador** almacena la información de un jugador del juego WordDance.

Una posible representación de Jugador podría ser la siguiente:

```
//FICHERO jugador.h
//tipo de jugador MAQUINA1:logica simple MAQUINA2:logica más compleja
enum TIPOJ {HUMANO, MAQUINA1,MAQUINA2,DESCONOCIDO};
class Jugador{
  private:
    TIPOJ tipo; //tipo de jugador
    string nombre;//nombre del jugador
    int winner_player; //partidas ganadas
    bool inicia_juego; //true si inicia el juego false en caso contrario
```

```
    …
};
//FIN DE jugador.h
```

6.2. TDA Strategy

Un objeto de tipo **Strategy** contiene las secuencias de siguientes posibles jugadas. Las secuencias de siguientes jugadas se almacenarán en un árbol general. Los nodos del árbol general almacena una palabra que difiere con la palabra que se almacena en el padre en una letra. A partir de esta información el objeto de tipo estrategia puede proponer la siguiente jugada al jugador.

Una posible representación podría ser la siguiente:

```
//FICHERO estrategia.h
class Strategy{
  private:
     ArbolGeneral<string>datos; //secuencias de posibles jugadas
     set<string> total_palabras;//palabras posibles para las secuencias jugadas
…
}
//FIN DE estrategia.h
```

La complejidad de este T.D.A reside en como a partir de la palabras posibles obtener la secuencia de siguientes jugadas (árbol del juego) almacenándolas en el Arbol General asociado (datos) y una vez definido este como elegir la mejor jugada. En la Figura 1 se muestra un ejemplo del árbol del juego.

El T.D.A Strategy debería al menos tener los siguiente métodos:

```
class Strategy{
…
   public:
     //Constructores
     Strategy();
     Strategy(const set<string> &todasw):total_palabras(todasw)

     //Inicia la secuencias de jugadas siguientes teniendo como raiz w
     bool CrearJugadas(const string & w);

     //limpia de datos la estrategia
     void clear();

     //Obtiene la siguiente jugada (palabra) usando la lógica implementada
     string getWord_1(string ultimaword);
     …
};
```

De entre estos métodos destacar CrearJugadas que inicia el ArbolGeneral con la secuencia de posibles jugadas partiendo de la palabra w (que será la raíz del árbol).

6.3. TDA Juego

Un objeto de tipo **Juego** orquesta el juego WordDance. Para ello mantiene las informaciones:

1) De los dos jugadores; 2) El conjunto de palabras; 3) Las palabras válidas para la partida; 4) La longitud de las palabras válidas; 5) Las palabras que han ocurrido en el juego hasta el momento; 6) La ultima palabra dicha; 7) El jugador que tiene el turno. Una posible representación de juego podría ser la siguiente:

```
//FICHERO juego.h
class Juego{
 private:
      Palabras mispalabras;//Todas las palabras
      int l_words;//Longitud de las palabras
      set<string> all_words;//Todas las palabras de longitud l_words sobre las que se puede crear
                            //secuencias al menos de longitud 3
      list<string> wsaid; //Las palabras que han ocurrido en el juego
      Jugador *j1,*j2;//Direcciones de los jugadores
      int turno; //0 jugador1  1 jugador2
      string ultimaword; //Ultima palabra dicha
….
};
//end juego.h
```

Cuando se inicia un objeto de tipo Juego se debe leer todas las palabras y de estas seleccionar solamente aquellas que tienen longitud ***l_words*** y además entre estas escoger aquellas que permita al menos realizar 3 jugadas. Esto quiere decir que descartamos aquellas palabras tal que a partir de una no encontramos 2 siguientes.

6.4. TDA Rankings

Un objeto de tipo **Ranking** mantiene la información de los jugadores de **WordDance** y las partidas ganadas. Con este objeto se permite una vez que los jugadores finalizan el juego modificar la información de las partidas ganadas y poder a través de este objeto de tipo Ranking obtener una ordenación global de los jugadores. Una posible representación del T.D.A Ranking es la que se da a continuación:

```
//FICHERO ranking.h
class Ranking{
 private:
     private:
        //nombre de jugador partidas ganadas totales
        map<string,int> datos;
….
  public:
};
//end ranking.h
```

6.5. Módulo Utilidades

El módulo utilidades agrupa un conjunto de funciones que permite ayudar al desarrollo de los anteriores T.D.As. Un ejemplo de las cabeceras que pueden aparecer en este módulo son las siguientes:

```
//FICHERO utilidades.h
//Obtiene el conjunto diferencia entre los conjuntos s1 y s2
```

```
set<string> WordDiference(const set<string> &s1,const set<string> &s2);
//Obtiene el número de letras en las que son diferentes las cadenas s1 y s2
int DistanceWords(const string &s1,const string &s2);
//Obtiene el conjunto de palabras de s que se diferencia de w en solo una letra
set <string> getWordDistanceOne(const set<string> &s,string w);
//Devuelve true si la palabra w esta en el camino de la raiz del arbol al nodo n
bool EstaEtiquetaCamino(ArbolGeneral<string> &a, ArbolGeneral<string>::Nodo n,const string & w);
//Crea el Arbol General a partir del nodo n de forma que la diferencia en letras de las palabras que
//almacena un nodo y su padre no puede ser mayor que 1. La variable lc permite no ramificar el arbol
//más all de lw. Es decir se llama con lc=0 y cuando lc alcanza lw por una rama se termina.
void CreaTree(ArbolGeneral<string> &a, ArbolGeneral<string>::Nodo n,const set<string> &pp,int lc,int
lt=numeric_limits<int>::max());
//Convierte una lista a un set
set<string> ConvertList_Set(const list<string> &s);
//Obtiene la longitud del mayor camino dentro del arbol
int MaxWay(ArbolGeneral<string> &a,ArbolGeneral<string>::Nodo n);
//Obtiene a partir de una palabra s y un conjunto de palabras todas las palabras que se diferencia
//en sola una letra. Y además el mayor camino que se obtendría si ramificamos el arbol de jugadas
//desde esa palabra.
pair<int,set<string>> PosiblesPalabrasDesde(const string &s,const set<string> &pp);
//end utilidades.h
```

Aquí podéis poner otras funciones que os haga falta para vuestro desarrollo.

7. Ficheros

7.1. Fichero con las Palabras

Para poder probar nuestro programa necsitamos un fichero que almacena las posibles palabras para jugar.

El fichero contiene:

1. La primera línea nos indica el idioma del conjunto de palabras
2. La siguiente linea es un comentario, que se inicia con el carácter '#'.
3. A continuación en cada línea viene la información de cada palabra. Cada atributo de un palabra se encuentra separada por ";". Los atributos son
   - Tipo de palabra
   - Valor de la palabra (puede ser una palabra simple o compuesta)
   - A continuación puede aparecer (no siempre) una lista de sinónimos separados por ';'

Un ejemplo de un trozo de estos ficheros es el siguiente:

```
Español
#TIPO;PALABRA;SINONIMOS(separadas por ;)
country;China;
country;Estados Unidos;
country;Italia;
country;Japon;
country;Reino Unido;
country;Reino de España;
country;Republica Checa;
country;Republica Federal de Alemania;
country;Republica Francesa;
country;Republica de la India;
preposition;a;
adverb;a lo largo de;junto con;
adverb;junto con;a lo largo de;
adverb;a traves de;dentro de;por;
adverb;dentro de;a traves de;por;
adverb;por;a traves de;dentro de;
adverb;a veces;
adverb;abajo;
adjective;abajo;debajo;
adjective;debajo;abajo;
verb;abandonar;dejar;
verb;dejar;abandonar;
animal;abeja;
adjective-basic;abierto;
verb-basic;abonar;pagar;
verb-basic;pagar;abonar;
time;abril;
verb;abrillantar;
verb-basic;abrir;
adjective;absoluto;completo;perfecto;total;
….
```

7.2. Fichero con el Ranking

El fichero con el Ranking contiene una serie de líneas. En cada línea se almacena el nombre del jugador y el número de partidas ganadas. Un ejemplo podría ser el siguiente:

```
Carmen;19
Rosa;1
Tomas;3
machine1;4
machine2;1
```

8. Esquema del Proyecto

En el siguiente esquema se ve los T.D.As que como mínimo debería tener el proyecto:

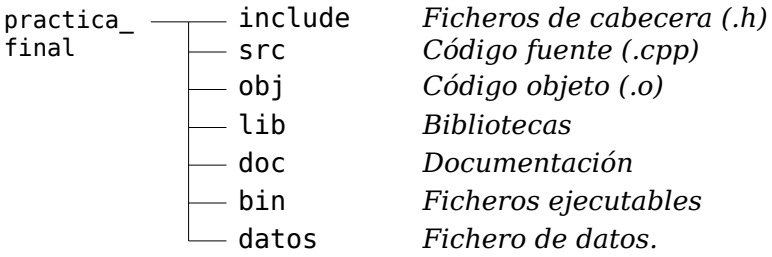
En este esquema se indica los T.D.As existentes en el proyecto y las dependencias de un T.D.A con otro. Por ejemplo el T.D.A ***Jugador*** depende de T.D.A ***Strategy***, ya que el jugador cuando es máquina deberá obtener su siguiente jugada siguiendo una estrategia. Por otro lado un objeto de tipo ***Juego*** necesita conocer la información de los jugadores, el conjunto total de palabras, y además el módulo utilidades que le proporciona funciones para la implementación de sus operaciones. A su vez el T.D.A ***Strategy*** usará el TDA ***ArbolGeneral*** para establecer el conjunto de siguientes jugadas. Nuestro programa ***WordDance*** dependerá de los T.D.As **Juego, Ranking y Jugador.**

Por otro lado los pasos fundamentales de nuestro programa se da en el main del fichero worddance.cpp del material

9. Práctica a entregar

El alumno deberá empaquetar todos los archivos relacionados en el proyecto en un archivo con nombre "practica_final.*tgz*" y entregarlo antes de la fecha que se publicará en la página web de la asignatura. Tenga en cuenta que no se incluirán ficheros objeto, ni ejecutables. Es recomendable que haga una "limpieza" para eliminar los archivos temporales o que se puedan generar a partir de los fuentes.

Los archivos deben estar distribuidos en directorios*:*



Para realizar la entrega, en primer lugar, realice la limpieza de archivos que no se incluirán en ella, y sitúese en la carpeta superior (en el mismo nivel de la carpeta "*practica_final*") para ejecutar:

```
prompt% tar zcv practica_final.tgz practica_final
```

tras lo cual, dispondrá de un nuevo archivo practica_final.tgz que contiene la carpeta practica_final así como todas las carpetas y archivos que cuelgan de ella.