Thomas Freund
A00831997

Análisis Actividad 1.2

```
99    //cambia un valor almazenado por otro en otro indice
100   void swap(int a[], int i, int j){
101       int aux;
102       aux = a[i];
103       a[i] = a[j];
104       a[j] = aux;
105   }
106
```

Linea	Comp	Caso
100	1	C1
101	1	C2
103	1	C3
104	1	C4

$$T(n) = C1 * 1 + C2 * 1 + C3 * 1 + C4 * 1$$

$$T(n) = c1 + c2 + c3 + c4 = C = 1$$

$$Complejidad: O\ (1)$$

```
170  void ordenaIntercambio(int arreglo[], int n){
171      int i, j, aux;
172      for (i=0 ; i <= n-2 ; i++){
173          for (j=i+1 ; j <= n-1 ; j++){
174              if (arreglo[j]<arreglo[i]){
175                  swap(arreglo,i,j);
176              }
177          }
178      }
179
180  }
```

Linea	Comp	Caso
170	1	C1
171	2n	C2
172	(N-1) +(x+1)+x	C3
174	3x	C4
175	2x	C5

$$T(n) = C1 * 1 + C2 * (n - 1) + C3\,((N - 1) + (x + 1) + x) + C4 * 3x + C5 * 2x$$

T(n)= 7x+ 3N +2

x = 1+2+3+4...n

x= n(n+1)/2

T(n)= (7/2)n^2+7n/2+ 3N +2

$$T(n) = an^2 + bn + c$$

Dado que se evalúa el peor caso y se lleva al límite donde $\lim_{n \to \infty}$, b y c se vuelven insignificantes.

$$T(n) = an^2$$

$$Complejidad: O\,(n^2)$$

```
108 void ordenaBurbuja(int a[], int n) {
109     int aux;
110     for (int i = 0; i < n - 1; i++) {
111         for (int j = 0; j < n-i-1; j++) {
112             if (a[j] > a[j + 1]) {
113                 aux = a[j+1];
114                 a[j + 1] = a[j];
115                 a[j] = aux;
116             }
117         }
118     }
119 }
120
```

Linea	Comp	Caso
109	1	C1
101	2n	C2
101	(N-1) +(x+1)+x	C3
103	3x	C4
104	2x	C5
103	2x	C6
104	1	C7

$$T(n) = C1 * 1 + C2 * 2 + C3\,((N - 1) + (x + 1) + x) + C4 * 3x + C5 * 2x + C6 * 2x$$

T(n)= 11x+ 3N +2

x = 1+2+3+4...n

x= n(n+1)/2

T(n)= (11/2)n^2+11n/2+ 3N +2

$$T(n) = an^2 + bn + c$$

Dado que se evalúa el peor caso y se lleva al límite donde $\lim_{n \to \infty}$, b y c se vuelven insignificantes.

$$T(n) = an^2$$

$$Complejidad: O\ (n^2)$$

```
156  void ordenaMerge(int a[], int inicio, int fin){
157
158    int mitad;
159    if(inicio < fin){
160
161      mitad = (inicio + (fin - 1)) / 2;
162      ordenaMerge(a,inicio,mitad);
163      ordenaMerge(a,mitad+1,fin);
164      merge(a,inicio,mitad,fin);
165
166    }
167  }
```

Línea	Costo	Repeticiones (peor caso)
158	C1	1
159	C2	1
161	C3	1
162	C4	T(n/2)
163	C5	T(n/2)
164	C6	n

Base case:

$$T(n) = 1,\ if\ n = 1$$
$$T(n) = 2 + 2T(n/2),\ if\ n \geq 1$$

Since this is a recursive function, we need to find a general solution through a pattern.

$$T(n) = n + 2T(n/2)$$
$$= n + n + (2 * 2T(n/2/2))$$
$$= 2n + 4T(n/4)$$
$$= n + 2n + (2 * 4T(n/4/2))$$
$$= 3n + 8T(n/8)$$

...

Solución General

$$T(n) = nk + 2^k T(n/2^k)$$

Usando el caso base.

$$n/2^k = 1$$
$$log_2 n = k$$

Sustituímos k:

$$T(n) = n(log_2 n) + 2^{log_2 n} T(n/2^{log_2 n})$$
$$T(n) = nlog_2 n + n + T(1)$$

$$T(n) = nlog_2 n + n + 1$$

$$Complejidad: O\ (nlog_2 n)$$

```
122 void merge(int a[], int inicio, int mitad, int fin){
123     int i = inicio, j = mitad + 1, k = 0, aux[fin - inicio + 1];
124 
125     while (i <= mitad && j <= fin){
126         if (a[i] < a[j]){
127             aux[k] = a[i];
128             i++;
129         } else {
130             aux[k] = a[j];
131             j++;
132         }
133         k++;
134     }
135 
136     while (i <= mitad){
137         aux[k] = a[i];
138         k++;
139         i++;
140     }
141 
142 
143     while (j <= fin){
144         aux[k] = a[j];
145         k++;
146         j++;
147     }
```

Línea	Costo	Repeticiones (peor caso)
123	C1	1
125	C2	n
126	C3	1
127	C4	1
128	C5	1
130	C6	1
131	C7	1
136	C8	n
137	C9	1
138	C10	1
139	C11	1

$$T(n) = C1 + C2 + C3(n + 1) + C4(n) + C5(n) + C6(n) + C7 + C8$$
$$T(n) = C1 + C2 + C3 * n + C3 + C4 * n + C5 * n + C6 * n + C7 + C8$$
$$T(n) = (C3 + C4 + C5 + C6)n + (C1 + C2 + C3 + C7 + C8)$$
Complejidad: **O(n)**

```
182 long int busqSecuencial(int arreglo[], int n, int dato){
183     for(int i = 0; i < n; i++){
184         if(arreglo[i] == dato){
185             return i;
186         }
187     }
188     return -1;
189 }
```

Linea	Comp	Caso
183	n	C1
184	1	C2
185	1	C3
188	1	C4

$$T(n) = (c1)n + (c2 + c3 + c4)\ donde\ a = c1,\ b = c1 + c4 + c3$$

$$T(n) = an + b$$

Dado que se evalúa el peor caso y se lleva al límite donde $\lim_{n\to\infty}$, b se vuelve insignificante.

$$T(n) = an$$
$$Complejidad: O\ (an) = O\ (n)$$

```
192 long int busqBinaria(int arreglo[], int min, int max, int dato){
193     int key;
194     key = (min+max)/2;
195 
196         if (dato == arreglo[key]){
197             return key;
198     } else if (dato < arreglo[key]){
199         max = key - 1;
200     } else {
201         min = key + 1;
202     }
203     return busqBinaria(arreglo, min, max, dato);
204 }
```

Linea	Comp	Caso
202	1	C1
204	1	C2
205	1	C3
206	1	C4

207	1	C5
208	1	C6
209	1	C7
211	1+T(n/2)	C8

T(n) = C1 +C2+ C3+ C4+ C5+ C6+ C7 + C8T(n/2) + C8

Where all the C# add up to a constant.

Cases for C8

$$T(n) = 1, if\ n = 1$$
$$T(n) = 1 + T(n/2), if\ n > 1$$

Since this is a recursive function, we need to find a general solution through a pattern.

$$T(n) = 1 + T(n/2)$$
$$= 1 + 1 + T(n/2\ /2)$$
$$= 2 + T(n/4)$$
$$= 1 + 2 + T(n/4/2)$$
$$= 3 + T(n/8)$$
$$...$$

General Solution

$$T(n) = k + T(n/2^k)$$

Using the base case.

$$n/2^k = 1$$
$$log_2 n = k$$

We substitute k:

$$T(n) = log_2 n + T(n/2^{log_2 n})$$
$$T(n) = log_2 n + 1$$
$$T(n) = log_2 n + 1$$
$$T(n) = log_2 n$$

Therefore the complexity of the recursion is:

$$O(log_2 n)$$