Instituto Tecnológico Autónomo de México

Máquinas de Turing
Manual de programación

Fundamentos Matemáticos de Computación

Brenda Becerra González

Andrés Cruz y Vera

Andrea Marín Alarcón

Emilio Alfonso Venancio Landeros

21 de abril de 2018

Índice

1. Introducción y objetivo **2**

2. Variables **2**

3. Clases y métodos **2**

3.1. Kolmogorov 2

3.1.1. cargaObjetivo 3

3.1.2. formateaInput 3

3.1.3. getSimulación 3

3.1.4. maquinaInicial 3

3.1.5. muta 3

3.1.6. evalua 4

3.1.7. distanciaHamming 4

3.1.8. Ejecutable 4

3.2. UTM 5

1. Introducción y objetivo

El programa presentado tiene como objetivo encontrar la Máquina de Turing de 64 estados que genere, o que más se aproxime, a una cadena dada. Se utilizó el programa desarrollado en el primer proyecto parcial para simular una Máquina de Turing.

2. Variables

Las variables que se describen a continuación son las variables globales de la clase Kolmogorov.

- **String mejorMT**: guarda la mejor máquina de Turing encontrada
- **String maquinaAct**: guarda la descripción de la máquina de Turing que se acaba de crear.
- **String mejorCadena**: guarda la mejor aproximación a la cadena meta.
- String cadenaAct: en esta variable se guarda el resultado de la simulación de la máquina actual.
- **String cadenaMeta**: es la cadena con la que se comparan los resultados de las máquinas de Turing creadas.
- **obj**: aquí se guarda el objetivo al que queremos llegar.
- **int fitnessOfBest**: es la variable donde se guarda el resultado de la evaluación de la mejor cadena.
- **int fitness**: se guarda el resultado de la evaluación de la cadena actual.
- **int L**: define la longitud de las descripciones de las máquinas de Turing a crear. En este caso como estamos generando MTs de 64 estados, la variable se inicializa en 1024.
- **int tamCinta**: define el tamaño de la cinta para la máquina de Turing.
- **int maxPmt**: define el máximo número de pasos que puede dar la máquina de Turing.
- **String cinta**: se guarda una cinta llena de ceros de longitud **tamCinta** la cual se usa para realizar las simulaciones de las máquinas de Turing

3. Clases y métodos

3.1. Kolmogorov

En esta clase se implementa un escalador de mutación aleatoria para encontrar la Máquina de Turing que produzca la mejor aproximación a una cadena dada.

Se busca que la cadena meta esté a partir de la mitad de la cinta.

3.1.1. cargaObjetivo

Método estático para leer la cadena meta desde un archivo. Si la cadena no está en formato binario-ASCII, es decir, no está compuesta por "0z "1.^{en}tonces se lee el archivo byte por byte y se transforma la cadena al formato deseado. La cadena meta se guarda en la variable global **obj**.

Una vez que se tiene el objetivo guardado se verifica que el tamaño de la cinta sea de al menos el doble que la longitud del objetivo. Es decir, si la longitud del objetivo es mayor que la mitad del tamaño de cinta entonces el tamaño de cinta se suplica hasta conseguir el tamaño deseado. Luego, se define el número máximo de pasos para la máquina de Turing como la mitad del tamaño de cinta.

3.1.2. formateaInput

Es un método estático que crea una cinta llena de ceros la cual se va a utilizar para la simulación de la máquina de Turing y se guarda en la variable global **cinta**. Además se crea otra cinta del mismo tamaño que la anterior con la cadena buscada a partir de la mitad de la cinta y lo demás son ceros. Esta es la cinta que se utiliza para comparar el resultado de la simulación con lo que queremos obtener y se guarda en la variable global **cadenaMeta**.

3.1.3. getSimulación

Este método utiliza la clase UTM para simular la máquina de Turing guardada en **maquinaAct** y guarda en **cadenaAct** la cinta obtenida.

Se pasan como parámetros del método *NewTape* de la clase UTM la descripción de la máquina de Turing guardada en **maquinaAct**, la cinta llena de ceros creada en el método *formateaInput()*, el máximo número de pasos que se guarda en la variable **maxPmt** y la posición inicial de la cabeza es a la mitad de la cinta.

3.1.4. maquinaInicial

Este método genera, de manera aleatoria, la descripción de una máquina de Turing de 64 estados la cual se guarda en **maquinaAct** y como es la primera máquina que se genera también se guarda en **mejorMT**.

Esto se hace generando un número aleatorio entre 0 y 1, si es menor que 0.5 entonces se escribe un "1", si es mayor entonces se escribe un "0z así 1024 veces.

3.1.5. muta

Este es uno de los métodos principales del escalador pues se genera una nueva descripción de una máquina de Turing.

Se escoge al azar uno o más caracteres de la cadena inicial y se mutan, es decir, si era un 0 se cambia por un 1 y viceversa. Se utiliza la función *Math.random* para determinar, tanto el número de caracteres a mutar como la posición de los mismos.

La nueva descripción de una máquina de Turing se guarda en **maquinaAct**.

3.1.6. evalua

Este método determina el *fitness* de la cadena generada por la máquina de Turing actual, que es lo que el escalador de mutación aleatoria busca minimizar.

Evalúa qué tan parecidas son la cadena generada con la cadena meta, primero es caracter a caracter, luego por pares de caracteres y luego por tercias. Por cada caracter diferente se suma 1, por cada pareja diferente se suma 2 y por cada tercia diferente se suma 4.

3.1.7. distanciaHamming

Es un método que calcula la distancia de Hamming entre dos cadenas. Esto se hace contando cuántos carácteres son diferentes entre sí.

3.1.8. Ejecutable

Dentro del *main* de la clase se implementa el escalador de mutación aleatoria utilizando los métodos y funciones antes descritos.

Primero se mandan a llamar los métodos *cargaObjetivo()* y *formateaInput()* para cargar la cadena meta. El número de iteraciones del escalador se inicializa en 5000 pero el usuario tiene la posibilidad de cambiarlo si quiere y se guarda en **steps**.

Una vez que se tiene el número de iteraciones se crea una primera máquina de Turing con el método *maquinaInicial()* y posteriormente se manda a llamar a *getsimulación()* y *evalua()* para obtener una primera aproximación a la cadena inicial. Como es la primera máquina que se crea su *fitness* se guarda como **fitnessOfBest**.

Luego se inicia un for que realiza los siguientes pasos el número de veces que haya definido el usuario:

1. Muta la máquina actual para obtener una nueva.
2. Obtiene la simulación de la máquina creada.
3. Evalúa la cadena obtenida de la simulación para obtener su **fitness**
4. Si el **fitness** obtenido es menor que **fitnessOfBest** entonces se guarda la máquina actual en **mejorMT**, la cadena actual en **mejorCadena** y el fitness actual en **fitnessOfBest**.
5. Si el **fitnessOfBest** es cero significa que ya encontramos la máquina de Turing que reproduce exactamente la cadena original por lo que ya no es necesario realizar más iteraciones y se sale del for.

De esta manera, cuando termina el for tenemos guardadas la mejor máquina de Turing y la mejor cadena que se pudo encontrar. Después se obtiene la distancia de Hamming entre la mejor cadena y el objetivo para poder calcular

el porcentaje de similitud entre ambas. Además se guarda la descripción de la mejor máquina de Turing y la mejor cadena en archivos de texto para que el usuario pueda consultarlos después.

Finalmente, se imprimen en la consola los resultados obtenidos por nuestro algoritmo.

3.2. UTM

Esta clase simula una Máquina de Turing, contiene un sólo método estático, *NewTape* que recibe cuatro parámetros: la descripción de la máquina de Turing a simular, la cinta sobre la que correrá a simulación de la máquina, el número máximo de iteraciones para la simulación y la posición inicial de la cabeza. Este método regresa la cinta después de correr la simulación.