Proyecto

Teoría de la computación

11 de mayo de 2020

Este es un proyecto grupal para el curso de Teoría de la Computación. El grupo es de máximo tres personas. Cada grupo escogerá un tipo de proyecto que desee desarrollar. Cada tipo de proyecto corresponde a un problema. El grupo deberá encontrar la literatura específica del problema (estado del arte) y proponer desarrollar algo en relación al problema.

En la mayoría de casos, corresponde a implementaciones eficientes para algoritmos ya estudiados para el problema, o para casos específicos del problema. Un avance más profundo también es válido, como por ejemplo proponer un nuevo algoritmo o un nuevo teorema, sin embargo no es necesario llegar a ese nivel.

Para evitar que muchos grupos escojan el mismo tipo de proyecto, cada grupo conformado deberá enviar los nombres de los integrantes y el tipo de proyecto escogido al correo del profesor: jgutierreza@utec.edu.pe. Se le dará prioridad a los grupos que escojan primero. A continuación definimos los tipos de proyecto que se pueden escoger.

1. Separamiento de palabras

Problema: Dadas dos cadenas w_1 y w_2 sobre el alfabeto Σ, encontrar un AFD A tal que A acepta w_1, A no acepta w_2 y A tiene el menor número de estados posibles.

Referencias

[1] Erik D. Demaine and Sarah Eisenstat and Jeffrey Shallit and David A. Wilson *Remarks on separating words*. http://arxiv.org/abs/1103.4513, 2011.

[2] J.M. Robson. *Separating strings with small automata*. Information Processing Letters, 30(4):209–214, 1989.

[3] Goralčík, P. and Koubek, V. *On discerning words by automata*. Information Processing Letters, 30(4):209–214, 1989.

[4] Wiki site: `https://en.wikipedia.org/wiki/Separating_words_problem`

2. Repeticiones en cadenas

Una cadena es una *repeticion* si es la concatenacion de dos o más cadenas iguales. Por ejemplo, *abab* es una repetición.

Problema: Dada una cadena, encontrar el número de subcadenas que son repeticiones.

Referencias

[1] David Eppstein *An optimal algorithm for computing the repetitions in a word*. Information Processing Letters, 12(5):244 - 250, 1981.

3. Minimización de AFD's

Problema: Dado un AFD, encontrar un AFD equivalente con el menor número de estados.

Referencias

[1] Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. *Introduction to Automata Theory, Languages, and Computation* . (2nd ed.), Addison-Wesley, 2001.

[2] Wiki site: `https://en.wikipedia.org/wiki/DFA_minimization`

4. Palabra sincronizadora

Dado un AFD A, una cadena w sobre el alfabeto de A es *sincronizadora* si es que existe un estado q de A tal que, todo estado p al recibir la cadena w termina en el estado q.

Problema: Encontrar el menor tamaño de una cadena sincronizadora en un AFD.

Referencias

[1] David Eppstein *Reset sequences for monotonic automata*. SIAM J. Computing, 19(3):500–510, 1990.

[2] A. N. Trahtman *The road coloring problem*. http://arxiv.org/abs/0709.0099

[3] Wiki site: `https://en.wikipedia.org/wiki/Synchronizing_word`

5. Altura de estrellas

Dada una expresión regular R, la altura de estrellas de R es el máximo número de estrellas anidadas. Por ejemplo, 1^* tiene altura 1 y $(1^* + 2^*)^*$ tendrá altura 2.

Problema: Dado un número n, existe un lenguaje regular con altura de estrellas igual a n?

Problema: Dado un lenguaje regular, cual es su altura de estrellas?

Referencias

[1] Eggan, L. C. *Transition graphs and the star-height of regular events.* Michigan Math. J., 12(4):385–397, 1963.

[2] Wiki site: `https://en.wikipedia.org/wiki/Star_height_problem`

6. Análisis sintáctico (Parsing) de GIC

Problema: Determinar si una cadena es producida por una Gramática Independiente del Contexto (GIC) dada.

Referencias

[1] Wiki site: `https://en.wikipedia.org/wiki/Context-free_grammar`

7. Autómatas celulares

Es un modelo distinto de autómatas, existen distintas aplicaciones de autómatas celulares.

Referencias

[1] Wiki site: `https://en.wikipedia.org/wiki/Cellular_automaton`

Fechas de entrega

Tendremos tres fechas de entrega

I. **Semana 9: viernes 05 de Junio (10 %).** En esta entrega se definirá el alcance del proyecto. Se deberá entregar la primera versión de la monografía con la introducción, definición del problema y estado del arte el estado del arte del problema y con la propuesta para las dos siguientes entregas.

II. **Semana 12: viernes 26 de Junio (40 %).**

Entrega parcial según lo definido en la primera entrega.

III. **Semana 15: viernes 17 de Julio (50 %).**

Entrega final según lo definido en la primera entrega. Habrá exposición.

Sobre la Monografía

Debe ser escrita en LaTeX. El contenido podrá variar de acuerdo al tipo de proyecto elegido por el grupo. Si su propuesta es de implementación, deberá contener las secciones 1–4. Si su propuesta es de otro tipo, el formato será flexible y conversado con el profesor. En cualquier caso, siempre deberá tener la Sección 1.

1. Introducción, definición del problema y estado del arte.
2. Pseudocódigo de los algoritmos que resuelven el problema (vea `https://en.wikibooks.org/wiki/LaTeX/Algorithms`). Para cada algoritmo, breve explicación de como funciona y del análisis del tiempo de ejecución del mismo.
3. Código de los algoritmos implementados (vea `https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings`).
4. Experimentación numérica: debe generar autómatas aleatórios para probar sus algoritmos (si su problema es de otro tipo esto puede variar). Presente una tabla comparando los tiempos de ejecución de cada algoritmo tomando como parámetro el número de estados del Autómata

Código de honor

Cada grupo debe trabajar individualmente. Cualquier código similar entre grupos hará que ambos tengan cero en la nota final del curso. De la misma manera, será penado con cero copiar código obtenido de otra fuente.