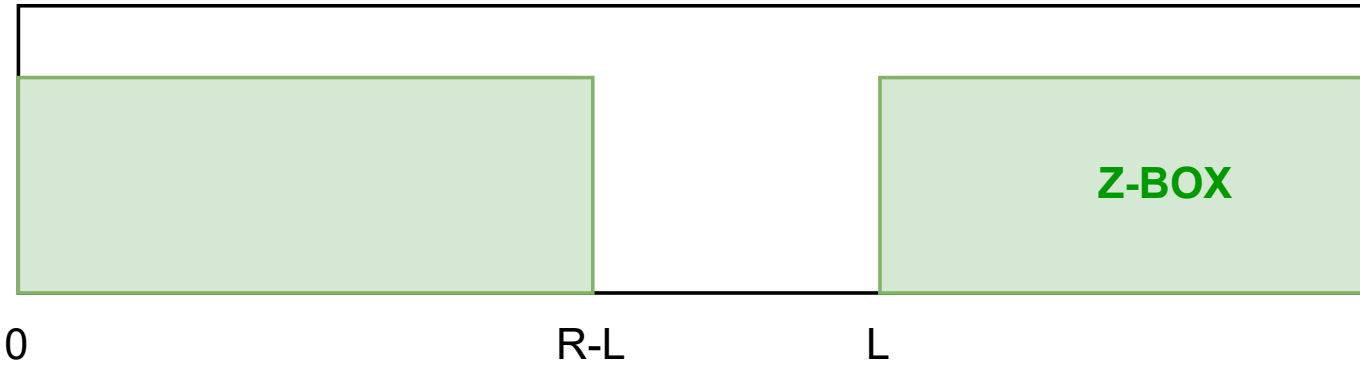
Algoritmo Z

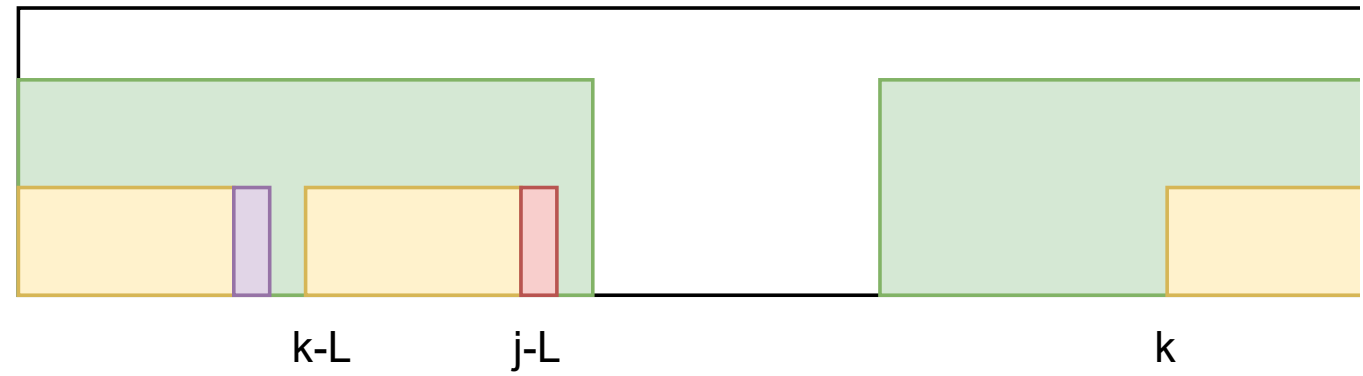
El algoritmo Z se utiliza para encontrar el prefijo común más largo (LC
tiene un string, por ejemplo S="casacas", este hallará que S[4:] = "cas
El algoritmo devuelve un vector con los valores de cada elemento, aur
asume como 0. Para el ejemplo anterior el vector sería Z=[0,0,0,0,3,0,
sería: por cada caracter k en el string S, iterar hasta encontrar el prime
vector Z.

Pero hay una forma de acelerar esto utilizando informacion que ya se
valor Z[L] = R-L+1, donde R y L son numeros. Esto implica directamer



Más adelante en el algoritmo, si queremos evaluar Z[k] podemos utiliz

informacion de k-L como se ve en la siguiente imagen:



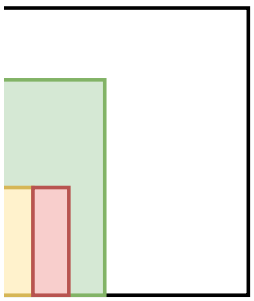
En este ejemplo, tenemos que el intervalo S[k-L : j-L] coincide con S[0 :
k+1] != S[j-L] por lo que Z[k-L] = j-k+1. Todo esto ya fue calculado previ
el que se calcula Z[k], y como se ve en la imagen esto implica que Z[k]
embargo, en otros casos, por ejemplo S="abacabazababacaba", Z[2] !=
ocurre cuando el LCP se sale del Zbox, en estos casos desconocemos
sigue por lo que si es necesario evaluar a este y elementos siguientes.

:P) entre un string y sus sufijos, es decir si se
s" coincide con S[0:] = "casacas" en 3 caracteres.
nque el primero es redundante asi que aveces se
,0]. Una alternativa más intuitiva para el LCP
er i tal que S[i] != S[k+i] y colocar ese i en el

conoce: asumamos que de un string S se halló el
nte que el intervalo S[L:R] es identico a S[0:R-L]

R

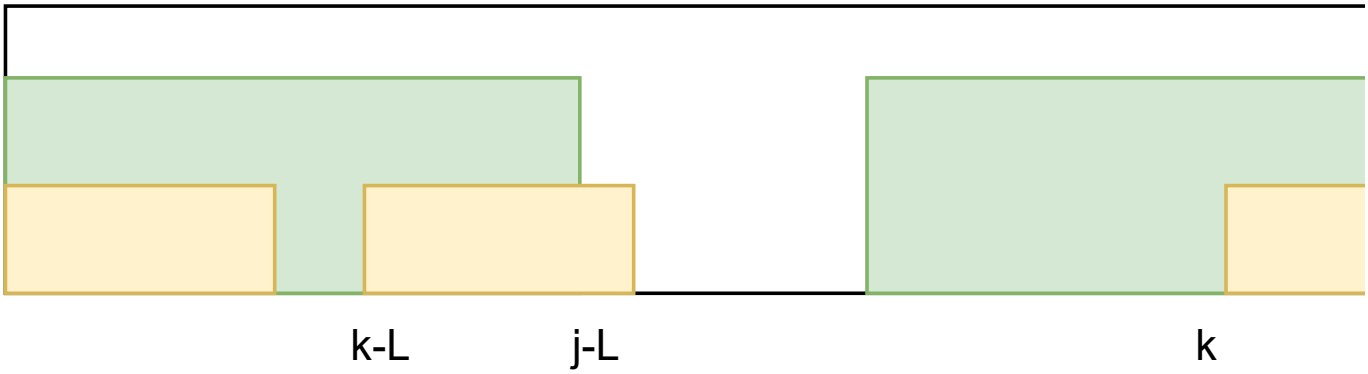
:ar



j

: j-k], pero S[j-
io al momento en
= Z[k-L]. Sin
= Z[10]. Esto
s que elemento
De forma

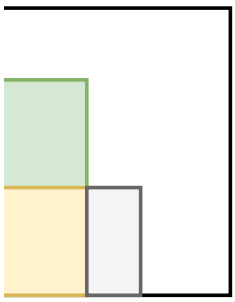
general: Z[k] >= Z[k-L], cuando ocurren casos como este:



Codigo en C++

```
vector<int> z_function(string s) {
    int n = s.size();
    vector<int> z(n);
    int l = 0, r = 0;
    for(int i = 1; i < n; i++) {
        if(i < r) {
            z[i] = min(r - i, z[i - l]);
        }
        while(i + z[i] < n && s[z[i]] == s[i
            z[i]++;
        }
        if(i + z[i] > r) {
            l = i;
            r = i + z[i];
        }
    }
    return z;
}
```

De forma



j

```
+ z[i]]) {
```

Lo que hace este codigo es utilizar la propiedad que comenté anteriormente para directamente hacer que Z[k] sea el valor de Z[k-L] si este está dentro del bloque (inicialmente, en la primera iteracion, como no hay bloque se saltea este paso), luego en el WHILE se avanza hasta que no haya coincidencia y finalmente se actualiza los valores de L y R ,extremos del Z-box, si se salió de este.

Analizando la complejidad, para cada uno de los n elementos se hace:

- Una comparacion O(1)

- Un while que se repite una cantidad x de veces

- Una actualizacion de 2 variables O(1)

Lo que es importante notar, es que cuando en un indice k se ejecuta el while x = Z[k] veces, en las siguientes iteraciones todos los elementos entre S[k : k+x] serán omitidos, por lo que, amortizadamente, el while se ejecutará O(n) veces a lo largo de la ejecucion del algoritmo.