Practica Final: Cifras y Letras

1. Módulos desarrollados

Se han desarrollado los siguientes módulos con la siguiente funcionalidad:

- **Letra y ConjuntoLetras**: el TDA Letra se usará para guardar la información asociada a una letra del juego, es decir, el carácter de la letra, el número de veces que aparece esa letra y los puntos que tendrá la letra en el juego. El TDA ConjuntoLetras este TDA permite cargar en memoria el archivo "ficheroLetras" o crear uno a partir de un diccionario
- **BolsaLetras**: guarda cada letra en una bolsa de la cual se irá sacando una cantidad de letras de forma aleatoria, con las cuáles jugará el usuario
- **IA**: permite que el jugador se enfrente con la máquina durante el juego. Su nivel de dificultad dependerá del tamaño del diccionario usado
- **Diccionario**: implementado en el TDA Diccionario, permite cargar un diccionario en memoria y trabajar con él. Internamente está implementado usando un set.
- **Diccionario2**: igual al anterior. Internamente está implementado un árbol.

El módulo IA y Diccionario, ambos se han implementado usando un árbol del mismo tipo y el mismo algoritmo de búsqueda e inserción de palabras, por lo que sus métodos son muy similares

2. Implementación

Todos los módulos, excepto los módulos IA y Diccionario2, se han implementado usando los contenedores de la STL de C++ y sus funciones asociadas.

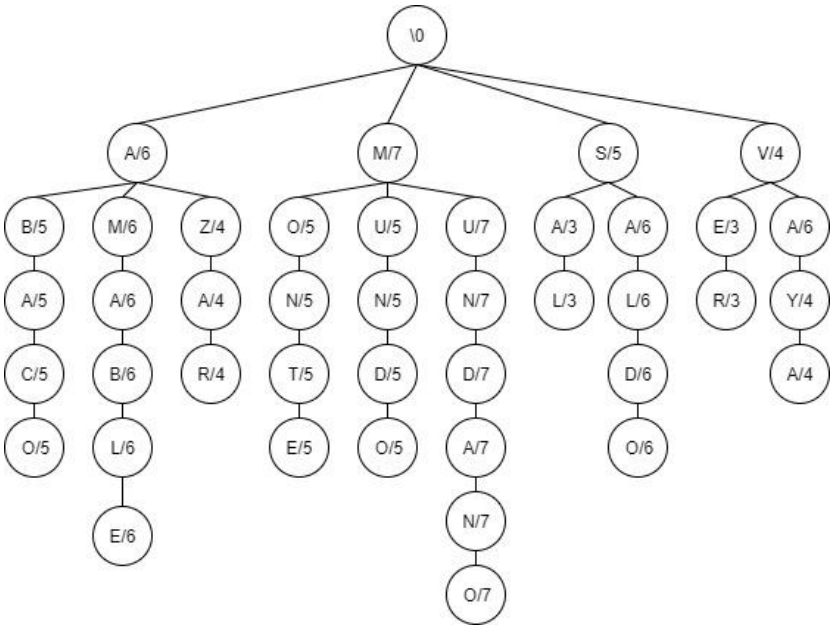
Para el módulo IA y Diccionario2 se ha usado el TDA ArbolGeneral proporcionado. Ambos, usan un árbol organizado de la siguiente forma:

- EL árbol solo tendrá **tres** niveles
- El nodo raíz no tiene valor asociado
- Cada nodo estará compuesto por dos valores: una letra y un entero
- El nivel 1, tendrá tantos nodos como letras del abecedario
- De cada nodo del nivel 1, colgarán varios hijos los cuales representan una palabra que empiece por la letra guardada en el nivel 1

Para cargar un diccionario en ambos casos, se aplica el siguiente procedimiento:

1- Se convierte la palabra a un árbol colgando cada letra como hijo a la izquierda de la letra anterior. El segundo campo de cada nodo (el entero mencionado anteriormente) guardará la longitud/puntos de la palabra según el modo de juego
2- Se inserta la palabra como hijo del nodo con la inicial de la palabra y se actualiza el valor del entero de su inicial en caso de que sea mayor

Tendríamos un árbol parecido al siguiente:



Para buscar un elemento en este tipo de árbol, no hace falta recorrer todo el árbol. Basta con buscar su inicial y explorar esa rama hasta encontrar la palabra buscada.

La IA, usará esta estructura para buscar las mejores soluciones dado un conjunto de letras aleatorio. Para ello hacemos lo siguiente:

1- Nos situamos en el nodo hijo más a la izquierda de la raíz (primera inicial guardada) y comprobamos si esa letra está en el conjunto de letras. Si está, exploramos esa rama en busca de una solución. Si no está, no saltamos a la siguiente inicial y repetimos este paso
2- Si la inicial está en nuestro conjunto de letras nos situamos en el hijo más a la izquierda y vamos explorando las soluciones.
3- Supongamos que hemos encontrado una solución de longitud 5. En el siguiente paso se pueden dar 3 casos según el valor del entero de ese nodo: que la siguiente palabra tenga una longitud menor, que tenga la misma longitud o que tenga mayor longitud.
   Si la palabra es de menor longitud, nos la saltamos pues esa solución no es válida.
   Si la palabra es de igual longitud exploramos esa palabra para ver si es una solución válida.
   Si la palabra es de mayor longitud comprobamos que es una solución válida. Si es una solución, esta será mejor que todas las que hayamos encontrado anteriormente, así que las borramos e insertamos la nueva solución

4- Una vez explorada una inicial y en caso de haber encontrado alguna solución, a la hora de explorar la siguiente inicial debemos hacer dos comprobaciones: que esa inicial está en nuestro conjunto de letras y que el entero de ese nodo no sea menor a la longitud de nuestras soluciones. Si tenemos que nuestras soluciones son de longitud 6, pero el entero de esa inicial es 5, esto quiere decir que la palabra más larga de esa inicial tendrá longitud 5 y, aunque haya una solución no será mejor a las que ya tenemos así que nos ahorramos de explorar ese nodo y pasamos al siguiente

3. Compilación y ejecución

Para compilar los ejecutables basta con ejecutar make.

Dentro de la carpeta bin se crearán 3 ejecutables:

- **cantidadLetras**: permite crear el "ficheroLetras".

  Para ejecutar: ./cantidadLetras archivo_diccionario letras_a_usar fichero_de_salida

- **letrasSET**: inicia el juego usando el diccionario implementado como un set

  Para ejecutar: ./letrasSET archivo_diccionario ficheroLetras n_letras_a_generar modo_juego(L/P)

- **letrasARB**: inicia el juego usando el diccionario implementado como un árbol

  Para ejecutar: ./letrasARB archivo_diccionario ficheroLetras n_letras_a_generar modo_juego(L/P)