Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Ciencias de la Computación
R-212 — Estructuras de Datos y Algoritmos I

Trabajo Práctico Final

Analizador sintáctico

1. Motivación del problema

Es común que muchas aplicaciones como editores de texto o motores de búsqueda interpreten una frase correctamente a pesar de que a esta le faltan espacios. Si se escribe *analizadorsintáctico* en la pestaña de búsqueda de Wikipedia[1] es posible que el texto se transforme a *analizador sintáctico* antes de efectuarse la búsqueda propiamente dicha en el sistema de bases de datos.

La pieza de software que introduce espacios en un texto de manera que este resulte legible la llamamos **analizador sintáctico** o **parser** y funciona de manera relativamente sencilla: dada una cadena de caracteres encuentra el conjunto de espacios y el conjunto de posiciones de dichos espacios que verifica que la frase este compuesta de todas palabras válidas. Consideramos palabras válidas a aquellas que se encuentran en un listado de palabras válidas, el cual llamaremos *diccionario*.[2]

Veamos un ejemplo. Supongamos que en nuestro diccionario están las siguientes palabras:

- deposito,
- dolar,
- dolares,
- es,
- quien,
- recibira

y que recibimos de entrada la frase `quiendepositodolaresrecibiradolares`. Dado que `quien`, `deposito`, `dolares` y `recibira` están en el diccionario, un parseo válido para la frase es `quien deposito dolares recibira dolares`. Notar que otro parseo válido de la frase es `quien deposito dolar es recibira dolares`.

En esta versión del analizador sintáctico nos interesará quedarnos con la **separación que maximice en cada paso el tamaño de las palabras**. En el ejemplo, de las opciones `dolares` o `dolar es` nos quedamos con `dolares` ya que es una palabra más larga que `dolar`.

Los analizadores sintácticos también suelen incluir la funcionalidad de **recuperarse luego de encontrar errores**. Es decir, si a partir de un punto no es posible formar ninguna palabra válida en el texto, se saltean 1 o más caracteres (considerados errores) para luego continuar el análisis. En ejemplo anterior, si la frase a analizar es `quienwdepositodolareszxrecibiradolares` entonces el resultado del análisis es `quien deposito dolares recibira dolares` y se saltearon los caracteres `w`, `z` y `x`.

La tarea del analizador sintáctico no necesariaente se centra en buscar palabras en un diccionario grande. De cualquier manera, una búsqueda eficiente es crítica para la performance del mismo, dado que parsear una frase implica buscar cada prefijo de la entrada.

[1] si, la mismisima enciclopedia libre

[2] en realidad este es un caso muy particular de parser, en general los parser identifican la estructura completa de un texto.

2. Objetivos

El programa que se tendrá que realizar, será un analizador sintáctico rudimentario: éste leerá un archivo de diccionario (provisto por la Cátedra), tomará un archivo de texto de entrada y deberá entregar frases espaciadas de manera que se maximice localmente el tamaño de las palabras indicando en cada caso si fue necesario recuperarse de errores.

3. Detalles de la implementación

La versión rudimentaria del analizador sintáctico que se pide no incluye ningún tipo de predicciones a futuro. Es decir, no debe considerar tomar un palabra de tamaño no maximal para en un futuro encontrar una palabra más larga (que se encuentre en el diccionario).

Si el texto de entrada es `quiendepositodolaresrecibiradolares` y el diccionario

- deposito,
- do,
- dolar,
- dolares,
- es,
- laresrecibira,
- quien,
- recibira

al momento de analizar si considerar la palabra `dolares` o `do` como siguiente palabra, se deberá elegir `dolares` ya que la cantidad de letras en `dolares` supera en 5 a la cantidad de letras en `do`.

El método main deberá permitir pasar el nombre del archivo de entrada y el nombre del archivo de salida.

3.1. Formato de archivo de entrada

El archivo de entrada tiene k lineas de texto que representan las frases a espaciar. Considere que los caracteres de las frases podrían ser letras mayúsculas, minúsculas, o una mezcla de ambas.

3.2. Formato de archivo de salida

El archivo de salida deberá tener k lineas, donde la k-ésima linea representa la k-ésima frase de entrada espaciada y un mensaje que indica si fue necesario recuperarse de errores.

4. Evaluación

Para la evaluación del Trabajo Práctico se tomarán en cuenta los siguientes elementos:

a) Estructuras de datos usadas.

b) Algoritmos propuestos.

c) Eficiencia.

d) Calidad del código entregado.

La entrega del trabajo también deberá incluir una descripción general de la solución propuesta en donde se expliquen las decisiones más importantes tomadas.