CURSO DE PROGRAMACIÓN COMPETITIVA
URJC - 2026

CURSO DE PROGRAMACIÓN COMPETITIVA URJC - 2026

Organizadores:

- **Isaac Lozano** (isaac.lozano@urjc.es)
- Sergio Salazar (sergio.salazar@urjc.es)
- Adaya Ruiz (am.ruiz.2020@alumnos.urjc.es)
- Olga Somalo (o.somalo.2021@alumnos.urjc.es)
- Lucas Martín (lucas.martin@urjc.es)
- Iván Penedo (ivan.penedo@urjc.es)
- Alicia Pina (alicia.pina@urjc.es)
- Sara García (sara.garciar@urjc.es)
- Raúl Fauste (raul.fauste@urjc.es)

CURSO DE PROGRAMACIÓN COMPETITIVA
URJC - 2026

¿Quién somos?

URJC Escuela Técnica Superior de Ingeniería Informática

CURSO DE PROGRAMACIÓN COMPETITIVA URJC - 2026

¿En que trabajamos?

Motivación

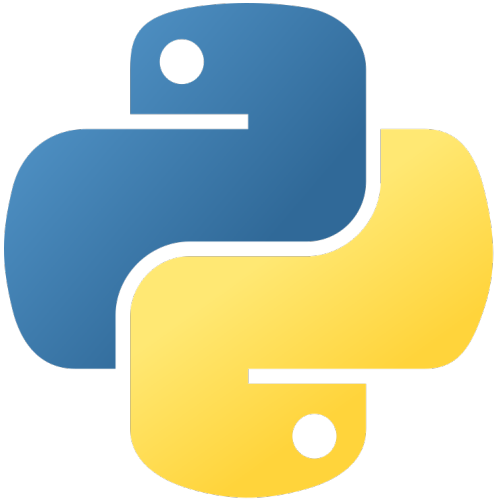
- ¿Qué aprenderás?
  - diseño de algoritmos
  - estructuras de datos
  - nociones de complejidad
  - …aprobar asignaturas!!! (ED, EDA, IP, POO, LP, DAA, P1, P2, AJ, …)

OBJETIVOS EN PROGRAMACIÓN COMPETITIVA

- Resolver los problemas en el menor tiempo posible
- Tener nociones intuitivas:
  - Tipos de problemas, algoritmos...
  - **Complejidad** vs límite de tiempo (eficiencia)
  - Estructuras de datos necesarias
- **Trabajo en equipo** (nombres creativos)
- Representar tu institución, país...

Motivación

- Empresas patrocinadoras
  - cazatalentos
  - concursos internos
  - entrevistas de trabajo



¿EMPRESAS QUE TRABAJÓ/A GENTE QUE COMPLETÓ EL CURSO?

Motivación

- Participación en concursos
  - SWERC ‘26 en ????: ? Equipos **NO TENEMOS EQUIPOS**
    https://urjc-cp.github.io/urjc-cp/swerc.html
  - AdaByron ’26 Madrid – 10 y 11 de abril
    https://ada-byron.es/2026/reg/madrid/
  - AdaByron Nacional – 3 y 4 de julio
    https://ada-byron.es/2026/nac/
  - 12 Uvas -> 31/12
    https://las12uvas.es/

CURSO DE PROGRAMACIÓN COMPETITIVA URJC - 2026

Curso:

- **9 sesiones**
- 6 de febrero - 10 de abril (incluidos)
- Viernes: 17:00 - 19:00
- Aula 111 o Aula 109 - Laboratorios 3

PLANIFICACIÓN DEL CURSO

- **Sesiones teórica y práctica**.
- En la teórica, los docentes explicarán los algoritmos y resolveremos problemas.
- En la práctica, tendremos un juez y se hará un **concurso de prueba** con enunciados cortos para fortalecer lo dado la semana anterior.

PLANIFICACIÓN DEL CURSO

- Bloque 1: Introducción (06/02)
- Bloque 2: Estructuras de Datos (13/02, 20/02)
- Bloque 3: Estructuras de datos II (27/02 , 06/03*)
- Bloque 4: Algoritmos de búsqueda y voraces (13/03, 20/03* y clasificatorio AdaByron)
- Bloque 5: Grafos (27/03)
- Concurso Final (10/04*)

Trataremos conseguiros premios para los concursos ☺

*** concurso por equipos**

CANALES Y FORMULARIO DE EQUIPOS

CP-2026

Formulario

¡¡NO TENGO EQUIPO!!

Los concursos oficiales FUERZAN que sean **3 personas**, en caso de ser menos o más no permiten participar.

Usad la clase para buscar otros compañeros.
Usad los grupos.

Preguntadnos, pero ser 3 para poder participar en cualquier evento.

¿Y MIS CRÉDITOS?

- 9 sesiones - asistir a MÍNIMO 7 de ellas = CRÉDITOS
- Concursos (AdaByron, SWERC) = CRÉDITOS

¿CÓMO SE VALIDA ASISTENCIA?

- **Sesiones teóricas:** Aplicación de la URJC de Asistencia en un momento aleatorio de la clase.
- **Sesiones prácticas:** Mínimo de 1 envío por persona/equipo revisando IP del envío + Aplicación URJC en un momento aleatorio.
- No rellenar el formulario = NO ASISTENCIA

¿Podemos preguntar cualquier duda?

- Sí y no..

TIPOS DE COMPETICIONES

ADA BYRON

swerc regionals 2021 2022

icpc International Collegiate Programming Contest

SWERC

- ACM-ICPC:
  - 5 horas de duración
  - Equipos: 3 personas (1 ordenador)
  - Puntuación: problemas resueltos (0/1)
    - Empates: tiempo + penalizaciones

TIPOS DE COMPETICIONES

Southwestern Europe Regional Contest (SWERC) 2025 — final standings

RANK	TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K	L
1	Participants2025 **Fire Code** The Open University of Israel	11 1320	68 1 try	56 3 tries	78 1 try	5 1 try	31 1 try	21 2 tries	130 2 tries	1 try	263 1 try	62 1 try	299 3 tries	187 1 try
2	**Ctrl+Alt+DelETHe** ETH Zürich	10 910	67 1 try	86 1 try	41 2 tries	8 1 try	14 1 try	52 2 tries	150 1 try		236 2 tries	21 1 try	1 try	175 1 try
3	**Northern Spy** The Open University of Israel	10 934	56 1 try	35 2 tries	74 1 try	3 1 try	23 1 try	21 1 try	181 1 try		271 1 try	82 1 try		128 3 tries
4	**UPC-2** Universitat Politècnica de Catalunya	10 1040	70 1 try	81 2 tries	76 1 try	3 1 try	25 1 try	21 1 try	217 2 tries	5 tries	245 1 try	50 1 try	1 try	132 5 tries
5	**Segfault go BRRRR** Delft University of Technology	10 1243	132 1 try	45 1 try	148 1 try	3 1 try	36 1 try	11 1 try	216 2 tries	1 try	235 1 try	25 2 tries	1 try	292 4 tries
6	**ENS Ulm 3** École Normale Supérieure de Paris	9 816	83 1 try	49 1 try	55 1 try	12 1 try	27 1 try	22 1 try	145 1 try	6 tries		66 1 try		277 5 tries
7	**!generous !atheists** Technion - Israel Institute of Technology	9 851	86 1 try	76 2 tries	106 1 try	4 1 try	21 1 try	40 2 tries	199 1 try		2 tries	60 2 tries		179 2 tries
8	**Team 2** ETH Zürich	9 917	148 1 try	69 3 tries	144 1 try	5 1 try	26 1 try	94 3 tries	76 1 try			36 2 tries		219 1 try
9	**UXT** Ecole Polytechnique	9 921	67 1 try	34 1 try	89 1 try	3 1 try	33 1 try	40 2 tries	176 1 try			84 1 try		275 6 tries
10	**TempName** Tel Aviv University	9 959	30 1 try	65 1 try	150 1 try	33 2 tries	10 1 try	42 1 try	269 2 tries			17 1 try		223 5 tries
11	**UPC-1** Universitat Politècnica de Catalunya	9 1002	149 1 try	51 1 try	170 3 tries	6 1 try	28 1 try	32 2 tries	130 1 try	1 try	14 tries	53 2 tries		263 3 tries
12	**Nababbi** Università di Pisa	9 1048	208 3 tries	71 3 tries	119 2 tries	5 1 try	29 1 try	45 2 tries	141 1 try			56 1 try		254 1 try
13	**ENS Ulm 1** École Normale Supérieure de Paris	8 662	42 1 try	81 2 tries	95 1 try	8 1 try	39 1 try	27 1 try	234 2 tries		1 try	76 2 tries		11 tries
14	**UniBois** University of Bologna	8 683	106 1 try	17 1 try	67 1 try	8 1 try	29 1 try	81 2 tries				118 2 tries		217 1 try

https://swerc.eu/2025/theme/scoreboard/

TIPOS DE COMPETICIONES

- ACM-ICPC (Proceso de selección)
  - Eliminatorias en la **universidad** si hay más de tres equipos
  - Eliminatorias en el conjunto de **países** que forman una región (South-Western Europe)
  - Eliminatorias entre los potenciales candidatos en todo el **continente** (Super regional europeo (Beta)
  - **Final Mundial**

TIPOS DE COMPETICIONES

- Codeforces y Topcoder
  - Concursos muy rápidos y frecuentes
  - Libre para cualquiera
  - Tres o cuatro **divisiones** para novatos y expertos
  - De 95 a 120 minutos de duración
  - Puedes ver y ‘romper’ el código de otros
  - Sistema de puntuación (mientras más tardes en resolver problemas, más te penalizan en puntos)

TIPOS DE COMPETICIONES

- Facebook Hacker Cup y Google Code Jam
  - Evento de gente masiva online
  - Al menos 4 rondas
  - Suele haber ronda de clasificación, 2 rondas de filtro y luego la fase final
  - Dos tipos de evaluación (small y large)
  - El caso small se corrige automáticamente
  - El caso large se corrige al terminar la competición
  - Se permite cualquier tipo de solución (incluso manual ó *hardcodeada*) que permita llegar al output

TIPOS DE COMPETICIONES

- USACO/COCI/IOI
  - Concursos dirigidos a alumnos de bachiller/secundaria
  - ¡NO SON TAN FÁCILES!
  - Son evaluados con sistemas de puntuación (no binario ni penalizando tiempo de solución)
  - Resultados después de la competición
  - Funcionan por temporadas (de noviembre a abril) por ser eliminatorias para el IOI (International Olympiads in Informatics)

CURSO DE PROGRAMACIÓN COMPETITIVA URJC - 2026

Organizadores:

- Isaac Lozano (isaac.lozano@urjc.es)
- Sergio Salazar (sergio.salazar@urjc.es)
- Adaya Ruiz (am.ruiz.2020@alumnos.urjc.es)
- **Olga Somalo** (o.somalo.2021@alumnos.urjc.es)
- Lucas Martín (lucas.martin@urjc.es)
- Iván Penedo (ivan.penedo@urjc.es)
- Alicia Pina (alicia.pina@urjc.es)
- Sara García (sara.garciar@urjc.es)
- Raúl Fauste (raul.fauste@urjc.es)

CARACTERÍSTICAS DE UN PROBLEMA

Enunciado: Se explica el problema con una narración que lo justifica
Análisis del Problema: Se requiere una solución determinista para el problema (siempre encontraremos una solución óptima y válida)
Entrada: Se especifica lo que nuestro programa debe leer
Salida: Se especifica lo que nuestro programa debe mostrar
Ejemplos I/O: Muestras de entrada/salida con el comportamiento esperado para el programa
Límites [Opcionales]: Lo máximo o mínimo en cuanto a variables que nuestro programa debe tomar en cuenta

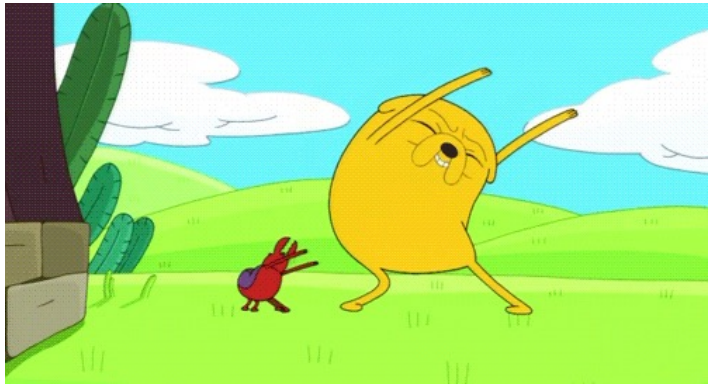
Problema de ejemplo

CARACTERÍSTICAS DE UN PROBLEMA

- Tipos de Lectura:
  - Un caso: Se lee un caso de prueba y a partir de la entrada se genera una salida y termina la ejecución
  - Múltiples casos: Se leen varios casos de pruebas y, dadas múltiples entradas, se generan múltiples salidas
- No hace falta guardar todos los resultados y mostrarlos al final
- ¡Cuidado con reutilizar estructuras de datos!

CARACTERÍSTICAS DE UN PROBLEMA: HE ESCRITO MI SOLUCIÓN ¿Y AHORA QUÉ?

AC



Tu solución es al menos tan buena como la esperada

- Imprimes correctamente todos los casos ocultos
- Tu código tarda menos en ejecutarse que el tiempo límite
- Tu código termina de ejecutarse sin problema

CARACTERÍSTICAS DE UN PROBLEMA: HE ESCRITO MI SOLUCIÓN ¿Y AHORA QUÉ?

WA

WRONG ANSWER

La solución que imprimes en algún caso no coindice con la esperada

CARACTERÍSTICAS DE UN PROBLEMA: HE ESCRITO MI SOLUCIÓN ¿Y AHORA QUÉ?

RTE

RUN TIME ERROR

Tu código muere por algún fallo en tiempo de ejecución

- Dividir entre cero
- Acceder a zonas de memoria no reservada
- Usar objetos en "null"
- Utilizar librerías externas (Python)

CARACTERÍSTICAS DE UN PROBLEMA: HE ESCRITO MI SOLUCIÓN ¿Y AHORA QUÉ?

TLE

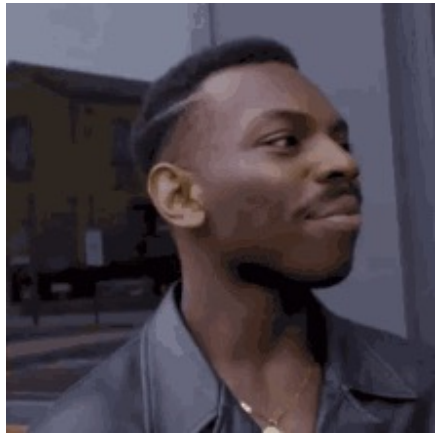
TIME LIMIT EXCEPTION

Tu código tarda demasiado en ejecutarse

- Hay que reducir la complejidad del algoritmo!!!
- Puede esconder otros veredictos
- Seguramente no es la manera adecuada de resolverlo

CARACTERÍSTICAS DE UN PROBLEMA: HE ESCRITO MI SOLUCIÓN ¿Y AHORA QUÉ?

MLE



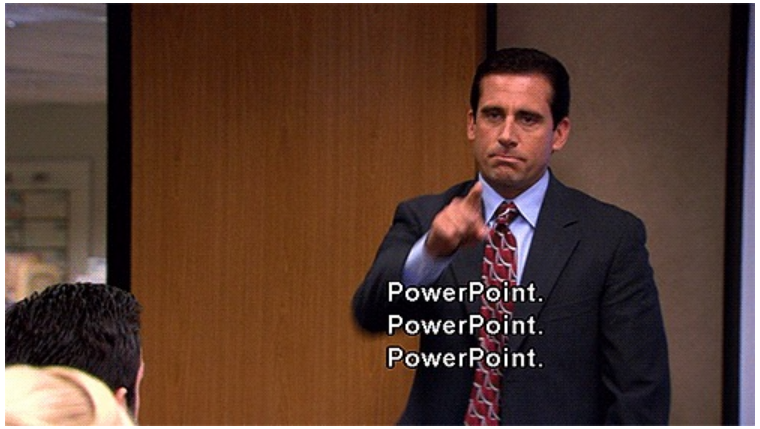
MEMORY LIMIT EXCEPTION

Tu código utiliza más memoria de la reservada

- Hay que reducir la memoria del algoritmo!!!
- Puede esconder otros veredictos

CARACTERÍSTICAS DE UN PROBLEMA: HE ESCRITO MI SOLUCIÓN ¿Y AHORA QUÉ?

PE



PRESENTATION ERROR

Tu código está bien, pero no lo has impreso de la manera correcta

- No siempre se contempla, en algunas ocasiones devolverá un WA

¿ Y SI NO LO CONSIGO?

- **SIGUE PROBANDO!!**

Un mal veredicto no es excusa para rendirse, SIGUE!!!

- **INTERNET**

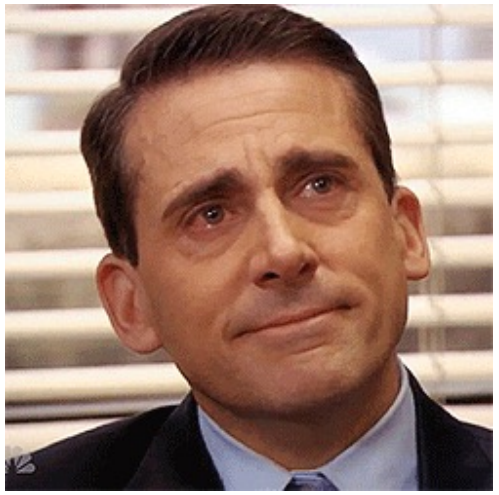
No lo tienes durante los concursos, pero entrenando no te quedes con la duda

- **CP3 y CP4**

La biblia de la programación competitiva (hay copias en la biblioteca)

- **Pide ayuda**

Usad el grupo de telegram para pedir ayuda

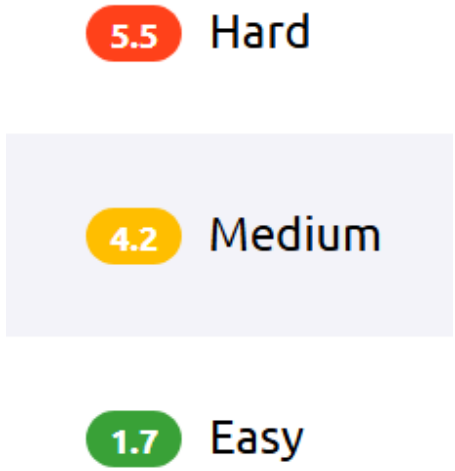


- Ideone
- Diffchecker

KATTIS

KATTIS

- Log in / Registro 
- Encontrar problemas
  - Ranking por dificultad de Kattis
  - Propuestos por nosotros



- IDE
  - Kattis directamente
  - Pycharm o IDE externo -> RECOMENDADO

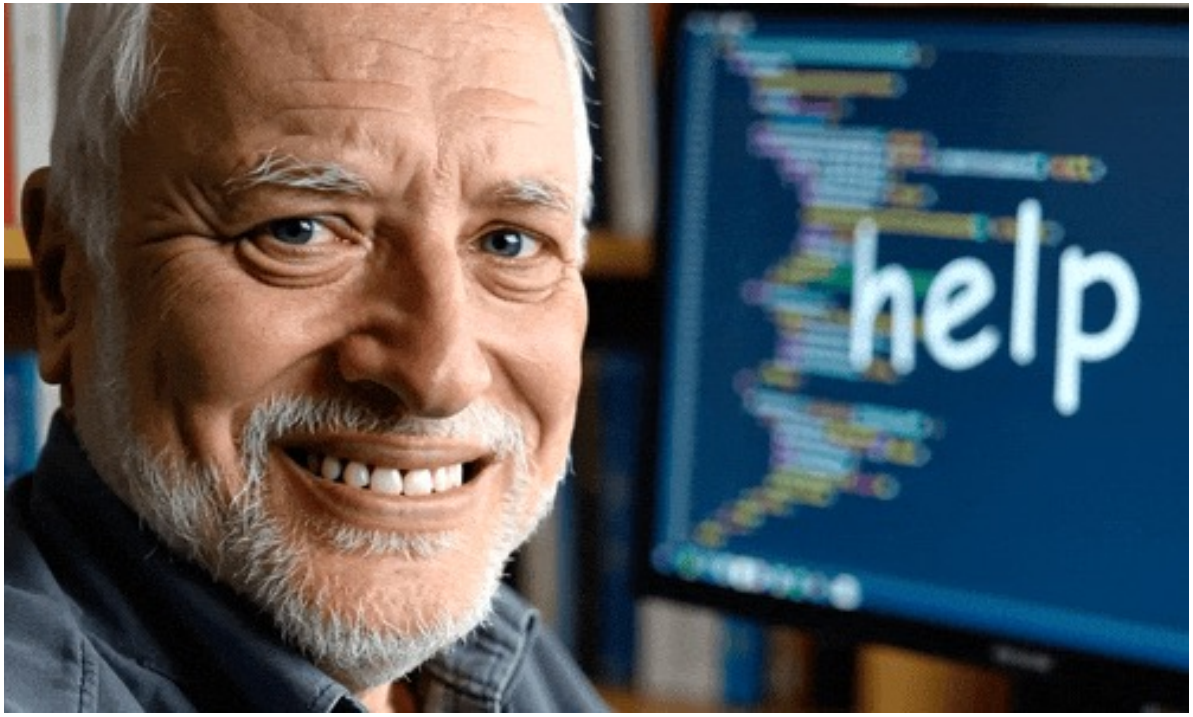
PYCHARM

- Crear un Proyecto
- Empezar un código
  - main + TAB

- Lectura de 1 solo caso

```
if __name__ == '__main__':
    name = input()
    age = int(input())
```

CODING TIME!



https://open.kattis.com/problems/timeloop

CARACTERÍSTICAS DE UN PROBLEMA: LECTURA DE T CASOS

Se recibe un entero T y luego vendrán T casos de prueba

```
n = int(input())
for i in range(n):
    #- leer datos de cada caso
    #- codigo + generar salida
```

https://judge.grafo.etsii.urjc.es/

CARACTERÍSTICAS DE UN PROBLEMA: LECTURA HASTA EOF

Se leen los casos hasta leer la marca EOF (End-Of-File)

```
import sys

for line in sys.stdin:
    #codigo
```

- line.strip() para quitar el salto de línea

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=607&page=show_problem&problem=3565

CARACTERÍSTICAS DE UN PROBLEMA: LECTURA HASTA CASO EN 0

Se lee el número de casos hasta que se consiga una condición de parada (generalmente cuando la entrada sea 0)

PYTHON

```
import sys

for line in sys.stdin:
    if(line=='0\n'):
        sys.exit()
    #codigo
```

- line.strip() para quitar el salto de línea

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=3402

¿Podemos tener todos los casos de prueba?

¿En una empresa te dan casos de prueba para encontrar un fallo?
¿En un examen te los dan?

Si se necesitan tras mucho esfuerzo se os pasarán para aprender, pero aprender a encontrar fallos es algo muy importante.
Busca tus propios casos, pon los casos límites y prueba el algoritmo.

Entrenamiento

- Páginas de entrenamiento.
  - Kattis
  - Acepta el Reto
  - Codeforces
  - OnlineJudge y uHunt
  - Spoj

  Recomendable tener un nick

Enlaces del curso

- El enlace del curso

https://urjc-cp.github.io/urjc-cp/

- Enlace de otras ediciones (algunas grabadas)

https://david8k.github.io/

Este año el curso NO se grabará.

Tengo mucho interés, ¿cómo avanzar?

1. Practicar lo máximo posible
   - Problemas propuestos de Kattis (concurso semanal).
   - Concursos de Codeforces (semanales).
2. **Revisar material de otros años** de las siguientes clases y practicar
3. Leer, entender y practicar cada tema del **libro Competitive Programming 3 o 4 (disponible en biblioteca)**.

Quien haga estos pasos durante varios años asegura medalla en el Europeo.

¿Preguntas?

CANALES Y FORMULARIO DE EQUIPOS

CP-2026

Formulario