Trabajo Práctico 3

Tipos Abstractos y Estructuras de Datos

Introducción a la Computación

1er cuatrimestre 2018

Observaciones generales:

- El trabajo se debe realizar en grupos de dos personas.
- El código fuente debe estar bien comentado.
- El informe y el código fuente deben enviarse por mail a la lista de docentes de la materia: `icm-doc@dc.uba.ar`.
- El programa entregado debe correr correctamente con el `Python 3.x` instalado en las computadoras del Laboratorio 4.
- Se evaluará la correctitud, claridad y modularidad del código y el informe entregado.
- La fecha límite de entrega es el **Domingo 1° de Julio a las 23:59**.

La empresa JUGATE CONMIGO S.A. ha decidido contratarlos para que formen parte del equipo de desarrollo de su ambicioso proyecto SCRABBLE ONLINE. Tal como su nombre lo indica se trata de una versión para Internet del famoso juego de palabras[1].

Cada uno de los jugadores de una sesión de SCRABBLE verá en la pantalla de su computadora (o dispositivo móvil) la imagen del tablero y de las fichas que posee para formar palabras. Una vez que un jugador coloca una palabra sobre el tablero el juego debe (entre otras cosas) chequear que la palabra sea válida. Además, como ayuda para los jugadores principiantes de la plataforma, se desea poder informar, dado el conjunto de letras en posesión de un jugador, el conjunto de las palabras conocidas que se pueden formar usando todas esas letras. El trabajo que se les encomienda es relativo a estas necesidades.

El juego debe poder jugarse en distintos idiomas. Por ahora ya está garantizada su comercialización en Francia, Argentina, Brasil y Alemania. De todas formas, como la intención es poder venderlo también en otros mercados, se espera que el software diseñado tenga buenos tiempos de respuesta independientemente del idioma en el que se esté jugando. Más precisamente, los tiempos de corrida deben ser independientes de la cantidad de palabras que posea el idioma seleccionado. Ésto es muy importante: el tiempo que le demande al programa comprobar si una palabra es válida será una gran parte del tiempo que los jugadores estarán esperando frente a su pantalla hasta que el juego pueda proseguir. También interesa que sea rápida y eficiente la consulta de posibles palabras armables para no desalentar a los principiantes con la espera.

Teniendo en cuenta estos requisitos, se les requiere implementar el tipo de dato que se describe a continuación respetando los órdenes de complejidad temporal indicados:

[1] Para más información sobre el Scrabble visitar `http://es.wikipedia.org/wiki/Scrabble`

TAD AYUDANTE

- *NuevoAyudante(N)* $\rightarrow$ Ayudante: Devuelve un nuevo ayudante vacío en $O(1)$.
- *A.CantidadDePalabrasConocidas()* $\rightarrow \mathbb{Z}$: Devuelve la cantidad de palabras conocidas por el ayudante *A* en $O(1)$.
- *A.AgregarPalabra(P)*: Agrega la palabra *P* al ayudante *A* en $O(|P| \log |P|)$. Si *P* ya existe en *A* la operación no tiene efecto sobre *A*.
- *A.PalabrasPosibles(P)* $\rightarrow [String]$: Devuelve una copia del conjunto de palabras conocidas posibles de agregar con las letras de la palabra *P* con complejidad $O(|P| \log |P| + |\{p : Palabra \mid A.Existe(P) \wedge p \in anagramas(P)\}|)$.
- *A.Existe(P)* $\rightarrow \mathbb{B}$: Devuelve TRUE si la palabra *P* es conocida por el ayudante *A* y FALSE en caso contrario en $O(|P|)$.

Se pueden hacer las siguientes suposiciones:

- las palabras sólo se ingresarán en minúscula;
- ninguna palabra tendrá tildes, eñes, ni diéresis (“ü”).

De ser necesario, se pueden definir TADs auxiliares. La entrega del trabajo práctico debe incluir:

1. un informe que contenga:
   a) las estructuras de representación propuestas para los TADs que se hayan utilizado;
   b) los invariantes de representación de las estructuras propuestas (en español);
   c) los algoritmos en pseudocódigo, justificando en cada caso que cumplen con los órdenes requeridos;
2. la implementación del TAD Ayudante en Python.

El TAD Ayudante debe implementarse en una clase llamada `Ayudante`. Su código fuente debe estar en un archivo llamado `TIPO_AYUDANTE.py` y respetar la siguiente signatura:

```
class Ayudante:
  def __init__(self)
  def cantidadDePalabras(self)
  def agregarPalabra(self, p) # p:string, correspondiente a una palabra
  def palabrasJugables(self, p) # p:string, conteniendo letras a usar
  def existe(self, p) # p:string, correspondiente a una palabra
```

A continuación, damos un ejemplo de uso y ejecución del TAD Ayudante:

```
dicc = ["hola", "halo", "ola", "loa", "orla"]
a = Ayudante()
for w in dicc:
  a.agregarPalabra(w)
print(a.Existe("hola"))
print(a.Existe("loha"))
s = a.PalabrasPosibles("laoh")
for w in s:
  print(w)
```

Salida del script anterior:

```
True
False
hola
halo
```