Alberto Llamas González

(1)

Compara s y t

Orden en caso peor

Match e indel → O(1)

MATCH, INSERT, DELETE → constantes

FUNCIÓN RECURSIVA

T(n) int string-compare (char *s, char *t, int i, int j){ TAMAño del caso n = |j - i|

```
int k;                                   O(1)
int opt[3];                              O(1)
int lowest_cost;                         O(1)
if (i == 0) return (j * indel(' '));     O(1)
if (j == 0) return (i * indel(' '));     O(1)
opt[MATCH] = string_compare(s,t,i-1,j-1) + match(s[i],t[j]);   T(n-2), O(1)
opt[INSERT] = string_compare(s,t,i,j-1) + indel(t[j]);         T(n-1), O(1)
opt[DELETE] = string_compare(s,t,i-1,j) + indel(s[i]);         T(n-1), O(1)
lowest_cost = opt[MATCH];                O(1)
For (k = INSERT; k <= DELETE; k++)       O(1) O(1) O(1)
    if(opt[k] < lowest_cost)             O(1)
        lowest_cost = opt[k];            O(1)
                                         } O(1)   } * O(n) (desde insert hasta delete)
return lowest_cost;                      O(1)
}
```

Viendo la eficiencia de la función podemos obtener la siguiente ecuación en recurrencias

$$T(n) = 2T(n-1) + T(n-2) + n + 1$$

Resolvemos la ecuación en recurrencias. Se trata de una Ecuación Lineal No Homogénea. Pasamos los términos recurrentes a un lado.

$$T(n) - 2T(n-1) - T(n-2) = n + 1$$

Resolvemos la parte recurrente como si fuera una ecuación Lineal Homogénea.

Alberto Llamas
González

Reescribimos $T(n-i)$ como x^i:

$$x^n - 2x^{n-1} - 1x^{n-2} = n+1$$

Sacamos factor común:

$$(x^2 - 2x - 1)\, x^{n-2} = n+1$$

Resolvemos la parte homogénea:

$$x = \frac{2 \pm \sqrt{4+4}}{2} = \frac{2 \pm \sqrt{8}}{2} \quad \begin{cases} 1+\sqrt{2} = x_1 \\ 1-\sqrt{2} = x_2 \end{cases}$$

$$P_H(x) = (x - (1+\sqrt{2}))(x - (1-\sqrt{2}))$$

Escribimos ahora la parte no homogénea como $b_1^n q_1(n)$:

$$n+1 = b_1^n q_1(n) = 1^n (n+1) \text{ donde } b_1 = 1 \text{ y } q_1(n) = n+1 \text{ y } d_1(q_1(n)) = 1$$

Polinomio característico: $p(x) = (x-(1+\sqrt{2}))(x-(1-\sqrt{2}))(x-1)^{2=d_1+1}$

Aplicamos la fórmula para el cálculo de eficiencia: $T(n) = \sum_{i=1}^{r} \sum_{j=0}^{M_i - 1} c_{ij} R_i^n n^j$

$$T(n) = c_{10}(1+\sqrt{2})^n + c_{20}(1-\sqrt{2})^n + c_{30} 1^n n^0 + c_{31} 1^n n =$$

$$= c_{10}(1+\sqrt{2})^n + c_{20}(1-\sqrt{2})^n + c_{30} + c_{31} n \in \boxed{O(1+\sqrt{2})^n}$$

Luego en el caso peor el orden será $O(1+\sqrt{2})^n$

Alberto Llamas Gonzalez

② Justificar si tiene orden exacto

Para que sea orden exacto $\Theta(f(n))$, $T(n) = k \cdot f(n)$. Por tanto debe cumplirse que

$$\left.\begin{array}{l} T_A(n) \leq k_1 f(n) \\ T_A(n) \geq k_2 g(n) \end{array}\right\} \Leftrightarrow k_1 = k_2 \text{ y } f(n) = g(n)$$

En el caso mejor, la distancia entre ambas cadenas será la misma, es decir ó $i == 0$ ó $j = 0$ por lo que el orden en el mejor caso será $\Omega(1)$

Como hemos calculado anteriormente, $O((1+\sqrt{2})^n)$ es el peor caso luego como $f(n) = (1+\sqrt{2})^n \neq 1 = g(n) \Rightarrow k_1 \neq k_2$ y por tanto no tiene orden exacto