Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 7 - "Búsqueda en Texto, Algoritmo basado en Autómata"

Profesores: Pablo Barceló
Gonzalo Navarro
Auxiliar: ~~Manuel~~ Ariel Cáceres Reyes

7 de Mayo del 2018

P1. Búsqueda en Texto

El problema de *Búsqueda en Texto* consiste en, dadas dos cadenas T (texto) y P (patrón) con caracteres en un alfabeto Σ, encontrar todas las ocurrencias de P en T (esto es, las subcadenas de T iguales a P).
Parametrizaremos los costos de las soluciones por los tamaños de T, P y Σ que serán n, m y σ respectivamente.

a) Muestre una solución que funcione en tiempo $\mathcal{O}(nm)$

b) Muestre que existe un autómata finito determinista que sea capaz de reconocer $(\Sigma) * P$

c) ¿Cómo se podría usar este autómata para resolver el problema de *Búsqueda en Texto*?

d) Muestre como construir este autómata de forma eficiente.

e) Muestre un algoritmo de tiempo $\mathcal{O}(n+m)$ (puede considerar que $\sigma \in \mathcal{O}(1)$).

f) Muestre un algoritmos de tiempo $\mathcal{O}(n+m)$ incluso cuando $\sigma \in \omega(1)$.

"I visualize a time when we will be to robots what dogs are to humans, and I'm rooting for the machines."
Claude Shannon

Soluciones

P1. Búsqueda en Texto

a) El algoritmo por fuerza bruta

```
1  for i = 0...n − m do
2  |  j ← 1 for j = 1...m do
3  |  |  if P[j]! = T[i + j] then
4  |  |  |  break
5  |  |  end
6  |  end
7  |  if j == m then
8  |  |  report(i + 1)
9  |  end
10 end
```

multiplicando el costo de los ifs obtenemos un costo $\mathcal{O}(nm)$.

b) El autómata es el siguiente:

- Conjunto de estados $Q = \{0, 1, \ldots, m\}$ siendo 0 el estado inicial y m el final.
- Función de transición

$$\begin{aligned} \delta : \Sigma \times Q \to & \quad Q \\ (q, a) \mapsto & \quad \delta(q, a) \quad = \sigma(P[1:q]a) \\ & \qquad\qquad = \text{largo del sufijo más largo de } P[1:q]a \text{ que es prefijo de } P \end{aligned}$$

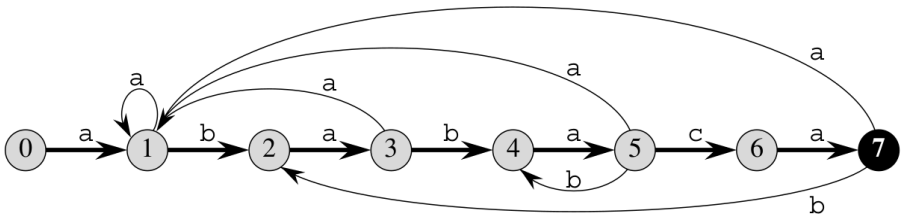

Figura 1: Autómata correspondiente al patrón $P = ababaca$.

Veamos ahora lo siguiente:

> Si el sufijo más largo de lo que se ha leído que es prefijo del patrón es $P[1:i]$, entonces el autómata se encuentra en el estado i

"I visualize a time when we will be to robots what dogs are to humans, and I'm rooting for the machines."
Claude Shannon

(esto pues justo antes de leer el último carácter $P[i]$ se estaba en el estado $j \geq i-1$, por lo que se aplicó $\sigma(P[1:j]P[i]) = \sigma(P[1:i])$ (esto último por ser $P[1:i]$ maximal) $= i$). Con esto es simple ver que cada vez que nuestro prefijo maximal leído sea P el autómata se encontrará en el estado m, reconociendo así $(\Sigma) * P$

c) Podemos correr el autómata anterior sobre T y reportar una ocurrencia cada vez que pasamos por el estado final.

d) Construir el autómata se reduce a construir su función de transición. El algoritmo más simple consiste en hacerlo para cada par $(q, a) \in Q \times \Sigma$ y para cada uno de ellos revisar si $P[1:k]$ es sufijo de $P[1:q]a$, con k desde $\min(q+1, m)$ hasta 0 incurriendo en un costo total de $\mathcal{O}(|\Sigma|m^3) = \mathcal{O}(m^3)$.

   Consideremos ahora la siguiente observación:

   Al calcular $\delta(q, a) = \sigma(P[1:q]a)$
   - Si $P[q+1] = a$, entonces $\delta(q, a) = q + 1$.
   - Si no $(P[q+1] \neq a)$, entonces $\delta(q, a) = \sigma(P[1:q]a) = \sigma(P[2:q]a) \leq q$, y por lo tanto, puede ser calculado corriendo el autómata parcialmente construído sobre $P[2:q]a$.

   Considerando esto podemos construir el autómata en $\mathcal{O}(|\Sigma|m^2) = \mathcal{O}(m^2)$.

   Finalmente, en el caso $P[q+1] \neq a$ no es necesario correr el autómata sobre $P[2:q]a$. Esto se puede lograr manteniendo una variable X que sea el resultado de correr el autómata sobre $P[2:q]$, es decir, $X = \delta(\ldots\delta(\delta(0, P[2]), P[3])\ldots, P[q])$. Con esto, $\delta(q, a) = \delta(X, a)$, cuando $P[q+1] \neq a$. Obteniendo un tiempo de $\mathcal{O}(|\Sigma|m) = \mathcal{O}(m)$.

e) Construimos el autómata de P en $\mathcal{O}(m)$ y luego lo corremos en tiempo $\mathcal{O}(n)$.

f) La solución anterior es en realidad $\mathcal{O}(n + \sigma m)$, y en caso que $\sigma \in \omega(1)$, no es $\mathcal{O}(n+m)$. Esta complejidad se logra con el algoritmo de Morris-Pratt. Este algoritmo define la función $f(i)$ como el largo del prefijo propio más largo de $P[1:i]$ que también es sufijo propio.

   Con esto el algoritmo mantiene un contador j que avanza con el texto y un contador i que avanza con el patrón mientras los caracteres calcen. En caso de no haber calce, i se actualiza a $f(i)$ repitiendo esto hasta que se encuentre un calce o $i = 0$.

   Se puede mostrar que el algoritmo anteriormente descrito es correcto y que $f(i)$ se puede construir en $\mathcal{O}(m)$ (de un modo similar a la construcción del autómata y con un análisis amortizado parecido al que haremos a continuación).

   Veamos ahora que el algoritmo (teniendo f precomputada) es $\mathcal{O}(n)$. Un análisis de peor caso nos lleva a pensar que por cada j, podríamos aplicar la función f $\mathcal{O}(m)$ veces hasta

encontrar un calce, por lo que la complejidad es $\mathcal{O}(nm)$, sin embargo, para poder aplicar la función f un número k de veces, antes debimos haber aumentado i k veces, finalmente, como i no puede haber aumentado mas de n veces, el número de veces que se aplica f en total es n.

"I visualize a time when we will be to robots what dogs are to humans, and I'm rooting for the machines."
Claude Shannon