**Patrick Murphy Cristóbal Heredia**

**ACTIVIDAD EN AULA 2**

**ANÁLISIS ASINTÓTICO DE 4 ALGORITMOS DE COMPARACIÓN**

Esta actividad se realiza durante el horario de clases en parejas. En la medida que vayan avanzando (cada vez que finalicen el trabajo de un código completo), deben presentar su avance a la profesora para ser calificados.

Por favor, inicien la actividad mirando el video de introducción a los algoritmos de ordenamiento.

1. **CÓDIGO 1: Bubble Sort**

Mira los videos sobre Bubble Sort. El pseudocódigo se muestra a continuación:

|  |
| --- |
| **function** F1(A)  n=A.size()  cambio=true  **while** (cambio) **do**  cambio=false  **for** 0 <= i < n-1 **do**  **if** list[i] > list[i+1] **then**  swap(list[i],list[i+1])  cambio=true  **end if**  **end for**  n=n-1  **end while**  **return** A  **end function** |
|  |

¿Cuál es el peor caso de ejecución de este algoritmo? (el caso en el que tiene que ejecutar más líneas de código) Entrega un arreglo de ejemplo de peor caso.

Arreglo inicial

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 9 | 8 | 7 | 6 | 5 | 4 |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |

¿Cuál es el mejor caso? Entrega un arreglo de ejemplo de mejor caso.

Arreglo inicial

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |

Completa las siguientes tablas con el análisis detallado de peor y mejor caso.

|  |  |  |  |
| --- | --- | --- | --- |
| **PEOR CASO** | | | |
|  | **CÓDIGO 1:** A: arreglo de entrada | Costo | Veces |
| 1. | **function** BubbleSort(A) | --- |  |
| 2. | n=A.size() | C1+C2 | 1 |
| 3. | cambio=true | C2 | 1 |
| 4. | **while** (cambio) **do** | C3 | N |
| 5. | cambio=false | C2 | N |
| 6. | **for** 0 <= i < n-1 **do** | C3+(C2+C4) | (N(N+1)/2) |
| 7. | **if** list[i] > list[i+1] **then** | C3+C4 | (N(N+1)/2) |
| 8. | aux=list[i] | C2 | (N(N+1)/2) |
| 9. | list[i]=list[i+1] | C2+C4 | (N(N+1)/2) |
| 10. | list[i]=aux | C2 | (N(N+1)/2) |
| 11. | cambio=true | C2 | (N(N+1)/2) |
| 12. | **end if** | --- |  |
| 13. | **end for** | --- |  |
| 14. | n=n-1 | C4+C2 | N |
| 15. | **end while** | --- |  |
| 16. | **return** A | C5 | 1 |
| 17. | **end function** | --- |  |
| **Expresión resumida de peor caso: C6 N2 + C7 N + C8** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **MEJOR CASO** | | | |
|  | **CÓDIGO 1:** A: arreglo de entrada | Costo | Veces |
| 1. | **function** BubbleSort(A) | --- |  |
| 2. | n=A.size() | C1 + C2 | 1 |
| 3. | cambio=true | C2 | 1 |
| 4. | **while** (cambio) **do** | C3 | 1 |
| 5. | cambio=false | C2 | 1 |
| 6. | **for** 0 <= i < n-1 **do** | C3+(C2+C4) | N |
| 7. | **if** list[i] > list[i+1] **then** | C3+C4 | N |
| 8. | aux=list[i] | ---- | --- |
| 9. | list[i]=list[i+1] | ---- | --- |
| 10. | list[i]=aux | ---- | --- |
| 11. | cambio=true | ---- | --- |
| 12. | **end if** | ---- | --- |
| 13. | **end for** | ---- | --- |
| 14. | n=n-1 | C2 + C4 | 1 |
| 15. | **end while** |  | --- |
| 16. | **return** A | C5 | --- |
| 17. | **end function** |  | --- |
| **Expresión resumida de mejor caso: C6 N + C7** | | | |

Agrega aquí dos gráficos hechos en Excel o alguna aplicación similar donde se muestre que las funciones que han listado para O() y Ω() efectivamente son límites superiores e inferiores, según la definición entregada en el video:

T(N) es O(g(N)) si a partir de algún valor k, T(N) ≤ C \* g(N)

T(N) es Ω (g(N)) si a partir de algún valor k, T(N) ≥ C \* g(N)

|  |  |
| --- | --- |
| T(N) ≤ C \* g(N) | T(N) ≥ C \* g(N) |
|  |  |

En base a los gráficos anteriores, completa la siguiente tabla:

|  |  |
| --- | --- |
| **Análisis asintótico para CÓDIGO 1** | |
| O() | O(N2); O(N3) |
| Θ() | Θ(N2) |
| Ω() | Ω(N2); Ω(N\*log2N); Ω(N) |

**b) CÓDIGO 2: Insertion Sort**

Mira los videos sobre Insertion Sort. El pseudocódigo se muestra a continuación:

|  |
| --- |
| **function** InsertionSort(A)  n=A.size()  **for** 1 <= j <= n-1 **do**  ins=A[j]  i=j-1  **while** (ins<A[i] and i>=0)  A[i+1]=A[i]  i --  **end while**  A[i+1]=ins  **end for**  **end function** |

¿Cuál es el peor caso de ejecución de este algoritmo? (el caso en el que tiene que ejecutar más líneas de código) Entrega un arreglo de ejemplo de peor caso.

Arreglo inicial

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 9 | 8 | 7 | 6 | 5 | 4 |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |

¿Cuál es el mejor caso? Entrega un arreglo de ejemplo de mejor caso.

Arreglo inicial

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 5 | 6 | 7 | 8 | 9 | 11 |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |

Completa las siguientes tablas con el análisis detallado de peor y mejor caso.

|  |  |  |  |
| --- | --- | --- | --- |
| **PEOR CASO** | | | |
|  | **CÓDIGO 2:** A: arreglo de entrada | Costo | Veces |
| 1. | **function** InsertionSort(A) | --- |  |
| 2. | n=A.size() | C1 + C2 | 1 |
| 3. | **for** 1 <= j <= n-1 **do** | C3+C4+C2 | N-1 |
| 4. | ins=A[j] | C2 | N-1 |
| 5. | i=j-1 | C4+C2 | N-1 |
| 6. | **while** (ins<A[i] and i>=0) | C3+C3 | N(N+1)/2 |
| 7. | A[i+1]=A[i] | C4+C2 | N(N+1)/2 |
| 8. | i -- | C4+C2 | N(N+1)/2 |
| 9. | **end while** | --- | --- |
| 10. | A[i+1]=ins | C4+C2 | N-1 |
| 11. | **end for** | --- | --- |
| 12. | **end function** | --- | --- |
| **Expresión resumida de peor caso: C5 N2 +C6 N + C7** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **MEJOR CASO** | | | |
|  | **CÓDIGO 2:** A: arreglo de entrada | Costo | Veces |
| 1. | **function** InsertionSort(A) | --- |  |
| 2. | n=A.size() | C1 + C2 | N-1 |
| 3. | **for** 1 <= j <= n-1 **do** | C3+C4+C2 | N-1 |
| 4. | ins=A[j] | C2 | N-1 |
| 5. | i=j-1 | C4+C2 | N-1 |
| 6. | **while** (ins<A[i] and i>=0) | C3+C3 | N-1 |
| 7. | A[i+1]=A[i] | --- | --- |
| 8. | i -- | --- | --- |
| 9. | **end while** | --- | --- |
| 10. | A[i+1]=ins | C4+C2 | N-1 |
| 11. | **end for** | --- | --- |
| 12. | **end function** | --- | --- |
| **Expresión resumida de mejor caso: C5 N + C6** | | | |

Agrega aquí dos gráficos hechos en Excel o alguna aplicación similar donde se muestre que las funciones que han listado para O() y Ω() efectivamente son límites superiores e inferiores, según la definición entregada en el video:

T(N) es O(g(N)) si a partir de algún valor k, T(N) ≤ C \* g(N)

T(N) es Ω (g(N)) si a partir de algún valor k, T(N) ≥ C \* g(N)

|  |  |
| --- | --- |
| T(N) ≤ C \* g(N) | T(N) ≥ C \* g(N) |
|  |  |

En base a los gráficos anteriores, completa la siguiente tabla:

|  |  |
| --- | --- |
| **Análisis asintótico para CÓDIGO 2** | |
| O() | O(N2); O(N3) |
| Θ() | Θ(N2) |
| Ω() | Ω(N2); Ω(N\*log2N); Ω(N) |

**c) Código 3: Selection Sort**

Mira los videos sobre Selection Sort. El pseudocódigo se muestra a continuación:

|  |
| --- |
| **function** SelectionSort(A)  n=A.size()  **for** 0 ≤ i ≤ n-2 **do**  min=i  **for** (i+1) ≤ j ≤ n-1 **do**  **if** (A[j]<A[min]) **then**  min=j  **end if**  **end for**  swap(A[i],A[min])  **end for**  **end function** |

¿Cuál es el peor caso de ejecución de este algoritmo? (el caso en el que tiene que ejecutar más líneas de código). Entrega un arreglo de ejemplo de peor caso.

Arreglo inicial

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 98 | 56 | 47 | 33 | 28 | 12 | 1 |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |

¿Cuál es el mejor caso? Entrega un arreglo de ejemplo de mejor caso.

Arreglo inicial

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |

Completa las siguientes tablas con el análisis detallado de peor y mejor caso.

|  |  |  |  |
| --- | --- | --- | --- |
| **PEOR CASO** | | | |
|  | **CÓDIGO 3:** A: arreglo de entrada | Costo | Veces |
| 1. | **function** SelectionSort(A) | --- | --- |
| 2. | n=A.size() | C1 | 1 |
| 3. | **for** 0 ≤ i ≤ n-2 **do** | C2+C3+C1 | N-1 |
| 4. | min=i | C1 | N-1 |
| 5. | **for** (i+1) ≤ j ≤ n-1 **do** | C2+C3+C1 | N(N+1)/2 |
| 6. | **if** (A[j]<A[min]) **then** | C2 | N(N+1)/2 |
| 7. | min=j | C1 | N(N+1)/2 |
| 8. | **end if** | --- | --- |
| 9. | **end for** | --- | --- |
| 10. | swap(A[i],A[min]) | C4 | N-1 |
| 11. | **end for** | --- | --- |
| 12. | **end function** | --- | --- |
| **Expresión resumida de peor caso: C5 N2+ C6 N + C7** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **MEJOR CASO** | | | |
|  | **CÓDIGO 3:** A: arreglo de entrada | Costo | Veces |
| 1. | **function** SelectionSort(A) | --- | --- |
| 2. | n=A.size() | C1 | 1 |
| 3. | **for** 0 ≤ i ≤ n-2 **do** | C2+C3+C1 | N-1 |
| 4. | min=i | C1 | N-1 |
| 5. | **for** (i+1) ≤ j ≤ n-1 **do** | C2+C3+C1 | N(N+1)/2 |
| 6. | **if** (A[j]<A[min]) **then** | C2 | N(N+1)/2 |
| 7. | min=j | C1 | --- |
| 8. | **end if** | --- | --- |
| 9. | **end for** | --- | --- |
| 10. | swap(A[i],A[min]) | C4 | N-1 |
| 11. | **end for** | --- | --- |
| 12. | **end function** | --- | --- |
| **Expresión resumida de mejor caso: C5 N2 + C6 N + C7** | | | |

Agrega aquí dos gráficos hechos en Excel o alguna aplicación similar donde se muestre que las funciones que han listado para O() y Ω() efectivamente son límites superiores e inferiores, según la definición entregada en el video:

T(N) es O(g(N)) si a partir de algún valor k, T(N) ≤ C \* g(N)

T(N) es Ω (g(N)) si a partir de algún valor k, T(N) ≥ C \* g(N)

|  |  |
| --- | --- |
| T(N) ≤ C \* g(N) | T(N) ≥ C \* g(N) |
|  |  |

En base a los gráficos anteriores, completa la siguiente tabla:

|  |  |
| --- | --- |
| **Análisis asintótico para CÓDIGO 3** | |
| O() | O(N2); O(N3) |
| Θ() | Θ(N2) |
| Ω() | Ω(N2); Ω(NLog(N)); Ω(N) |

**d) Código 4: Counting Sort**

Mira los videos sobre Counting Sort. El pseudocódigo se muestra a continuación:

|  |
| --- |
| **function** counting-sort(A,k)  C ← new array(k+1)  R ← new array(length(A))  pos ← 0  **for** 0 ≤ j < length(A) **do**  C[A[j]] ← C[A[j]] + 1  **end for**  **for** 0 < i < (k+1) **do**  **for** pos ≤ r < pos+C[i] **do**  R[r]=i  **end for**  pos=r  **end for**  **return** R  **end function** |

¿Cuál es el peor caso de ejecución de este algoritmo? (el caso en el que tiene que ejecutar más líneas de código) Entrega un arreglo de ejemplo de peor caso.

Arreglo inicial

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 20000000 | 15000000 | 1000000 | 15151551 | 2222 | 144159 | 2334559 |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |

¿Cuál es el mejor caso? Entrega un arreglo de ejemplo de mejor caso.

Arreglo inicial

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |

Completa las siguientes tablas con el análisis detallado de peor y mejor caso.

|  |  |  |  |
| --- | --- | --- | --- |
| **PEOR CASO** | | | |
|  | **CÓDIGO 4:** A: arreglo de entrada; k: máximo valor en A | Costo | Veces |
| 1. | **function** CountingSort(A,k) | --- |  |
| 2. | C ← new array(k+1) | C1 | 1 |
| 3. | R ← new array(length(A)) | C1 | 1 |
| 4. | pos ← 0 | C1 | 1 |
| 5. | **for** 0 ≤ j < length(A) **do** | C1+C2+C3 | N |
| 6. | C[A[j]] ← C[A[j]] + 1 | C1+C3 | N |
| 7. | **end for** | --- |  |
| 8. | **for** 0 < i < (k+1) **do** | C1+C2+C3 | K |
| 9. | **for** pos ≤ r < pos+C[i] **do** | C1+C2+C3 | N |
| 10. | R[r]=i | C1 | N |
| 11. | **end for** | --- |  |
| 12. | pos=r | C1 | Depende de la cantidad de datos distintos que hayan |
| 13. | **end for** | --- |  |
| 14. | **return** R | C4 | 1 |
| 15. | **end function** | --- |  |
| **Expresión resumida de peor caso: C6 K + C7 N** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **MEJOR CASO** | | | |
|  | **CÓDIGO 4:** A: arreglo de entrada; k: máximo valor en A | Costo | Veces |
| 1. | **function** CountingSort(A,k) | --- | --- |
| 2. | C ← new array(k+1) | C1 | 1 |
| 3. | R ← new array(length(A)) | C1 | 1 |
| 4. | pos ← 0 | C1 | 1 |
| 5. | **for** 0 ≤ j < length(A) **do** | C1+C2+C3 | N |
| 6. | C[A[j]] ← C[A[j]] + 1 | C1+C3 | N |
| 7. | **end for** | --- |  |
| 8. | **for** 0 < i < (k+1) **do** | C1+C2+C3 | 0 |
| 9. | **for** pos ≤ r < pos+C[i] **do** | C1+C2+C3 | 0 |
| 10. | R[r]=i | C1 | 0 |
| 11. | **end for** | --- |  |
| 12. | pos=r | C1 | 0 |
| 13. | **end for** | --- |  |
| 14. | **return** R | C4 | 1 |
| 15. | **end function** | --- |  |
| **Expresión resumida de mejor caso: C6 N** | | | |

Agrega aquí dos gráficos hechos en Excel o alguna aplicación similar donde se muestre que las funciones que han listado para O() y Ω() efectivamente son límites superiores e inferiores, según la definición entregada en el video:

T(N) es O(g(N)) si a partir de algún valor k, T(N) ≤ C \* g(N)

T(N) es Ω (g(N)) si a partir de algún valor k, T(N) ≥ C \* g(N)

|  |  |
| --- | --- |
| T(N) ≤ C \* g(N) | T(N) ≥ C \* g(N) |
|  |  |

En base a los gráficos anteriores, completa la siguiente tabla:

|  |  |
| --- | --- |
| **Análisis asintótico para CÓDIGO 4** | |
| O() | O(N); O(K); O(K + N); O(NLogN); O(N2) |
| Θ() | Θ(N) |
| Ω() | Ω(N); Ω(LOG N) ; Ω(1) |