Trabajo Práctico Especial
Estructuras de Datos y Algoritmos
Segundo Cuatrimestre 2015

Objetivo

El objetivo del trabajo práctico es implementar un algoritmo que resuelva tableros de una variante del juego *Scrabble*.

Se tiene una grilla en la cual se deben ubicar letras para formar palabras, tanto en sentido horizontal como vertical. Se cuenta con una lista de letras que se pueden utilizar, y el objetivo es ir formando palabras (de a una por vez), hasta que ya no se pueda formar ninguna otra. Cada letra tiene un valor asignado, una vez terminado un tablero se suman los valores de las letras que fueron utilizadas y se obtiene el puntaje final.

Se pide implementar un programa en Java que dado un conjunto de letras y un diccionario de posibles palabras, encuentre la distribución de las mismas en el tablero que maximiza el puntaje total obtenido.

Requerimientos

Descripción del problema

Se tiene inicialmente una grilla de 15 x 15 vacía. Y se cuenta además con una lista de letras disponibles. En cada movimiento se debe formar una palabra nueva en la grilla. Las palabras deben tener como mínimo 2 letras y como máximo 7.

Para el primer movimiento, se puede formar cualquier palabra con las letras disponibles, y ubicarla de manera tal que alguna de las letras de la misma ocupe el casillero central.

Luego, en los próximos movimientos, se debe utilizar por lo menos una de las letras ya existentes en el tablero para formar la nueva palabra (completando las letras restantes con letras que aún no hayan sido utilizadas). En el tablero solamente se pueden colocar palabras completas (utilizando las letras que aún se encuentren disponibles). Se pueden repetir palabras, siempre que haya letras disponibles que lo permitan. Si cualquiera de las letras que conforman la palabra a agregar se encuentra adyacente a alguna otra letra ya existente en el tablero, se debe verificar que se esté formando una palabra válida. En caso contrario no es un movimiento posible.

Cada letra tiene un puntaje asociado, según la siguiente tabla:

Letras	**Valor**
A, E, I, L, N, O, R, S, T, U	1
D, G	2
B, C, M, P	3
F, H, V, W, Y	4
K	5
J, X	8
Q, Z	10

Al finalizar un tablero, se calcula el puntaje sumando los valores de las letras que hayan sido utilizadas en el tablero.

El problema a resolver consiste en encontrar una secuencia de jugadas que genere el tablero de valor máximo, para un listado inicial dado de letras.

A continuación se muestra un ejemplo, en un tablero reducido de 7 x 7.

Se cuenta con un diccionario que contiene únicamente las siguientes palabras:
auto, casa, tabla, pelo, libro, monitor, queso

Se tiene inicialmente el siguiente listado de letras:
A, C, E, L, Q, A, A, B, E, O, P, S, T, U.

Con estas letras, una posible palabra a formar es "CASA". Al ser la primera, alguna de las 4 letras que la conforman debe ocupar el centro del tablero. Una posible ubicación es la siguiente:

	C	A	S	A		

Letras disponibles: E, L, Q, A, B, E, O, P, T, U

La próxima palabra a colocar en el tablero debe utilizar sí o sí alguna de las 4 letras ya ubicadas en el tablero. Una posible jugada es entonces:

		T				
	C	A	S	A		
		B				
		L				
		A				

Letras disponibles: E, Q, E, O, P, U

Con las letras restantes, otra jugada posible puede ser:

		T				
	C	A	S	A		
		B				
P	E	L	O			
		A				

Letras disponibles: E, Q, U

Con las letras restantes ya no se puede formar ninguna otra palabra. Para evaluar este tablero, se suman los puntajes de las letras que fueron utilizadas, obteniendo 17.

Esta es una solución, pero no necesariamente es la mejor. Para el ejemplo anterior, el mayor puntaje posible es 19.

Uno de los posibles tableros con este puntaje es el siguiente:

			Q			
			U			
			E			
	C	A	S	A		
			O			

Letras disponibles: L, A, B, E, P, T

Notar que si bien en este caso se utilizaron menos letras y se formaron menos palabras que en el anterior, se obtuvo un puntaje mayor.

Implementación

Se debe implementar una aplicación en Java que resuelva este problema. Debe ser capaz de obtener la solución exacta (en el tiempo que sea necesario), o bien la mejor solución encontrada hasta el momento (proporcionando un límite de tiempo). Además se debe poder opcionalmente mostrar de manera visual los estados explorados por la aplicación durante la ejecución del algoritmo.

La aplicación deberá leer de un archivo de entrada el diccionario de posibles palabras, y de otro archivo las letras que tiene disponibles para resolver el problema. Una vez hallada la respuesta, debe escribirla en un archivo de salida.

No es necesario validar el correcto formato de los archivos de entrada.

Formato del archivo de letras

El archivo de letras es un archivo de texto de una sola línea, en donde se enumeran las letras disponibles para el algoritmo. Como máximo puede contener 80 letras.

Para el ejemplo anterior, podría ser:

letras.txt

```
ACELQAABEOPSTU
```

Las letras pueden aparecer tanto en mayúscula como en minúscula, y se las debe tratar de manera indistinta. Solamente se consideran caracteres ASCII estándar entre A y Z (no considerar letra Ñ ni acentos).

Formato del archivo de diccionario

El archivo de diccionario es un archivo de texto, en donde cada línea contiene una palabra.

Para el caso del ejemplo anterior, un posible diccionario es:

diccionario.txt

```
auto
casa
tabla
pelo
libro
monitor
queso
```

El algoritmo debe ignorar el hecho de que haya palabras en mayúscula o minúscula. Si bien pueden aparecer palabras con caracteres ASCII no estándar (acentos, letra "ñ", etc.), se las puede ignorar ya que nunca se podrán formar estas palabras (al no tener estos caracteres disponibles en el listado de letras). En estos casos, no considerar el diccionario como un archivo inválido, sino ignorar estas palabras.

Formato del archivo de salida

El archivo de salida es un archivo de texto, que contiene una matriz de 15 x 15 caracteres, con el estado final del tablero. Para los casilleros vacíos, completar con un espacio en blanco. Sino, con la letra que se encuentra en dicha ubicación.

Para el ejemplo anterior, la salida esperada es:

`salida.txt`

```
   Q
   U
   E
 CASA
   O
```

Sintaxis de la línea de comandos

Para ejecutar el programa, se debe respetar la siguiente sintaxis en la línea de comandos (los corchetes indican parámetros optativos):

java -jar tpe.jar diccionario letras salida [-visual] [-maxtime n]

Parámetro	Descripción
`diccionario`	Nombre del archivo de diccionario a utilizar.
`letras`	Nombre del archivo de letras a utilizar.
`salida`	Nombre del archivo de salida que se quiere generar con la respuesta.
`-visual`	Ejecuta el programa en modo visual, mostrando una ventana en donde se ven los estados que están siendo explorados por el algoritmo. Al encontrar la solución (o bien terminarse el tiempo), se debe mostrar la solución encontrada en pantalla (además de almacenarla en el archivo indicado).
`-maxtime n`	Indica el límite de tiempo en segundos para la ejecución del algoritmo. Pasado este tiempo, el programa debe terminar, generando en la salida la mejor respuesta obtenida hasta el momento. (Si el programa encuentra la solución exacta en menos tiempo, retorna antes).

Ejemplos de uso:

```
# java -jar tpe.jar diccionario.txt letras.txt salida.txt
```

Ejecuta la aplicación leyendo el diccionario de diccionario.txt, las letras a utilizar de letras.txt. Encuentra la mejor solución (tardando el tiempo que sea necesario), y la escribe en el archivo salida.txt.

```
# java -jar tpe.jar dict.txt letras tablero.txt –maxtime 60
```

Ejecuta la aplicación leyendo el diccionario del archivo dict.txt, las letras a utilizar del archivo letras. Busca la solución durante 60 segundos, y pasado dicho tiempo escribe en el archivo tablero.txt la mejor solución obtenida en ese tiempo.

```
# java -jar tpe.jar dict.txt letras.txt tablero.txt -visual
```

Ejecuta la aplicación leyendo el diccionario de diccionario.txt, las letras a utilizar de letras.txt. Encuentra la mejor solución (tardando el tiempo que sea necesario), y mostrando en pantalla el estado del tablero en cada paso de la ejecución del algoritmo. Finalmente muestra en pantalla la mejor solución y además la escribe en el archivo tablero.txt.

Entrega

El trabajo se realizará en grupos de hasta 2 integrantes. La entrega del código fuente se deberá realizar a través del repositorio SVN provisto por la cátedra, antes del **martes 27 de octubre las 18 hs**. Cada grupo deberá enviar un mail a la cátedra indicando el número de revisión a considerar. Si el trabajo no estuviera aprobado se cuenta con una instancia de recuperación.

Existe la posibilidad de entregar en fecha tardía, el **martes 3 de noviembre**. En este caso se aplicará una penalización de 2 (dos) puntos en la nota final del trabajo y no se contará con una instancia de recuperación.

El código entregado debe contener un *buildfile* de Ant, cuyo target default genere un archivo **jar** con la aplicación desarrollada.

Al comienzo de la clase de laboratorio del **martes 27 de octubre**, cada grupo deberá presentar en forma impresa un informe que explique en forma clara:

- Algoritmo utilizado
- Estructuras de datos utilizadas
- Problemas encontrados durante el desarrollo y decisiones tomadas
- Tablas de comparación de tiempos
- Conclusiones

Además el **martes 3 de noviembre** se debe presentar ante la cátedra el TP, explicando cual es su funcionamiento general, cómo se abordaron los problemas, arquitectura de las clases, resultados obtenidos y problemas encontrados en el desarrollo.