Paula Martinez 13866

Algoritmos y Estructura de Datos I

Análisis de Complejidad Teórica

1) *Calcular la cantidad de OE (operaciones elementales) para cada una de las operaciones del TAD secuencia implementada sobre arreglos:*

Access(Array,posicion) =

```
def access(Array,position):    # 3 OE
    return Array[position]     # 2 OE
```

3+2 = **5**

Search(Array,Element) =

```
def search(Array,element):  # 3 OE
    n=len(Array)            # 2 OE
    for i in range(0,n):    # 3 OE + n OE
        if Array[i]==element:   # n OE
            return i            # 2n OE
    return None             # 2 OE
```

3+2+3+n+2n+2= **10 + 3n**

Insert(Array,element,posicion) =

```
def insert(Array,element,position):  # 4 OE
    n=len(Array)                     # 2 OE
    if position>=n:                  # 1 OE
        return None                  # 2 OE
    ArrayR=algo1.Array(n,0)          # 2 OE
    for i in range(0,n):             # 3 OE + n OE
        ArrayR[i]=Array[i]           # n OE
    for i in range (position,n-1):   # 3 OE + n OE
        ArrayR[i+1]=Array[i]         # n OE
    ArrayR[position]=element         # 1 OE
    for i in range(0,n):             # 3 OE + n OE
        Array[i]=ArrayR[i]           # n OE
    return position                  # 2 OE
```

4+2+1+2+2+3+n+3+n+1+3+n+2 = **23 + 3n**

Delete(Array,element) =

```
def delete(Array,element):          # 3 OE
    position=search(Array,element)  # 4 OE
    if position==None:              # 1 OE
        return None                 # 2 OE
    n=len(Array)                    # 2 OE
    ArrayR=algo1.Array(n,0)         # 2 OE
    for i in range(0,position):     # 3 OE + n OE
        ArrayR[i]=Array[i]          # n OE
    for i in range(position,n-1):   # 3 OE + n OE
        ArrayR[i]=Array[i+1]        # n OE
    for i in range(0,n):            # 3 OE + n OE
        Array[i]=ArrayR[i]          # n OE
    return position                 # 2 OE
```

3+4+1+2+2+2+3+n+3+n+3+n+2 = **25 + 3n**

2) *Calcular el orden de complejidad O(f) para cada una de las operaciones del ejercicio 1.*

Access(Array,posicion) = **O(1)**

Search(Array,Element) = **O(n)**

Insert(Array,element,posicion) = **O(n)**

Delete(Array,element) = **O(n)**

3) *Calcular el orden de complejidad O(f) para los siguientes códigos:*

Código 1:

```
if a>b:
  c=a+b
else:
  for d in range(1,10):
    c=a+b*d
```

Orden de complejidad = **O(1)**

Código 2:

```
a=1
while a<n:
  a=a+1
```

Orden de complejidad = **O(n)**

Código 3:

```
for i in range(1,n):
        j=0
         while j<i:
                a=a*(1+j)
                j=j+1
```

Orden de complejidad = **O(n^2)**

Código 4:

```
for a in range(1,n):
    for b in range(a,n):
        if L[a]==L[b]:
            delete(L,L[b])
```

Orden de complejidad = **O(n^3)**