Reporte #1 - Análisis de herramientas

Delivery Date	@September 12, 2021
Equipo	Individual
Materia	Implementación de métodos computacionales

Carla Oñate Gardella A01653555

Analiza la solución que realizaste para el **Resaltador de sintaxis v1.0** en cuanto a su **eficiencia** haciendo intuitivamente un cálculo de la **complejidad computacional de tiempo** de tu programa (***Big O***) y contrastándolo con el tiempo real que se observa al ejecutar tu programa.

> Haciendo mis cálculos el resultado total de complejidad de mi programa es O(n^2). Esta complejidad es muy mala y no se ve lento el programa debido a la poca cantidad de texto que usé en las pruebas. Sin embargo si fuera un archivo de texto muy largo mi programa se tardaría mucho tiempo en correr. La parte que hizo que el programa fuera más lento y que generó que tuviera una complejidad tan alta es en la función `identifierToHtml()`. En esta función se escribió la lógica para poder diferenciar entre un identificador y una palabra reservada. Esta parte de mi programa creo es la más difícil de entender porque tuve que resolver algunos problemas que surgieron. Aquí no pude usar un `regex_replace()` porque no todas las palabras que identifica la regex tienen el mismo tag de HTML. Es por esto que tuve que ciclar la lista de coincidencias en la línea para ir cambiando una por una y evitar errores. Esto generó que la complejidad no fuera la deseada.
>
> El tiempo que toma mi programa en ejecutar es de: 35135 microsegundos. Este tiempo fue para un archivo de unas 10 líneas aproximadamente. Por lo que se toma unos 35 milisegundos en correr el archivo. Esto fue la primera vez que se corrío el archivo, la segunda vez el tiempo bajo a 20890 microsegundos. Haciendo un promedio te queda que el programa corre en 28012 microsegundos. Es difícil comparar si el programa corre rápido viendo este valor del tiempo. Esto porque para mí como persona 0.028 segundos es imperceptible pero la máquina no es el caso.

Por lo que en general yo concluyo que mi programa no es eficiente y debería de ser usado para archivos muy largos ya que tardará mucho tiempo en correr. Esta conclusión la obtengo con el valor de BigO que se obtuvo del programa.

Investiga sobre algunas **herramientas alternativas** a las que usaste para resolver la situación problema (construcción del resaltador de sintaxis). Identifica sus ventajas y desventajas. Identifica al menos un par de herramientas alternativas.

He encontrado que existen diferentes técnicas para resolver el problema de la actividad. Algunos de estos los vimos en clase.

1. Notación de regex

   Encontré que existen diferentes maneras de escribir las expresiones regulares. En mi caso estaba usando la de ECMAScript, pero en mi compilador algunos símbolos no eran aceptados. Sin embargo tenían maneras de poder poner condicionales en la expresión o para revisar los carácteres antes de la expresión y no sólo después. Varios proyectos en internet implementan estas soluciones usando estos tipos de símbolos. Esto tiene como ventaja que puedes ahorrarte líneas de código al tener más flexibilidad en la expresión regular.

   Ventajas

   - Facíl implementación
   - herramientas ya existentes
   - Buena documentación
   - Se encuentran de manera fácil los patrones
   - Funcionan para muchas situaciones pero de manera básica

   Desventajas

   - Requiere práctica para encontrar la regex correcta - desventaja especifica de mi caso
   - Sirve para resaltadores pequeños
   - Hace que el código sea dificil de leer
   - No es recomendable usarla para todos los problemas

- No puedes usarlo para lenguajes no regulares - más restricciones de uso

2. Automata Finito

   Esta actividad al igual se pudo haber resuelto como se hizo en la actividad 3 con un autómata finito. Solo que la desventaja de esto es que no cuentan con herramientas como la de `regex_replace` que te ayuda a hacer el cambio del texto original al HTML. No veo ventajas al usar esta herramienta ya que usar `regex` funciona de una manera más sencilla.

Establece **criterios** para comparar las herramientas que usaste con las otras alternativas (ej. eficiencia, facilidad de implementación, etc.) y plasma la comparación en una **tabla comparativa**.

> Mi solución para este problema fue usar regex pero con una notación más vieja. Es por esto que algunas ventajas y desventajas son iguales que usando regex con una notación más actual. La diferencia entre mi solución y con el tipo de regex que encontré en línea, es que se tienen menos funcionalidades en la expresión regular. Esto porque existen maneras de agregar condicionales pero en mi caso no servían y mandaba un error. Al ver esta tabla se ve que todas las opciones son similares. Esto porque tanto Regex como los autómatas solo definen lenguajes regulares, esto significa que se tienen más restricciones para el uso de estas soluciones.

Regex(Actual)

Ventajas

- Fácil implementación
- Herramientas ya existentes
- Buena documentación
- Se encuentran de manera fácil los patrones
- Funcionan para muchas situaciones pero de manera básica

Desventajas

- Requiere práctica para encontrar la regex correcta - desventaja

Automata Finito

Ventajas

- Tiene una representación visual para entender mejor como funciona.
- Fácil diseño.
- Fácil implementación.
- Sirve para una variedad de problemas de dimensión pequeña.
- Buena herramienta para reconocer patrones básicos.

Desventajas

especifica de mi caso

- Sirve para resaltadores pequeños
- Hace que el código sea dificil de leer
- No es recomendable usarla para todos los problemas
- No puedes usarlo para lenguajes no regulares - más restricciones de uso

- Difícil implementación para problemas grandes, como un resaltador de un lenguaje tipo C++. Son demasiadas opciones y hace complejo el proceso.
- Solo reconoce lenguajes regulares
- No tiene tanto poder como otras herramientas
- Con esto lees el archivo y reconoces pero no te ayuda a reescribir.
- Tienes que generar la lógica para reemplarzar el texto, algo que ya tiene `regex_replace`.

Investiga también la **frecuencia** con la que aparecen **nuevas tecnologías** para atender las necesidades que satisface un resaltador de sintaxis y trata de explicar las razones de dicha frecuencia.

> Existen muchos analizadores de sintaxis actualmente, muchos de estos son para aplicaciones web pero sirven para resaltar muchos tipos de lenguajes. Hay analizadores que fueron creados hace unos años pero han sido actualizados mientras que otros se crean con nuevas tecnologías. Esta frecuencia se debe a la disponibilidad de nuevas tecnologías que hacen que resolver este problema sea más eficiente. El mundo de las tecnologías computacionales avanza tan rápido que los programas existentes o soluciones tienen que irse adaptando. Esto es lo que genera que salgan nuevos programas de manera seguida a pesar de que existen ya soluciones a esa problemática.

Reflexiona sobre la **importancia de las herramientas formales de las ciencias computacionales** que has aprendido hasta el momento y el rol que tienen en la aparición de nuevas tecnologías que atienden necesidades que satisfacen los resaltadores de sintaxis.

Todo este tipo de herramientas que hemos visto en clase y en las clases pasadas son la que generan nuevas soluciones a nuevos problemas. Con esto quiero decir, que este tipo de herramientas, como las gramáticas, regex, entre otros, dan solución a problemas de maneras más eficientes. Al ser más eficientes se pueden generar mejores resultados. Y siempre se pueden ir mejorando para llegar a una solución incluso más eficiente. Estas herramientas son importantes porque nos ayudan a llevar algo teórico a la práctica y de una manera fácil y rápida. Por lo que es más fácil intentar hacer un resaltador de sintaxis usando una expresión regular, un autómata o una gramática, a generar el código de cero para implementar la solución.

Reflexiona sobre las posibles **implicaciones éticas** involucradas en la aplicación de las herramientas que has analizado en los puntos anteriores. En caso de que no identifiques alguna, trata de justificarlo.

El uso de cada herramienta viene con sus ventajas y desventajas, dependiendo de estas se tiene una mejor solución al problema. Pero esta es una manera muy simple de ver las cosas, porque cada programador define su solución por lo que no se tiene una misma solución para todo. Con esto quiero decir, que en cada línea de código el programador inconscientemente programa los prejuicios que tiene. Este es un problema actual del cual se está generando consciencia. Nosotros tenemos que estar conscientes de las afectaciones que pueden tener nuestras soluciones. Un ejemplo es si tu programa que pilotea un avión lo hiciste usando una tecnología A que no es la mejor pero funcionaba en vez de hacerlo con la tecnología B que es más confiable. ¿Qué consecuencias podría tener este programa por haber usado A en vez de B? Es por esto que siempre se descubren mejores maneras y herramientas formales para generar estas soluciones. Aquí es donde puede afectar qué tipo de herramienta formal se usa y las implicaciones que pueda tener.

Conclusiones

En general con esta actividad se pudo analizar la diferencia entre las herramientas que hemos utilizado como regex y automatas finitos para generar soluciones aplicables al mundo real. Se puede ver que cada herramienta es diferente aunque las dos te ayuden a llegar a la respuesta. S tiene que analizar ciudadosamente que herramienta usar para poder generar software de valor, que sea eficiente y confiable. Como se analizó en el

primer punto mi solución tiene una eficiencia muy baja por lo que no sería recomendable usarlo en una situación real. Sin embargo fue un buen ejercicio para poder usar regex y entender como esta herramienta se implementa. Al igual se llega a la conclusión que estas herramientas que estamos aprendiendo son la base teorica para poder generar programas eficientes. Sin estas herramientas formales sería complicado tener software tan complejo y a la vez eficiente.

Bibliografía

E. (2020a). GitHub - Eriice/code-prettify: Automatically exported from code.google.com/p/google-code-prettify. GitHub. https://github.com/Eriice/code-prettify

Gallardo, A. (2021, August 20). Solución: Los marcadores de sintaxis pueden funcionar de dos formas muy generales. Leer Más. Foro Ayuda. https://foroayuda.es/escribir-un-resaltador-de-sintaxis/

Compiler Design - Types of Parsing. (2020). Tutorialspoint. https://www.tutorialspoint.com/compiler_design/compiler_design_types_of_parsing.htm

Smith, J. (2021, September 3). Syntax Analysis: Compiler Top Down & Bottom Up Parsing Types. Guru99. https://www.guru99.com/syntax-analysis-parsing-types.html

G. (2013, June 12). Finite State Machines and Regular Expressions. GameDev.Net. https://gamedev.net/tutorials/_/technical/general-programming/finite-state-machines-and-regular-expressions-r3176/

Bracey, K. (2015, May 11). 25 Syntax Highlighters: Tried and Tested. Web Design Envato Tuts+. https://webdesign.tutsplus.com/articles/25-syntax-highlighters-tried-and-tested--cms-23931

Prism. (2020). Prism. https://prismjs.com/

What is syntax highlighting and how does it work? (2013, June 12). Meta Stack Exchange. https://meta.stackexchange.com/questions/184108/what-is-syntax-highlighting-and-how-does-it-work

Ainsley, C. (2018, May 17). Introducing Iro — An Easier Way To Create Syntax Highlighters. Medium. https://medium.com/@model_train/creating-universal-syntax-highlighters-with-iro-549501698fd2

Kovačević, A. (2020, March 20). What It Takes To Be an Ethical Programming Professional. Simple Programmer. https://simpleprogrammer.com/ethical-programming-professional/