



# Programación Competitiva

**Computer Society**

**Wilmer Arévalo**



# Tabla de Contenido

## Introducción

01

### Presentación

Presentación de CS,  
presentación del líder

02

### Programa General

Programa general de los  
contenidos del curso

03

### Problemas Introductorios

Acercamiento a CP

04

### Más problemas

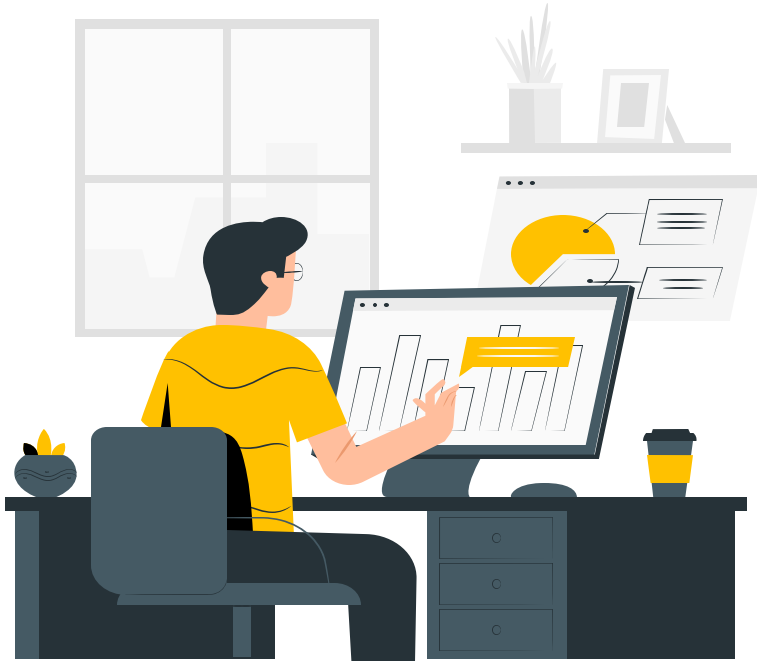
Problemas de CP para  
cambiar perspectivas



# 01

## Presentación

¿Qué es Computer Society?,  
¿Quiénes somos nosotros?





# ¿Quiénes somos?

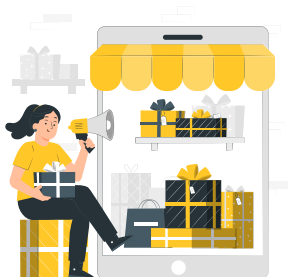
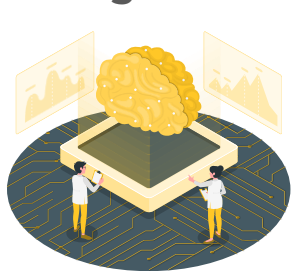


**Computer Society** es un grupo estudiantil de la Universidad de Los Andes y parte de la rama IEEE.

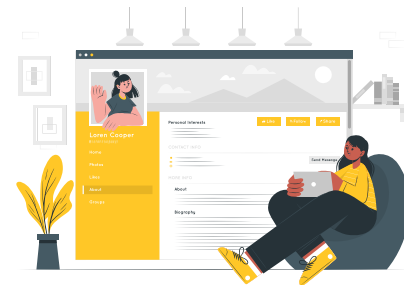
Promueve el **intercambio de conocimientos y experiencias** en ingeniería de sistemas y tecnologías de la información.

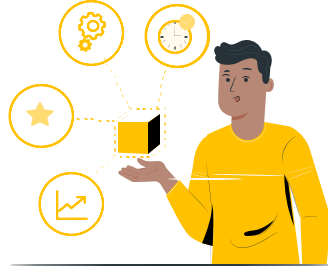


Fomenta la interacción entre sus **miembros y la comunidad en general**.



**Computer Society**  
Unindes





# Programación Competitiva

"Este grupo tiene como objetivo fortalecer los conocimientos en **Programación Competitiva**, mejorando la lógica algorítmica y la resolución de problemas, además de fomentar la participación en competencias y proyectos afines."





## El líder



**Wilmer Arévalo**

Líder del equipo de  
Programación Competitiva

**ArquiSoft**  
Arquitectura  
y Diseño de  
Software

**DSW**  
Desarrollo  
de Software  
en Equipos

**EDA**  
Estructuras  
de Datos y  
Algoritmos



# IEEEEXTREME 2024

## Top 15 Colombia



**Computer Society**  
Unindes





# 02

## Programa General

Programa general de los contenidos del curso







# Programa General



## Greedy Algorithms

Aproximación con algoritmos voraces



## Number Theory

Problemas con teoría de números



## Recursive Algorithms

Aproximación con algoritmos recursivos



## Graphs

Representación, exploración y optimización con grafos



## Backtracking

Aproximación con algoritmos de Backtracking



## Dynamic Programming

Aproximación con algoritmos de programación dinámica



# 03

## Problemas Introdutorios

¡Ahora sí, empecemos!





## Two Sum

Dada una lista de números y un número objetivo, encontrara dos números de la lista, en diferentes posiciones, cuya suma sea igual al número objetivo.

Debe retornar la posición de los números encontrados, o -1 en caso de que no se pueda.

7	2	12	15	9	5	11
---	---	----	----	---	---	----



# ¿Alguna idea?



## Fuerza Bruta

La vieja confiable, probar con todas las posibles configuraciones.



## Memoización

¿Y si aprovechamos una estructura de datos auxiliar para recordar cálculos?



## Noción de Orden

¿Qué pasaría si nos dijeran que la lista está ordenada?



# ¿Alguna idea?

```
def twoSum(array, target):  
    length = len(array)  
    for i in range(length):  
        for j in range(length):  
            sum = array[i] + array[j]  
            if i != j and sum == target:  
                return i, j  
    return -1
```

**Fuerza Bruta**



```
def twoSum(array, target):  
    memo = {}  
    for i in range(len(array)):  
        number = array[i]  
        complement = target - number  
        if complement in memo:  
            return memo[complement], i  
        memo[number] = i  
    return -1
```

**Memoización**



## Próximamente

Retomaremos este tema  
más tarde

**Noción de Orden**





# ICPC Ballons

El ICPC es una competencia de programación en donde se otorga:

- 2 globos si se resuelve un problema por primera vez.
- 1 globo si resuelve un problema que ya había sido resuelto.

Cada problema tiene un id (A-Z). Dada una cadena que represente el orden en el que se resolvieron los problemas, calcule el total de puntos otorgados.

A

B

C

A

7



# ¿Alguna idea?



## Fuerza Bruta

La vieja confiable, probar con todas las posibles configuraciones.



## Memoización

¿Y si aprovechamos una estructura de datos auxiliar para recordar cosas?



## ¿Cuántos caracteres tiene el alfabeto?

Sabemos que el id es una letra del alfabeto, ¿eso nos sirve?



# ¿Alguna idea?

```
def icpcBallons(sequence):
    points = 0
    length = len(sequence)
    for i in range(length):
        a = sequence[i]
        found = False
        for j in range(length):
            b = sequence[j]
            if a == b and not found:
                if j < i:
                    points += 1
                    found = True
        if not found:
            points += 2
    return points
```

Fuerza Bruta



```
def icpcBallons(sequence):
    memo = set()
    points = 0
    for problem in sequence:
        if problem in memo:
            points += 1
        else:
            memo.add(problem)
            points += 2
    return points
```

Memoización



```
def icpcBallons(sequence):
    memo = [0]*26
    points = 0
    for problem in sequence:
        i = ord(problem) - ord('A')
        if memo[i] == 1:
            points += 1
        else:
            memo[i] = 1
            points += 2
    return points
```

¿Cuántos caracteres  
tiene el alfabeto?





# 04

## Más Problemas...

¡Pon a prueba tus habilidades!



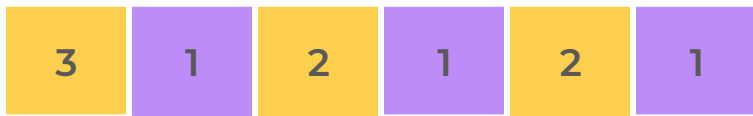


# ACM ICPC

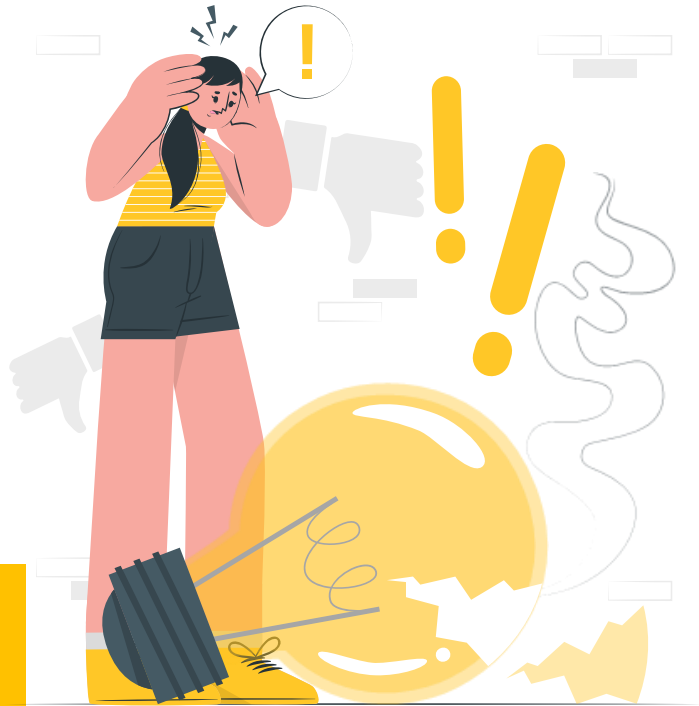
Dados 6 números enteros, tu tarea es determinar si es posible dividirlos en dos grupos de tal manera que la suma de los números de cada grupo sea igual.

- Consideraciones

Resolver este problema con 3 ciclos anidados es sencillo, pero ineficiente. ¿Puedes resolverlo solo con dos ciclos anidados?



# Polygon



Se tiene una matriz  $n \times n$  con cañones en el borde izquierdo y superior, que disparan proyectiles en línea recta hasta chocar con otro proyectil o un borde.

Dada una matriz de 0s y 1s (donde 1 indica un proyectil), se debe verificar si pudo haber sido generada por los disparos o si es inválida.

0	1	0
0	1	1
0	0	0



# Stalin Sort

El Stalin Sort es un algoritmo humorístico que ordena eliminando los elementos fuera de lugar en lugar de reordenarlos. Un arreglo se considera vulnerable si puede ordenarse en orden no creciente aplicando Stalin Sort repetidamente a sus subarreglos. Dado un arreglo de  $n$  enteros, el objetivo es determinar el mínimo número de eliminaciones necesarias para hacerlo vulnerable.



# Perfect Standing



Una competencia de programación tiene 5 problemas (A, B, C, D y E), cada uno con un puntaje específico. Hay 31 combinaciones posibles de soluciones, desde resolver un solo problema hasta todos. El objetivo es listar todas las combinaciones ordenadas de mayor a menor puntaje según los valores dados para cada problema.

A	B	C	D	E
400	500	600	700	800

ABCDE	...	A
-------	-----	---



## Recursos Adicionales





# ¡Gracias!

## Computer Society



**Wilmer Arévalo**

CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), infographics & images by [Freepik](#) and illustrations by [Stories](#)