

# JUSTIFICACIÓN DE UN TEXTO



U N I V E R S I D A D  
COMPLUTENSE  
M A D R I D

**ALBERTO VERDEJO**

# Justificación de un texto

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda.

←————— L —————→

# Justificación de un texto

- ▶ Dadas  $n$  palabras de longitudes  $l_1, l_2, \dots, l_n$ , distribuirlas en un párrafo con líneas de longitud  $L$ .
- ▶ Si una línea tiene las palabras de la  $i$  a la  $j$ , el número de espacios extra es

$$L - (j - i) - \sum_{k=i}^j l_k.$$

- ▶ Añadir dichos espacios tiene la siguiente penalización:

$$\text{penaliza}(i, j) = \left( L - (j - i) - \sum_{k=i}^j l_k \right)^3$$

# Solución 1

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha **mucho** tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda.

# Solución 1

$p\acute{a}rrafo(i,j)$  = penalización *mínima* al formatear las palabras de la  $i$  a la  $n$  empezando en una línea con  $j$  espacios libres



# Definición recursiva

- Casos recursivos,  $i \leq n$

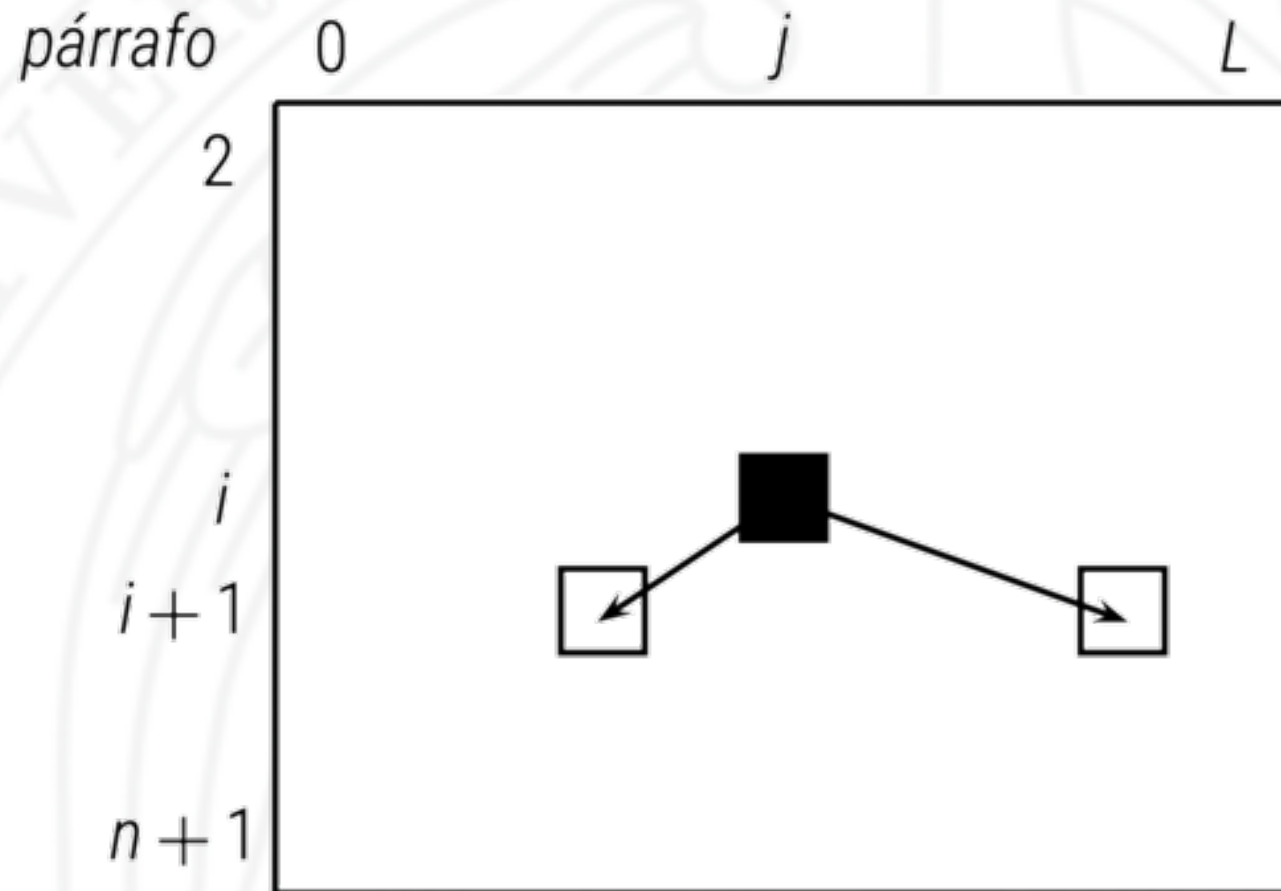
$$\text{párrafo}(i,j) = \begin{cases} \text{párrafo}(i+1, L - l_i) + j^3 & \text{si } l_i + 1 > j \\ \min(\text{párrafo}(i+1, L - l_i) + j^3, \\ \text{párrafo}(i+1, j - (l_i + 1))) & \text{si } l_i + 1 \leq j \end{cases}$$

- Casos básicos:

$$\text{párrafo}(n+1, j) = 0$$

- Llamada inicial:  $\text{párrafo}(2, L - l_1)$

# Tabla



# Implementación

```
int justificar(vector<int> const& l, int L, int i, int j,
               Matriz<int> & tabla) {
    int n = l.size();
    int & res = tabla[i][j];
    if (res == -1) {
        if (i == n) res = 0;
        else if (l[i] + 1 > j)
            res = justificar(l, L, i+1, L - l[i], tabla) + j*j*j;
        else
            res = min(justificiar(l, L, i+1, L - l[i], tabla) + j*j*j,
                      justificiar(l, L, i+1, j - (l[i] + 1), tabla));
    }
    return res;
}
```



## Solución 2

En un lugar de la Mancha, de cuyo nombre **no quiero acordarme, no ha mucho** tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda.

## Solución 2

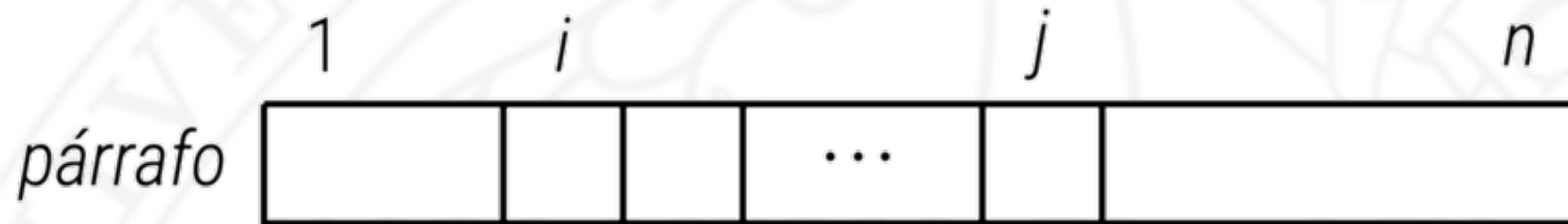
$p\acute{a}rrafo(i)$  = penalización *mínima* al formatear las palabras de la  $i$  a la  $n$  empezando en una línea en blanco

$$p\acute{a}rrafo(i) = 0 \quad \text{si } caben(i, n)$$

$$p\acute{a}rrafo(i) = \min_{\substack{i \leq j < n \\ caben(i, j)}} \{penaliza(i, j) + p\acute{a}rrafo(j + 1)\}$$


$$\text{donde } caben(i, j) = (j - i) + \sum_{k=i}^j l_k \leq L$$

# Tabla



# Implementación

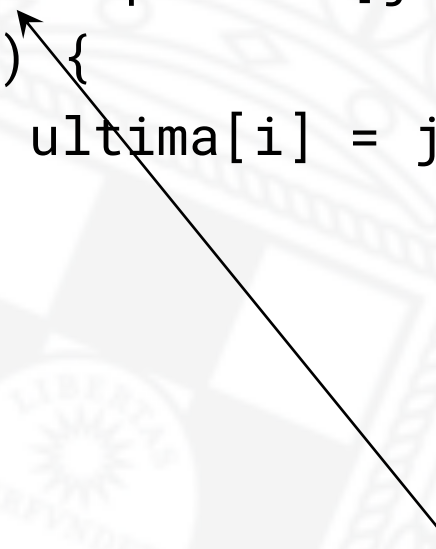
```
int justificar(vector<int> const& l, int L, vector<int> & ultima) {  
    int n = l.size();  
    vector<int> parrafo(n);  
    ultima = vector<int>(n);  
    // casos básicos  
    int i = n-1; int suma = l[i];  
    while (i >= 0 && ((n-1)-i) + suma <= L) {  
        parrafo[i] = 0;  
        ultima[i] = n-1;  
        --i;  
        suma += l[i];  
    }  
}
```


$$caben(i,j) = (j-i) + \sum_{k=i}^j l_k \leq L$$

# Implementación

```
// casos recursivos
```

```
while (i >= 0) {  
    int j = i; suma = l[i]; parrafo[i] = INF;  
    while (j < n-1 && (j-i) + suma <= L) {  
        int pen = L - (j-i) - suma;  
        int nuevo = pen*pen*pen + parrafo[j+1];  
        if (nuevo < parrafo[i]) {  
            parrafo[i] = nuevo; ultima[i] = j;  
        }  
        ++j; suma += l[j];  
    }  
    --i;  
}  
return parrafo[0];  
}
```


$$\text{penaliza}(i,j) = \left( L - (j-i) - \sum_{k=i}^j l_k \right)^3$$



# Implementación

```
vector<string> texto;  
...  
int i = 0;  
while (i < texto.size()) {  
    cout << texto[i]; ++i;  
    int ult = ultima[i];  
    while (i <= ult) {  
        cout << ' ' << texto[i];  
        ++i;  
    }  
    cout << '\n';  
}
```