Escritura del problema del ordenamiento de datos

Juan Sebastián Herrera Guaitero[1]

[1]Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana
Bogotá, Colombia
jsebastianherrera@javeriana.edu.co

11 de agosto de 2022

Resumen

En este documento se presenta la formalización de una secuencia mayoritaria, junto con la descripción de un algoritmo iterativo y recursivo que lo solucionan. Además, se presenta un análisis teórico de la complejidad de los dos algoritmos presentados. **Palabras clave:** algoritmo, formalización, complejidad.

Índice

1. Formalización del problema

Dada una secuencia S de elementos $a \in \mathbb{T}$ se pide encontrar el elemento mayoritario entre los elementos dados S_i tal que $S_i \geq \lceil \frac{|S|}{2} \rceil$.
Recordemos que los números pueden ser naturales ($\mathbb{N}$), enteros ($\mathbb{Z}$), racionales o quebrados ($\mathbb{Q}$), irracionales ($\mathbb{I}$), complejos ($\mathbb{C}$) y $\mathbb{T}$ puede ser cualquier conjunto.

1.1. Definición del problema "secuencia mayoritaria"

Así, el problema de la secuencia mayoritaria se define a partir de; una secuencia S de elementos $a \in \mathbb{T}$ cuyo objetivo principal es encontrar un **n** tal que $\exists n_{S_i}$ donde

$$n_{S_i} \geq \lceil \frac{|S|}{2} \rceil$$

.

- Entrada: $S = \langle a_i \in \mathbb{T} \rangle$.
- Salida:
  - **true** si $\exists n_{S_i}$ donde $n_{S_i} \geq \lceil \frac{|S|}{2} \rceil$
  - **false** si $\forall n_{S_i}$ implica $n_{S_i} < \lceil \frac{|S|}{2} \rceil$

2. Algoritmos de solución

En esta sección encontrará la solución de manera iterativa y de la forma dividir y vencer en diferentes subsecciones.

2.1. Posible solución

Dada una secuencia $S =< 1, -1, 1, 0, -1, 1, 1 >$ donde $|S| = 7$:

1. Obtener U, que son aquellos valores que solo tienen una única ocurrencia en toda la secuencia.
2. Comparar los elementos U con los de S hasta que $c_{u_i} \geq \lceil \frac{|S|}{2} \rceil$

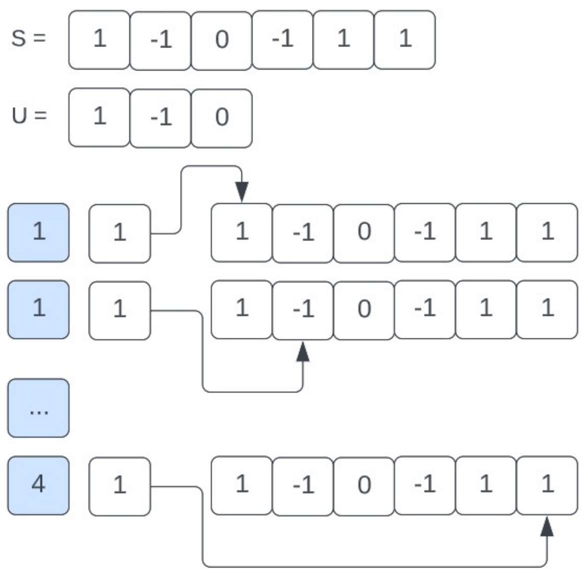

Figura 1: Posible solución

3. Al final, comparar el contador(Caja azul) con $\lceil \frac{|S|}{2} \rceil$ para dar el resultado.

2.2. Algoritmo iterativo

Algoritmo 1 Secuencia mayoritaria

```
procedure SecMayoritaria(S)
    contador ← 0
    i ← 1
    U ← set(S)                                   ▷ Se obtienen los valores únicos de S.
    while i < |U| ∧ contador < |S|//2 do          ▷ '//' Quiere decir valor piso
        contador ← 0
        for j ← 1 to |S| do
            if S_j = U_i then
                contador+ ← 1
            end if
            i+ ← 1
        end for
    end while
    if contador > |S|//2 then
        return True
    else
        return False
    end if
end procedure
```

Complejidad: Por inspección de código hay dos ciclos (un *mientras-que* anidado dentro de un ciclo *para-todo*) anidados que, en el peor de los casos, S y U son iguales por tanto $O(|S|^2)$. Por otro lado, en caso de que todos los elementos de S sean de un mismo valor, ocurrirá el mejor caso $\Omega(|S|)$.

Invariante: En la primera iteración i, el elemento U_i es el elemento con mas ocurrencias.

1. **Inicio:** $i \leftarrow 1$, la secuencia no esta vacia.
2. **Iteración:** $1 \leq i < |S|$, en cada iteración el contador esta aumentando si se cumple que $S_j = U_i$
3. **Terminación:** $contador > \frac{|S|}{2}$, se cumple que es una secuencia mayoritaria.

2.3. Algoritmo por dividir y vencer

Algoritmo 2 Contar frecuencia

```
function frecuencia(S,l,r,m)
    count ← 0
    for i ← 1 to r, i ← i + 1 do
        if S_i = m then
            count ← count + 1
        end if
    end for
    return count
end function
```

Algoritmo 3 Secuencia mayoritaria recursivo

```
procedure MAYORITARIO(S,l,r)
    if l = r then
        return S_l
    end if
    mid ← (r − l)/2 + 1
    lm ← MAYORITARIO(S, l, mid)
    rm ← MAYORITARIO(S, mid + 1, r)
    if lm = rm then
        return lm
    end if
    lc ← FRECUENCIA(S, l, r, lm)
    rc ← FRECUENCIA(S, l, r, rm)
    if lc > rc then
        return lm
    else
        return rm
    end if
end procedure
```

Complejidad: Relación de recurrencia:

1. $T(n) = 2T(n/2) + O(n), cuando\ n > 1$
2. $T(n) = O(1), cuando\ n = 1$

Cuando aplicamos el teorema maestro obtenemos que para el peor caso contamos con $O(nlogn)$.

Invariante: Durante los llamados recursivos $lm = rm$

1. **Inicio:** $l! = r$, se buscan los mayoritarios en diferentes direcciones.
2. **Iteración:** Comparar los mayoritarios $l- > m \wedge m- > r$ y calcular las ocurrencias de cada uno.
3. **Terminación:** Al final, con el valor retornado se procede a validar si $r \geq \lceil \frac{|S|}{2} \rceil$