Análisis de Algoritmos y Estructura de Datos

Refuerzo/ejercicios
Algoritmos de búsqueda y ordenamiento

Prof. Violeta Chang C

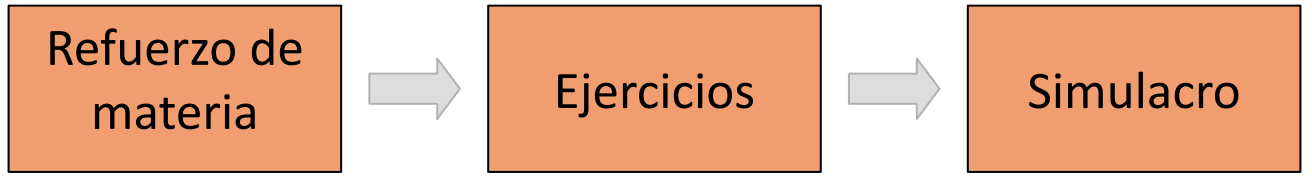
Semestre 2 – 2023

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Algoritmos de búsqueda y ordenamiento

- **Contenidos**:
  - Algoritmos de búsqueda
  - Algoritmos de ordenamiento
- **Objetivos**:
  - Explicar algoritmos de búsqueda y ordenamiento
  - Realizar trazas de algoritmos de búsqueda y ordenamiento
  - Utilizar algoritmos de búsqueda y ordenamiento en la resolución de problemas

Ruta de sesión



Algoritmos de búsqueda

Conjunto de datos del mismo tipo

Valor exacto que se busca

V/F si lo encuentra o posición en conjunto de datos de valor encontrado

Búsqueda lineal/secuencial

```
busquedaSecuencial(arreglo,datoBuscado):num
    n←tamaño(arreglo)
    Para i←1 hasta n paso 1
        Si arreglo(i)=datoBuscado entonces
            devolver(i)
    devolver(0)
```

O(1)
O(n)
O(1)

O(n)

Búsqueda binaria

```
busquedaBinaria(arreglo,inicio, final,datoBuscado):num
    centro←piso((inicio+final)/2)
    Si arreglo(centro)=datoBuscado entonces
        devolver(centro)
    sino
        Si arreglo(centro)>datoBuscado entonces
            devolver(busquedaBinaria(arreglo,inicio,centro-1,datoBuscado))
        sino
            devolver(busquedaBinaria(arreglo,centro+1,final,datoBuscado))
```

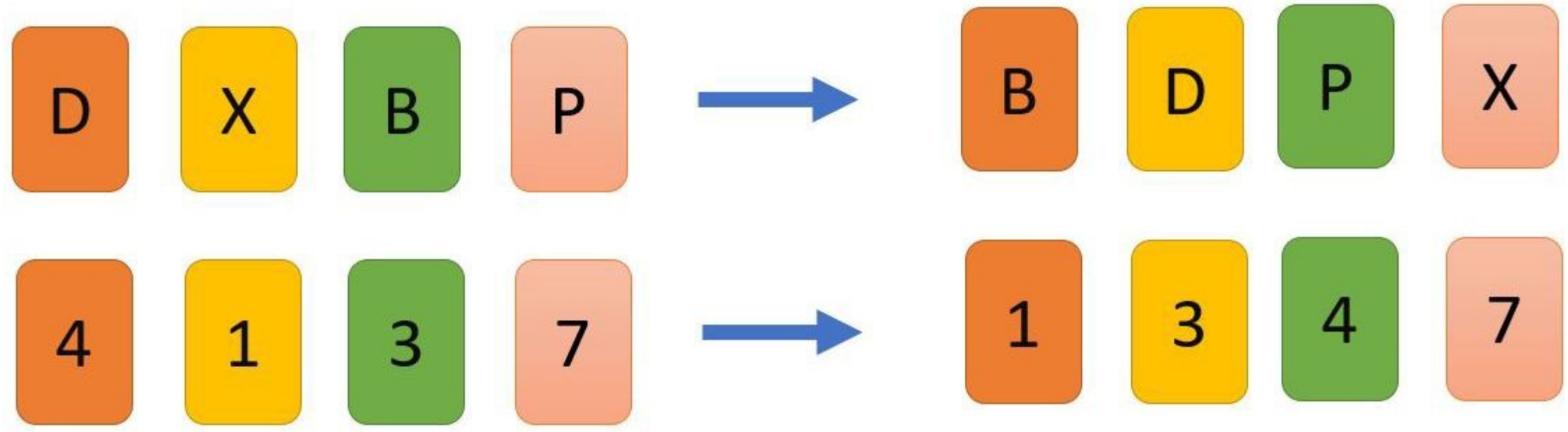
$$T(n) = T(n/2) + O(1)$$

- $T(n) = aT\left(\frac{n}{b}\right) + O(n^k), si\ n \geq b$

$$O(\ ) = \begin{cases} n^k & si\ a < b^k \\ n^k \log_b n & si\ a = b^k \\ n^{\log_b a} & si\ a > b^k \end{cases}$$

$$\Rightarrow O(log_2 n)$$

Algoritmos de ordenamiento



Ordenamiento por selección

```
ordenamientoSeleccion(arregloA):arreglo
    n←tamaño(arregloA)
    Para i←1 hasta n-1 paso -1
        indiceMejor←i
        valorMejor←arregloA(i)
        Para j←i+1 hasta n paso 1
            Si arregloA(j)<valorMejor entonces
                indiceMejor←j
                valorMejor←arregloA(j)
        Si indiceMejor<>i entonces
            arregloA←intercambiar(arregloA,i,indiceMejor)
    devolver(arregloA)
```

O(1)

$O(n^2)$

O(1)

$\Rightarrow O(n^2)$

Ordenamiento por inserción

```
ordenamientoInsercion(arregloA):arreglo
    n←tamaño(arregloA)
    Para i←2 hasta n paso 1
        j←i
        Mientras j>=2 Y arregloA(j)<arregloA(j-1) hacer
            arregloA←intercambiar(arregloA,j,j-1)
            j←j-1
    devolver(arregloA)
```

O(1)

O(n^2)

O(1)

$\Rightarrow O(n^2)$

Ordenamiento por burbuja

```
ordenamientoBurbuja(arregloA):arreglo
    n←tamaño(arregloA)
    Para i←n hasta 1 paso -1
        Para j←1 hasta i-1 paso 1
            Si arregloA(j)>arregloA(j+1) entonces
                arregloA←intercambiar(arregloA,j,j+1)
    devolver(arregloA)
```

O(1)

$O(n^2)$

O(1)

$\Rightarrow O(n^2)$

Ordenamiento rápido (Quicksort)

```
quickSort(arregloA,inicio,fin)
    Si inicio<fin entonces
        pivote←particiona(arregloA,inicio,fin)
        quickSort(arregloA,inicio,pivote-1)
        quickSort(arregloA,pivote+1,fin)
```

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^k), si\ n \geq b$$

$$O(\ ) = \begin{cases} n^k & si\ a < otro\ caso \\ n^k \log_b n & si\ a = b^k \\ n^{\log_b a} & si\ a > b^k \end{cases}$$

$$T(n) = 2T(n/2) + O(n)$$

⇒ O(nlog_2n) …en caso ***promedio***

Ejercicio 1 - en parejas (20 minutos)

Se tiene una secuencia de 2n números enteros no necesariamente distintos. Escribir un algoritmo en seudocódigo de orden O(n log n) que reciba una secuencia de 2n números y que muestre aquella partición que maximiza la máxima suma de un par formado por números de la secuencia de entrada

Por ejemplo, se tiene la secuencia es (5,3,1,9). Las particiones posibles son ((5,3),(1,9)), ((5,1),(3,9)), y ((5,9),(3,1)). La suma por pares de dichas particiones son (8,10), (6,12), y (14,4). Por lo tanto, la tercera partición ((5,9),(3,1)) tiene 14 como su suma máxima, la cual es la máxima de las sumas máximas respecto a las otras dos particiones.

Calcular el orden de complejidad del algoritmo propuesto, y justificar cómo se respeta la complejidad solicitada

Ejercicio 2 - en parejas (15 minutos)

El rango establece la proximidad de los datos de una colección, restando al dato mayor el dato menor. Por ejemplo, el rango de la colección {4,6,2,4,3,1} es 5. Escribir un algoritmo en seudocódigo que calcule y retorne el rango de una colección de n números.

En base a los antecedentes anteriores, para responder esta pregunta se pide:

- Escribir un algoritmo eficiente para resolver el problema planteado
- Calcular la complejidad del algoritmo propuesto
- Justificar la complejidad obtenida

Simulacro (PEP1 1/2023)

- **Instrucciones**:
  - Resolver esta evaluación de manera individual, sin consultar material adicional
  - El tiempo para resolver la evaluación es de 30 minutos
  - Responder la evaluación escribiendo con letra de tamaño adecuado, legible y ordenado
  - Revisar la pauta de evaluación (30 puntos)

    I. El algoritmo propuesto apunta a resolver el problema planteado (6 puntos)

    II. El algoritmo resuelve correctamente el problema planteado (12 puntos)

    III. El algoritmo está escrito en pseudocódigo ordenado y consistente, en el formato establecido, identificando entradas y salidas (6 puntos)

    IV. Calcula correctamente O() y justifica con argumentos correctos (6 puntos)

Simulacro (PEP1 1/2023)

Existe la función voltear(A,i) que revierte el arreglo A desde la posición 1 hasta i. Considerando esta operación, escribir un algoritmo para ordenar decrecientemente un arreglo A de n enteros desordenados. Solo se puede usar la operación voltear y no se pueden ocupar arreglos auxiliares. La idea es ocupar la menor cantidad de volteretas para ordenar el arreglo. ¿Cuál es la complejidad del algoritmo propuesto?

Simulacro (PEP1 1/2023)

I. El algoritmo propuesto apunta a resolver el problema planteado	*6*	*El algoritmo propuesto apunta a resolver el problema planteado*	*6*
		El algoritmo propuesto apunta a resolver el problema planteado, pero la idea no es totalmente correcta.	*3*
		El algoritmo propuesto NO apunta a resolver el problema	*0*
II. El algoritmo resuelve correctamente el problema planteado.	*12*	*El algoritmo resuelve correctamente el problema planteado*	*12*
		El algoritmo tiene errores leves, o bien resuelve al menos el 75% del problema sin errores	*9*
		El algoritmo tiene errores graves, o bien resuelve al menos el 50% del problema sin errores	*6*
		No resuelve el problema o no cumple el ítem I	*0*
III. El algoritmo está escrito en pseudocódigo ordenado y consistente, en el formato establecido, identificando entradas y salidas.	*6*	*Pseudocódigo ordenado y consistente, en el formato establecido, identificando entradas y salidas*	*6*
		Pseudocódigo ordenado y consistente, en el formato establecido, pero con algunos errores	*3*
		Pseudocódigo no ordenado ni consistente, o sin el formato establecido, o no cumple el ítem I	*0*
IV. Calcula correctamente O() y justifica con argumentos correctos	*6*	*Calcula correctamente O() de algoritmo propuesto y justifica con argumentos correctos*	*6*
		Calcula correctamente O() de algoritmo propuesto pero no justifica con argumentos correctos, o argumentos son incorrectos	*3*
		No calcula correctamente O() de algoritmo propuesto ni justifica con argumentos correctos	*0*

Actividad de cierre

- Ir a menti.com e ingresar código 3544 6768

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Próximas fechas...

- Resumen de la semana:
  - Algoritmos de búsqueda
  - Algoritmos de ordenamiento

~~cátedra~~ – ~~refuerzo~~ – ~~laboratorio~~

- Próxima semana:
  - **PEP1: martes 03/octubre**
  - **Tarea1: miércoles 04/octubre**

- Próxima clase:
  - Punteros y tipo de dato abstracto

U1 - S3