

Correctitud Broken Keyboard (1988)

$$\text{length} = i - 0 + 1 = i \quad \text{length} = j - i - 1 \quad \text{length} = k - j - 1$$

'[' := inicio.

']' := fin.

Sea $T = A \dots K \] L \dots O [P \dots U [V \dots Z$ donde T es un texto de tamaño n .
 $\overset{0}{\nearrow} \quad \uparrow \quad \uparrow \quad \uparrow \quad \nearrow_{n-1}$
 $T[i] \quad T[j] \quad T[k] \quad \text{length} = n - 1 - (k+1) + 1 = n - k - 1$

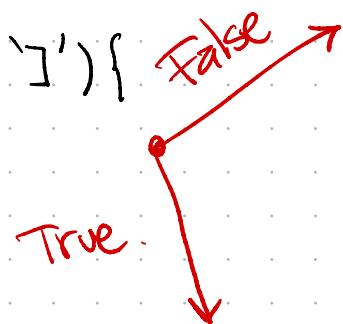
Además, los subconjuntos tienen un largo definido que se separan por los caracteres '[' := inicio y ']' := fin. Así, se tiene que:

$$\sum \text{length}'s = i + (j - i - 1) + (k - j - 1) + (n - k - 1) = n - 3$$

cantidad de ']' o '[' en el texto.
 ↓ en general
 $n - (\#['] + \#']')$

Luego, en el primer if se tiene que:

```
if (ch != '[' and ch != ']') {
    str = str + ch
}
```



Implica que T comienza con ']' o con '['. No se asigna nada a la variable string str

En caso de ser cierto, entonces:

$$str = str + ch$$

initialmente: $str = \emptyset$

$$str = \emptyset + 'T[0] \dots T[i-1]'$$

$$\Rightarrow str = 'A \dots K'$$

$A \dots K$

Como $ch \neq '[' \circ ch \neq ']'$, entonces se sabe que $T[i] = ']' \circ ']'$, ya que se agregó el texto anterior.

Ahora, a saber:

inicio front <deque> ans



fin.
rear

home = True or False

'[' ']'
inicio fin

```
while( ch != 'salto de linea') {
    if (ch == '[' or ch == ']') {
        Push (str, ans, home)
        if (ch == '[') {
            home = True
        }
        else {
            home = False
        }
    }
    else {
        str = str + ch
    }
}
prev_ch = ch
```

Explicación: Este ciclo es para una linea y se corta con '\n'.

Inicialmente entrará si o si al primer if, ya que si ya se guardó el string, entonces ahora le toca al ']' o '['. En caso contrario, $T[0] = ']'$ ó $T[0] = '['$. Así, la función Push:

home = False (fin)

str = 'A...m'

Push back ($T[0] \dots T[i-1]$)



ans.size() = i + 1

str.clear() → str = empty string

Luego, en el otro if hay dos casos:

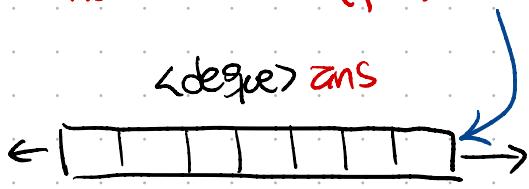
2do if $\begin{cases} \text{home} = \text{True} \\ \text{home} = \text{False} \end{cases}$

Ahora como $T[i] = ']'$, entonces el texto que sigue será puesto al final. En este caso, iremos por el camino de $\text{home} = \text{False}$. Sigue leyendo el texto que ahora sería $ch = 'T[i+1] \dots T[j-1]'$ como no entra en el primer if, entonces:

$str = str + ch \Rightarrow str = 'T[i+1] \dots T[j-1]'$

Como $T[j] = '['$, entonces entra al primer if:

home = False (fn) Push-back ($str = 'T[i+1]...T[j-1]'$)



$$ans = \underbrace{A \dots K}_{length=j-1} L \dots O$$

$$ans.size() = / + (j-1) = j-1 \quad \checkmark (T[0] \dots T[j])$$

str.clear()

Sabiendo que $T[j] = '['$, entra al segundo if. Por lo tanto, home = True.

Ahora, $ch = 'T[j+1] \dots T[k-1]'$, entonces:

$$str = str + ch = 'T[j+1] \dots T[k-1]' = 'P \dots u'$$

Como $T[k] = '['$, entra al primer if:

Front-back ($str = 'T[j+1] \dots T[k-1]'$)



$$\underbrace{k-j-1}_{\text{length}} \quad \underbrace{i}_{\text{index}} \quad \underbrace{j-i-1}_{\text{length}}$$

$$ans = P \dots U A \dots K L \dots O$$

home = True (inicio) $ans.size() = (k-j-1) + / + (j-i-1) = k-2 \quad \checkmark$
 $\# ']' o '[' hasta k$

Ya que $T[k] = '['$, entra también al segundo if. Por ende, home = True

Ahora $ch = 'T[k+1] \dots T[n-1]'$, entonces:

$$str = str + ch = 'T[k+1] \dots T[n-1]' = 'v \dots z'$$

Por último, dentro del while se tiene lo sig:

$$prev_ch = ch.$$

if (prev-ch != '[' and prev-ch != ']') {

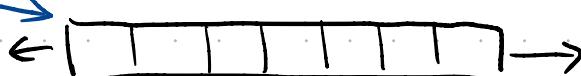
Push (str, ans, home)

}

En este caso, como:

Front-back ($\text{str} = 'T[k+1] \dots T[n-1]'$)

<deque> ans



str.clear()

$$\begin{aligned}\text{ans.size}() &= (n-k-1) + (\cancel{k-j-1}) + \cancel{i} \\ &+ (\cancel{j-k-1}) \\ &= n-3 = \text{T.length()} \quad \square\end{aligned}$$

Al salir del while está esta condición, la cual se debiese cumplir por la naturaleza del algoritmo del ciclo mencionado.

$\text{str} = 'T[k+1] \dots T[n-1]'$

$\text{home} = \text{True} \text{ (inicio)}$

$n-k-1 \quad k-j-1 \quad i \quad j-i-1$

$\text{ans} = \underbrace{v \dots z}_{n-k-1} \underbrace{P \dots A}_{k-j-1} \underbrace{U \dots L}_{i-j+i-1} \dots O$

Finalmente, se imprime la solución con un for recorriendo la deque.

OBS: En caso de existir más texto con saltos de linea de por medio, la única diferencia es que el primer while se ejecutará más de una vez.

ANEXO

Invariante

Se propone la siguiente invariante:

$$\text{ans.size}()^{i+1} = \text{ans.size}()^i + \text{str.length}()^{i+1}$$

Esempio:

$[a[b[c]]] \text{FGD}$

↓
inizio

$a[b[c]] \text{FGD}$

↓
inizio

$ba[c]] \text{FGD}$

↓
inizio

$cba]] \text{FGD}$

↓
Fin
↓
Fin
↓
Fin

$cba \text{FGD}$

0 1 2 3 4 5 6 7 8 9 10 11

$[a[b[c]]] \text{FGD}$

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

$i=0 \quad j=2 \quad k=4 \quad L=6 \dots m=9 \quad n=11$

$$1) j-i = 2-1$$

$$2) k-j = 4-2 = 2-1$$

$$3) L-k = 6-4 = 2-1$$

$$4) n-m = 11-9 = 2+1$$