Proyecto para la casa

Teoría de la computación

16 de octubre de 2018

1. Introducción

Este es un proyecto grupal para el curso de Teoría de la Computación, sirve como nota dentro del rubro "tareas para la casa". El grupo es de máximo tres personas.

2. Minimización de AFD

En el presente proyecto nos centraremos en el siguiente problema

Problema MinAFD. Dado un AFD A, encontrar un AFD B que reconozca el mismo lenguaje que A y que tenga el menor núnero de estados posibles.

El método más sencillo consiste en transformar A en un AFN B inviertiendo sus transiciones, después reducir dicho autómata usando los métodos conocidos y repetir este procedimiento una vez más. (vea `https://en.wikipedia.org/wiki/DFA_minimization#Brzozowski's_algorithm`). Este algoritmo, conocido como el algoritmo de **Brzozowski** [2], sin embargo, puede llevar tiempo exponencial. Una primera labor será implementar este algoritmo.

Pregunta 1. Implementar el algoritmo de Brzozowski

Entrada del algoritmo: Un AFD A con n estados y alfabeto $\{0, 1\}$.

Salida del algoritmo: Un AFD B que reconoce $L(A)$ y tiene el menor número de estados posibles.

Tiempo de ejecución del algoritmo: $O(2^n)$.

A continuación estudiaremos algoritmos polinomiales para el problema de minimización. Todas las técnicas a seguir están basadas en el concepto de equivalencia de estados. Para eso hacemos la siguiente definición. Decimos que dos estados p y q son *equivalentes* si para toda cadena w, al comenzar en el estado p y procesar la cadena w se llega a un estado final si y solo si al comenzar en el estado q y procesar la cadena w se llega a un estado final. Si dos estados no son equivalentes, decimos que son *distinguibles*.

Considerando que A tiene n estados, podemos representar esta relación en una tabla de doble entrada para decidir si dos estados son equivalentes o distinguibles. Comenzaremos implementando algoritmos para llenar esta tabla. Estos algoritmos se encuentran en la sección 4.4.1 del libro de Hopcroft [1].

Pregunta 2. Implementar el algoritmo de equivalencia de estados

Entrada del algoritmo: Un AFD A con n estados y alfabeto $\{0, 1\}$.

Salida del algoritmo: Una matriz M de tamanho $n \times n$, donde $M[p, q] = 0$ si p y q son estados distinguibles y $M[p, q] = 1$ si p y q son estados equivalentes.

Tiempo de ejecución: $O(n^4)$

Esta manera de llenar dicha tabla puede ser optimizada. Ver el final de la sección 4.4.2 del libro de Hopcroft. [1].

Pregunta 3. Implementar la versión mejorada del algoritmo de equivalencia de estados

Entrada del algoritmo: Un AFD A con n estados y alfabeto $\{0, 1\}$

Salida del algoritmo: Una matriz M de tamanho $n \times n$, donde $M[p, q] = 0$ si p y q son estados distinguibles y $M[p, q] = 1$ si p y q son estados equivalentes

Tiempo de ejecución: $O(n^2)$

Finalmente, usaremos el algoritmo de la pregunta 3 para resolver nuestro problema MinAFD. Este algoritmo es conocido como el algoritmo de Moore. Ver la sección 4.4.3 del libro de Hopcroft [1].

Pregunta 4. Implentar el algoritmo de Moore.

Entrada del algoritmo: Un AFD A con n estados y alfabeto $\{0, 1\}$

Salida del algoritmo: Un AFD B que reconoce $L(A)$ y tiene el menor número de estados posibles

Tiempo de ejecución: $O(n^2)$

El tiempo de ejecución puede aun ser mejorado a $O(n \lg n)$. La idea de este algoritmo es comenzar con una posible partición de estados e ir refinándola en cada paso. Este algoritmo es conocido como el algoritmo de Hopcroft [3] (ver `https://en.wikipedia.org/wiki/DFA_minimization#Hopcroft's_algorithm`.)

Pregunta 5. Implentar el algoritmo de Hopcroft

Entrada del algoritmo: Un AFD A con n estados y alfabeto $\{0, 1\}$

Salida del algoritmo: Un AFD B que reconoce $L(A)$ y tiene el menor número de estados posibles

Tiempo de ejecución: $O(n \lg n)$

Sus algoritmos deben estar correctos. Para ello, pruebe los algoritmos con al menos tres casos de prueba "pequeños". Estos son: Figura 4.8, Figura 4.14 y Figura 4.15 de [1]

Finalmente, haremos una experimentación numérica para la comparación de tiempo de ejecución de los algoritmos.

3. Consideraciones sobre la implementación

- Pueden usar python, C, C++, o Java.
- Dado un AFD $A = (Q, \Sigma, \delta, q_0, F)$, siempre consideramos que $\Sigma = \{0, 1\}$ y $Q = \{0, 1, 2, \ldots, n-1\}$.
- El formato para lectura del AFD es por entrada estandar y debe usar el siguiente. La primera linea de la entrada tiene, separados por espacios en blanco, el número n de estados del autómata, el estado inicial, el número de estados finales, y los estados finales. Las siguientes $2n$ lineas contienen las transiciones del autómata en el siguiente formato: $p\ a\ q$ si $\delta(p, a) = q$. Por ejemplo, si A es el clásico AFD que acepta una cantidad impar de 0s, A será leído de la siguiente manera:

```
2 0 1 1
0 0 1
0 1 0
1 0 0
1 1 1
```

- La salida del algoritmo debe darse en el mismo formato del ítem anterior.

4. Monografía

Deberá contener las siguientes secciones y debe ser escrito en LaTeX

1. Introducción y definición del problema.
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. Presente una tabla comparando los tiempos de ejecución de cada algoritmo tomando como parámetro el número de estados del AFD.

5. Fechas de entrega

Tendremos dos fechas de entrega

1. **Lunes 12 de Noviembre.** Entrega parcial: Preguntas 1,2 y 3. Se revisará en clase el código de cada grupo individualmente.
2. **Lunes 26 de Noviembre.**

   Entrega final: Preguntas 1,2,3,4,5,6. Se revisará en clase el código de cada grupo individualmente. También deberán entregar la monografía impresa.

6. Código de honor

Cada grupo debe trabajar individualmente. Cualquier código similar entre grupos hará que ambos tengan nota **cero**. De la misma manera, será penado con **cero** copiar código obtenido de otra fuente. Más aún, los alumnos involucrados en un plagio recibirán un descuento de 40 % en su nota final y una llamada de atención.

Referencias

[1] Hopcroft, John E. and Motwani, Rajeev and Ullman, Jeffrey D. Introduction to Automata Theory, Languages, and Computation, 2Nd Edition. *Addison-Wesley Publishing Company.*, 2001.

[2] Brzozowski, Janusz A. Canonical regular expressions and minimal state graphs for definite events *Volume 12 of MRI Symposia Series*, 1962.

[3] Hopcroft, John E. An $n \log n$ algorithm for minimizing states in a finite automaton, Theory of machines and computations (Proc. Internat. Sympos., Technion, Haifa, 1971), New York: Academic Press, pp. 189–196, MR 0403320