TALLER PROGCOMP: TRACK STRINGS

STRING HASHING

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

April 6, 2025

Part I

PROBLEMA

COMPAREMOS DOS STRINGS

s **vs** *t*

s y *t* son dos strings y queremos determinar si $s == t$.

COMPAREMOS DOS STRINGS

s **vs** *t*

s y *t* son dos strings y queremos determinar si $s == t$.

Solución directa

- Verificar que los largos sean iguales
- Caracter por caracter verificando que sean los mismos

COMPAREMOS DOS STRINGS

s **vs** *t*

s y *t* son dos strings y queremos determinar si $s == t$.

Solución directa

- Verificar que los largos sean iguales
- Caracter por caracter verificando que sean los mismos

Complejidad: $O(n)$

COMPAREMOS DOS STRINGS

s **vs** *t*

s y *t* son dos strings y queremos determinar si $s == t$.

Solución directa

- Verificar que los largos sean iguales
- Caracter por caracter verificando que sean los mismos

Complejidad: $O(n)$
¿Es posible hacerlo mejor?

Part II

HASHING

Valor de Hash

El valor de hash de un string corresponde al resultado de $hash(s)$, donde este te retorna un número.

Valor de Hash

El valor de hash de un string corresponde al resultado de $hash(s)$, donde este te retorna un número.
¿Cual es la gracia?

Valor de Hash

El valor de hash de un string corresponde al resultado de $hash(s)$, donde este te retorna un número.
¿Cual es la gracia? $s == t$ si $hash(s) == hash(t)$. :o

Valor de Hash

El valor de hash de un string corresponde al resultado de $hash(s)$, donde este te retorna un número. ¿Cual es la gracia? $s == t$ si $hash(s) == hash(t)$. :o En ¡$O(1)$!

HASHING

Valor de Hash

El valor de hash de un string corresponde al resultado de $hash(s)$, donde este te retorna un número. ¿Cual es la gracia? $s == t$ si $hash(s) == hash(t)$. :o En ¡$O(1)$!

Peligro

Encontrar una buena función de hashing es díficil. :(

Polynomial Rolling Hashing

$hash(s) = (s[0] \cdot p^{n-1} + s[1] \cdot p^{n-2} + \ldots + s[n-1] \cdot p^0) \bmod m$
p y m son constantes elegidas.
Cada caracter debe estar en valor número, por ejemplo valor ASCII o alguno personal.

Polynomial Rolling Hashing

$hash(s) = (s[0] \cdot p^{n-1} + s[1] \cdot p^{n-2} + \ldots + s[n-1] \cdot p^0) \bmod m$
p y m son constantes elegidas.
Cada caracter debe estar en valor número, por ejemplo valor ASCII o alguno personal.
Complejidad: $O(n)$

Polynomial Rolling Hashing

$hash(s) = (s[0] \cdot p^{n-1} + s[1] \cdot p^{n-2} + \ldots + s[n-1] \cdot p^0) \bmod m$
p y m son constantes elegidas.
Cada caracter debe estar en valor número, por ejemplo valor ASCII o alguno personal.

Complejidad: $O(n)$

Preprocesamiento

Definimos

- $H[i] = hash(s[0 \ldots i])$
  - $H[0] = s[0]$
  - $H[i] = (H[i-1] \cdot p + s[i]) \bmod m$

Polynomial Rolling Hashing

$hash(s) = (s[0] \cdot p^{n-1} + s[1] \cdot p^{n-2} + \ldots + s[n-1] \cdot p^0) \bmod m$
p y m son constantes elegidas.
Cada caracter debe estar en valor número, por ejemplo valor ASCII o alguno personal.

Complejidad: $O(n)$

Preprocesamiento

Definimos

- $H[i] = hash(s[0 \ldots i])$
  - $H[0] = s[0]$
  - $H[i] = (H[i-1] \cdot p + s[i]) \bmod m$
- PP
  - $PP[0] = 1$
  - $PP[i] = (PP[i-1] \cdot p) \bmod m$

SUBSTRING HASHING $hash(s[i \ldots j])$

$$hash(s[i \ldots j]) = hash(s[0 \ldots j]) - hash(s[0 \ldots i-1]) = H[j] - H[i-1]$$

Substring Hashing $hash(s[i \dots j])$

$$hash(s[i \dots j]) = hash(s[0 \dots j]) - hash(s[0 \dots i-1]) = H[j] - H[i-1]$$

$$hash(s[0 \dots j]) = H[j] = s[0] \cdot p^{j} + s[1] \cdot p^{j-1} + \dots + s[j] \cdot p^{0}$$

$$hash(s[0 \dots i-1]) = H[i-1] = s[0] \cdot p^{i-1} + s[1] \cdot p^{i-2} + \dots + s[i-1] \cdot p^{0}$$

SUBSTRING HASHING $hash(s[i \ldots j])$

$$hash(s[i \ldots j]) = hash(s[0 \ldots j]) - hash(s[0 \ldots i-1]) = H[j] - H[i-1]$$

$$hash(s[0 \ldots j]) = H[j] = s[0] \cdot p^{j} + s[1] \cdot p^{j-1} + \ldots + s[j] \cdot p^{0}$$

$$hash(s[0 \ldots i-1]) = H[i-1] = s[0] \cdot p^{i-1} + s[1] \cdot p^{i-2} + \ldots + s[i-1] \cdot p^{0}$$

$$\begin{aligned} hash(s[i \ldots j]) = {} & s[0] \cdot p^{j} + s[1] \cdot p^{j-1} + \ldots + s[j] \cdot p^{0} \\ & - (s[0] \cdot p^{i-1} + s[1] \cdot p^{i-2} + \ldots + s[i-1] \cdot p^{0}) \end{aligned}$$

Tenemos un problema :(

SUBSTRING HASHING $hash(s[i \ldots j])$

$$hash(s[i \ldots j]) = hash(s[0 \ldots j]) - hash(s[0 \ldots i-1]) = H[j] - H[i-1]$$

$$hash(s[0 \ldots j]) = H[j] = s[0] \cdot p^{j} + s[1] \cdot p^{j-1} + \ldots + s[j] \cdot p^{0}$$

$$hash(s[0 \ldots i-1]) = H[i-1] = s[0] \cdot p^{i-1} + s[1] \cdot p^{i-2} + \ldots + s[i-1] \cdot p^{0}$$

SUBSTRING HASHING $hash(s[i \ldots j])$

$$hash(s[i \ldots j]) = hash(s[0 \ldots j]) - hash(s[0 \ldots i-1]) = H[j] - H[i-1] \cdot p^{j-i+1}$$

$$hash(s[0 \ldots j]) = H[j] = s[0] \cdot p^{j} + s[1] \cdot p^{j-1} + \ldots + s[j] \cdot p^{0}$$

$$hash(s[0 \ldots i-1]) \cdot p^{j-i+1} = H[i-1] = s[0] \cdot p^{j} + s[1] \cdot p^{j-1} + \ldots + s[i-1] \cdot p^{j-i+1}$$

SUBSTRING HASHING $hash(s[i \dots j])$

$$hash(s[i \dots j]) = hash(s[0 \dots j]) - hash(s[0 \dots i-1]) = H[j] - H[i-1]$$

$$hash(s[0 \dots j]) = H[j] = s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[j] \cdot p^0$$

$$hash(s[0 \dots i-1]) \cdot p^{j-i+1} = H[i-1] \cdot p^{j-i+1} = s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[i-1] \cdot p^{j-i+1}$$

$$\begin{aligned} hash(s[i \dots j]) &= s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[j] \cdot p^0 \\ &\quad - (s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[i-1] \cdot p^{j-i+1}) \\ &= s[i] \cdot p^{j-i} + s[i+1] \cdot p^{j-i-1} + \dots + s[j] \cdot p^0 \end{aligned}$$

SUBSTRING HASHING $hash(s[i \dots j])$

$$hash(s[i \dots j]) = hash(s[0 \dots j]) - hash(s[0 \dots i-1]) = H[j] - H[i-1]$$

$$hash(s[0 \dots j]) = H[j] = s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[j] \cdot p^0$$

$$hash(s[0 \dots i-1]) \cdot p^{j-i+1} = H[i-1] = s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[i-1] \cdot p^{j-i+1}$$

$$\begin{aligned} hash(s[i \dots j]) &= s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[j] \cdot p^0 \\ &\quad - (s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[i-1] \cdot p^{j-i+1}) \\ &= s[i] \cdot p^{j-i} + s[i+1] \cdot p^{j-i-1} + \dots + s[j] \cdot p^0 \end{aligned}$$

Lo logramos!

SUBSTRING HASHING $hash(s[i \dots j])$

$$hash(s[i \dots j]) = hash(s[0 \dots j]) - hash(s[0 \dots i-1]) = H[j] - H[i-1]$$

$$hash(s[0 \dots j]) = H[j] = s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[j] \cdot p^0$$

$$hash(s[0 \dots i-1]) \cdot p^{j-i+1} = H[i-1] \cdot p^{j-i+1} = s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[i-1] \cdot p^{j-i+1}$$

$$\begin{aligned} hash(s[i \dots j]) &= s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[j] \cdot p^0 \\ &\quad - (s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[i-1] \cdot p^{j-i+1}) \\ &= s[i] \cdot p^{j-i} + s[i+1] \cdot p^{j-i-1} + \dots + s[j] \cdot p^0 \end{aligned}$$

Lo logramos!
Con esto, con un costo de precomputo de $O(n)$, podemos comparar substrings en $O(1)$.

SUBSTRING HASHING $hash(s[i \dots j])$

$$hash(s[i \dots j]) = hash(s[0 \dots j]) - hash(s[0 \dots i-1]) = H[j] - H[i-1]$$

$$hash(s[0 \dots j]) = H[j] = s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[j] \cdot p^0$$

$$hash(s[0 \dots i-1]) \cdot p^{j-i+1} = H[i-1] = s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[i-1] \cdot p^{j-i+1}$$

$$\begin{aligned} hash(s[i \dots j]) &= s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[j] \cdot p^0 \\ &\quad - (s[0] \cdot p^j + s[1] \cdot p^{j-1} + \dots + s[i-1] \cdot p^{j-i+1}) \\ &= s[i] \cdot p^{j-i} + s[i+1] \cdot p^{j-i-1} + \dots + s[j] \cdot p^0 \end{aligned}$$

Lo logramos!
Con esto, con un costo de precomputo de $O(n)$, podemos comparar substrings en $O(1)$.

- Por ejemplo, dado un string s y un patrón p ¿cuántas veces aparece p en s?

Una implementación la pueden encontrar en el Handbook-USM.

¿Es seguro utilizar hashing?

IMPORTANTE

¡Hashing es probabilistico!

¿Es seguro utilizar hashing?

IMPORTANTE

¡Hashing es probabilistico! Puede suceder que $hash(s) == hash(t)$ y $s \neq t$.

¿Es seguro utilizar hashing?

IMPORTANTE

¡Hashing es probabilistico! Puede suceder que $hash(s) == hash(t)$ y $s \neq t$.

Constantes

Ideal que p y m sean números primos, por ejemplo pueden usar:

¿Es seguro utilizar hashing?

IMPORTANTE

¡Hashing es probabilistico! Puede suceder que $hash(s) == hash(t)$ y $s \neq t$.

Constantes

Ideal que p y m sean números primos, por ejemplo pueden usar:

- $p = 23$ y $m = 10^9 + 9$ (para solo letras minúsculas)
- $p = 53$ y $m = 10^9 + 9$ (solo letras minúsculas y mayúsculas)
- $p = 911382323$ y $m = 972663749$ (otra opción)

¿Es seguro utilizar hashing?

IMPORTANTE

¡Hashing es probabilistico! Puede suceder que $hash(s) == hash(t)$ y $s \neq t$.

Constantes

Ideal que p y m sean números primos, por ejemplo pueden usar:

- $p = 23$ y $m = 10^9 + 9$ (para solo letras minúsculas)
- $p = 53$ y $m = 10^9 + 9$ (solo letras minúsculas y mayúsculas)
- $p = 911382323$ y $m = 972663749$ (otra opción)

También pueden usar dos hashings :o, con diferentes p y m.

¿Es seguro utilizar hashing?

IMPORTANTE

¡Hashing es probabilistico! Puede suceder que $hash(s) == hash(t)$ y $s \neq t$.

Constantes

Ideal que p y m sean números primos, por ejemplo pueden usar:

- $p = 23$ y $m = 10^9 + 9$ (para solo letras minúsculas)
- $p = 53$ y $m = 10^9 + 9$ (solo letras minúsculas y mayúsculas)
- $p = 911382323$ y $m = 972663749$ (otra opción)

También pueden usar dos hashings :o, con diferentes p y m.

Colisiones

Dependiendo de la situación las colisiones pueden aumentar, por ejemplo la paradoja del cumpleaños.

¿Es seguro utilizar hashing?

IMPORTANTE

¡Hashing es probabilistico! Puede suceder que $hash(s) == hash(t)$ y $s \neq t$.

Constantes

Ideal que p y m sean números primos, por ejemplo pueden usar:

- $p = 23$ y $m = 10^9 + 9$ (para solo letras minúsculas)
- $p = 53$ y $m = 10^9 + 9$ (solo letras minúsculas y mayúsculas)
- $p = 911382323$ y $m = 972663749$ (otra opción)

También pueden usar dos hashings :o, con diferentes p y m.

Colisiones

Dependiendo de la situación las colisiones pueden aumentar, por ejemplo la paradoja del cumpleaños. ¡Importante hacer el analisis del problema!

REFERENCES I