Taller 1

Pontificia Universidad Javeriana

Nicolas Lopez Fernandez

Agosto 1 2019

0.1. Descripcion y formalizacion del problema

El problema, informalmente, se define como encontrar el dato minimo y dato maximo de un arreglo rotado.

Asi, para el arreglo S=[67,89,90,23,25,34,50,60], el minimo es 23 y el maximo es 90.

Formalmente, se dice que: Dada una secuenca rotada S de elementos $a_i \in \mathbb{T}$, donde se defina la relacion de orden parcial !!!!MENOR!!!! para el minimo y !!!!MAYOR!!!! para el maximo, informar los numeros b,$c \in S$ donde b cumple con la relacion de orden parcial !!!!MENOR!!!! y c cumple con la relacion de orden parcial !!!!MAYOR!!!! de toda la secuencia S. Ahora, la definicion del contrato seria:

- **Entradas**: Una secuencia rotada S de n numeros: $S = \langle a_{n-1}, a_n, a_1, a_2, \cdots, a_{n-2} \rangle$ donde $a_i \in \mathbb{T}$
- **Salidas**: Dos resultados $b = a_1$ y $c = a_n$ donde b y c son el menor y mayor dato de la secuencia rotada S

0.2. Fuerza bruta

Algorithm 1 Algoritmo maximo y minimo por fuerza bruta.

```
1: procedure MaxMinRotadoBF(S)
2:     maxi ← 0
3:     mini ← 0
4:     for i ← 1 to |S| do
5:         if S[i] ≥ maxi then
6:             maxi ← S[i]
7:         end if
8:     end for
9:     mini ← maxi
10:    for i ← 1 to |S| do
11:        if S[i] ≤ mini then
12:            mini ← S[i]
13:        end if
14:    end for
15:    return [mini, maxi]
16: end procedure
```

0.3. Dividir y vencer

Algorithm 2 Algoritmo Maximo y minimo por dividir y vencer.

```
    procedure MMRotado(S, l, h)
2:      if h ≤ l then
            return [0, −1]
4:      else if h == l then
            return [S[l], S[h]]
6:      else if h == 1 then
            return [S[0], S[1]]
8:      else if S[0] > S[1] then
            return [S[0], S[1]]
10:     else
            if h == 2 then
12:             S.append(|S| + 1)
                h ← h + 1
14:         end if
            mitad ← (h + 2)/2
16:         if S[mitad] > S[mitad − 1] then
                aux ← S[mitad : |S|]
18:             return MMRotado(aux, 0, |S|)
            end if
20:         if S[mitad] < S[mitad + 1] then
                aux = S[0 : mitad + 1]
22:             return MMRotado(aux, 0, mitad)
            end if
24:     end if
    end procedure
```

Algorithm 3 Algoritmo Maximo y minimo por dividir y vencer.

```
    procedure MaxMinRotadoDV(S) return MMRotado(S, 0, |S|)
    end procedure
```

0.4. Invariantes

El algoritmo de "Fuerza Bruta" tiene como invariante: En cada ciclo de busqueda se encuentra o el maximo o el minimo

El algoritmo de "Dividir-y-vender" tiene como invariante: El menor y el mayor siempre esta en la secuencia que se esta dividiendo

0.5. Analisis de complejidad

En el algoritmo de "Fuerza Bruta", resulta evidente por inspeccion de codigo que su orden de complejidad es $0(n^3)$

Para la version "Dividir-y-vencer", se tiene la ecuacion de recurrencia:

$$T(n) = \begin{cases} O(1) & ; \quad b \geq e \\ T\left(\frac{n}{2}\right) + O(n) & ; \quad b < e \end{cases} ; \tag{1}$$

que tiene un orden de complejidad $\Theta(\log n)$, despues de usar el teorema maestro.