Suffix array

Pedro O. Pérez M., PhD.

Análisis y diseño de algoritmos avanzados
Tecnológico de Monterrey

pperezm@tec.mx

12-2022

Contenido

- Los algoritmos de hashing son muy útiles en muchas áreas de las ciencias computacionales, sobre todo porque, usando una buena función de hash, es posible reducir la búsqueda a un tiempo casi constante $O(1)$.
- Por ejemplo, dadas las cadenas S_1, de longitud n_1, y S_2, de longitud n_2, indicar si son iguales o no. Si realizamos la comparación a fuerza bruta, el algoritmo resultante tiene un orden $O(min(n_1, n_2))$. Sin embargo, usando una buena función de hash podríamos logar un orden $O(1)$.

- La idea principal de una función hash para cadenas es convertir una cadena en un número entero. Para esto, la función debe cumplir la condición de que, si dos cadenas son iguales, les debe asignar el mismo número entero. Cuando la función le asigna el mismo número a dos cadenas diferentes se dicen que hubo una **colisión**. La función ideal sería aquella que a cada cadena le asignara un entero diferente.
- En realidad, es muy complicado hacer una función con tal condición, pero es suficiente con generar una función donde la probabilidad de asignarle el mismo entero a dos cadenas diferentes sea muy baja, y ese tipo de funciones sí se han podido generar.
- Un ejemplo es la llamada **Polynomial Rolling Hash Function**.

Sea S, una cadena de longitud n, a la que se le quiere aplicar la función hash, y sean $S[0]$, $S[1]$, ..., $S[n-1]$ las representaciones en enteros de cada uno de los caracteres que lo forman, la función Polynomial Rolling Hash Function para la cadena S se define como:

$$\begin{aligned} prhs(S) &= (S[0] * p^0 + S[1] * p^1 + ... + S[n-1] * p^{n-1}) \mod m \\ &= (\sum_{i=0}^{n-1} S[i] * p^i) \mod n \end{aligned}$$

Donde, p y m son número enteros positivos que se deben seleccionar. Una forma común es que ambos sean números enteros primos.

- En el caso de P, se recomienda que sea un número primo cercano al número de caracteres en el alfabeto. Por ejemplo, para los caracteres en español, que son 27, el número primo más cercano es el 31, por lo que $p = 31$ es una buena opción. Si se incluyen también las mayúsculas, el conjunto sube a 54 y entonces $p = 53$ es una buena opción.
- En lo que respecta a m, debe ser un número muy grande, debido a que la probabilidad de que haya colisión es, aproximadamente, $1/m$. Una buena selección ha sido un número primo muy grande, por ejemplo, $m = 10^9 + 9$, lo que nos da una probabilidad de colisión de alrededor de 10^{-9}.

- Si bien, una probabilidad de colisión de 10^{-9} es muy baja, si se hace una sola comparación. En algunos problemas es necesario comparar una cadena con un conjunto de otras cadenas. Si ese conjunto es de 10^6 elementos, la probabilidad de que suceda una colisión aumenta a 10^{-3}, y si se comparan las 10^6 cadenas entre si, la probabilidad se acerca peligrosamente a 1.
- Una forma muy común y simple de obtener mejores probabilidades de colisión, incluso con muchas comparaciones, es hacer dos funciones hash, con diferentes p y m. Se aplican las dos funciones hash y sólo se declaran iguales las cadenas si el resultado de ambas funciones son iguales. En este caso, si m para la segunda función es también cercana a 10^9, aplicar las dos funciones es equivalente a tener una m aproximadamente igual a 10^{18}.

```cpp
long long prhf(const std::string &s) {
  int p = 31;
  int m = 1e9 + 9;
  long long result = 0;
  long long power = 1;

  for (int i = 0; i < s.size(); i++) {
    result = (result + ((s[i] - 'a' + 1) * power)) % m;
    power = (power * p) % m;
  }
  return result;
}
```

Arreglos de Sufijos (Suffix Array)

El Arreglo de sufijos (Suffix Array) es una estructura de datos que fue propuesta por Udi Manber y Gene Myers en 1990. Es muy utilizada en varias aplicaciones relacionadas con el análisis de cadenas. Se define de la siguiente forma:

- Dada una cadena S, de longitud n, su arreglo de sufijos es un arreglo de enteros que contiene las posiciones que tienen los $n + 1$ sufijos en la cadena $T = S\$$, ordenados lexicográficamente, considerando \$ como el primer carácter del alfabeto.

Procedure 1 SUFFIX_ARRAY

1. FORMAR LA CADENA DE DE TRABAJO $T \leftarrow S\$$
2. FORMAR TODOS LOS SUFIJOS DE T
3. ORDENAR LEXICOGRÁFICAMENTE LOS SUFIJOS ENCONTRADOS CONSIDERANDO A $ COMO PRIMER SÍMBOLO DEL ALFABETO
4. FORMAR EL ARREGLO A CON LAS POSICIONES DE INICIO EN LA CADENAS T DE CADA UNO DE LOS SUFIJOS

return A