Temario para problemas de OMI etapa nacional.

Autores: Luis Enrique Vargas Azcona y Sebastián Sánchez Lara

Revisión 2021

# Objetivos de este temario:

* Generar una base de conocimiento que se espera todos los participantes de la OMI tengan.
* Simplificar la propuesta de problemas.
* Apoyar a los estados con sus procesos selectivos, funcionando como guía sobre en que preparara a sus estudiantes.
* En los estados en los que no hay un proceso selectivo completo, darles a los alumnos una guía sobre lo que deberían intentar revisar y aprender por su cuenta.
* Elevar el nivel de los participantes en el PES, al tener una base sólida de materiales que deben conocer antes de la OMI.

# Sobre el temario

Esta propuesta está basada en la versión más reciente del syllabus de la IOI ( [ioi-syllabus-2019.pdf (ioinformatics.org)](https://ioinformatics.org/files/ioi-syllabus-2019.pdf) ). Como tal, cualquier tema que el ISC considere como excluido, también será considerado como excluido aquí.

Como tal, también usaremos las mismas categorías para definir la viabilidad de los temas que se usa en el syllabus que serán denotadas por números

* (0) Incluido, ilimitado: Temas que se consideran como información básica, se espera que los participantes conozcan el concepto, y se usará sin aclaraciones en las descripciones de los problemas.
* (1) Incluido, a ser definido: Temas que se esperan conocidos por los participantes, pero si se llega a usar en la descripción de un problema, irá junto a una definición formal.
* (2) Incluido, no para descripciones de problemas: Estos temas no serán usados en la descripción de problemas, pero su uso es esperado para resolver los problemas. Estos serán los de principal interés para prepararse para el concurso nacional.
* (3) Fuera de tema: Todos los temas que no sean mencionados en el syllabus deben ser considerados en esta categoría. No se espera que los competidores conozcan del tema, y si se llegaran a usar, el COMI se asegurará que sea razonable resolver el problema sin conocimiento previo del tema, y que pueda ser expresado claramente con temas de (0) y (1)
* (4)Excluido: Los temas en esta categoría están excluidos de la OMI, y se asegurará que ningún problema de la OMI requiera que el participantes conozca de estas áreas. Además se procurará que el conocimiento de estos temas no ayude a generar una solución más simple o que de más puntos. Es decir, el conocimiento o desconocimiento de estos temas no debería impactar en la puntuación de un participante.

Se debe aclarar que de ninguna manera se restringe el uso de ninguna técnica que el participante decida usar durante el concurso.

# 1 Matemáticas

## Aritmética

* (0) Operaciones en enteros incluyendo exponenciación.
* (0) Comparaciones entre enteros (Mayor que, menor que).
* (0) Propiedades básicas de los enteros (Signo, paridad, divisibilidad, primalidad).
* (0) Bases numéricas.
* (0) Números primos, primos relativos.
* (0) Fracciones, porcentajes.
* (0) Mínimo común múltiplo, máximo común divisor.
* (1) Aritmética modular básica(suma, resta, multiplicación).
* (2) Divisibilidad en aritmética modular.
* (2) Criba de Eratóstenes.
* (2) Pequeño teorema de Fermat.
* (3) Teorema de Euler.
* (4) Números complejos, números reales.

## Geometría

* (0) Conceptos de punto, línea, cuadrado, rectángulo, círculo, triángulo y polígono.
* (0) Coordenadas en un plano, parametrización de una recta (pendiente, ordenada).
* (0) Conceptos de perpendicularidad, líneas paralelas, ángulo.
* (0) Conceptos de área y perímetro de una figura geométrica.
* (1) Concepto de cóncavo y convexo.
* (2) Distancia euclidiana, distancia Manhattan.
* (2) Teorema de Pitágoras.
* (3) Trabajar en el código con números de precisión flotante (float, double), o manejar de manera compleja números racionales (fracciones).
* (3) Vectores.
* (4) Geometría en más de 2 dimensiones.
* (4) Curvas cónicas.
* (4) Funciones trigonométricas.
* (4) Números complejos, números irracionales.

## 

## Estructuras discretas

* (1) Funciones (inyectivas, sobreyectivas, biyectivas, funciones inversas, composición de funciones).
* (1) Relaciones (reflexividad, simetría, transitividad, relaciones de equivalencia, relaciones de orden, orden lexicográfico).
* (1) Conjuntos (inclusión\exclusión, complementos, conjuntos cartesianos, conjunto potencia).
* (4) Cardinalidad y contabilidad de conjuntos infinitos.

## Lógica básica

* (0) Lógica de primer orden (No se que procede, porque no esperamos que sepan esto de manera formal).
* (0) Conectores lógicos
* (0) tablas de verdad
* (0) Cuantificadores universales y de existencia.(Para todo, Para alguno, Existe...)
* (2) Modus ponens y modus tollens
* (4) Temas adicionales de lógica formal.

## Técnicas de demostración.

* (0) Definiciones matemáticas recursivas.
* (1) Nociones de implicación, contrapositiva, negación y contradicción.
* (2) Pruebas directas, pruebas por contraejemplo, contrapositiva, y contradicción.
* (2) Inducción matemática.
* (3) Inducción fuerte

## Conteo Básico

* (0) Argumentos de conteo(Principio aditivo, principio multiplicativo)
* (1) Sucesiones (Sucesión Aritmética, Sucesión geométrica, números de Fibonacci)
* (1) Permutaciones y Combinaciones (Definición básica)
* (1) Funcion Factorial
* (1) Coeficientes binomiales
* (2) Principio de inclusión Exclusión.
* (2) Principio de casillas
* (2) Identidad de Pascal, teorema del binomio
* (3) Suma de progresiones geométricas.
* (3) Números de Catalán.
* (4) Resolver recurrencias

## Grafos y árboles

* (1) Árboles y sus propiedades básicas.
* (1) Grafos no dirigidos(Grado, camino, ciclo)
* (1) Grafos dirigidos.
* (1) Árboles generadores.
* (1) Estrategias de recorrido de árboles y grafos
* (1) Grafos decorados/coloreados.
* (2) Multigrafos, grafos con autociclos.
* (2) Algoritmo de Dijkstra.
* (2) Matriz de adyacencia, listas de adyacencia y formas estándar de representación de grafos.
* (3) Grafos bipartitos, biparticiones
* (4) Hipergrafos
* (4) Clases específicas de grafos.
* (4) Parámetros estructurales
* (4) Test de planaridad
* (4) Encontrar separadores para grafos planos.

## Probabilidad discreta

Todos las aplicaciones donde son finitos casos, y por lo tanto puede ser expresado como argumentos de combinatoria es (3), (4) en cualquier otro caso.

## Otros temas de matemáticas

* (4) Geometría en 3 o más dimensiones.
* (4) Algebra lineal.
* (4) Cálculo.
* (4) Teoría de juegos combinatorios (teorema de Sprague-Grundy, juegos de NIM)
* (4) Estadística

# Ciencias computacionales

## Fundamentos de Programación

* (0) Sintaxis básica de C++.
* (0) Tipos de variables.
* (0) Entrada y salida de consola.
* (0) Lectura y escritura de archivos.
* (0) Estructuras de control de flujo(if, while, for)
* (0) Funciones
* (3) Metaprogramación.
* (4) Opciones de compilación.
* (4) Uso de múltiples hilos.

## Algoritmos y resolución de problemas.

* (2) Estrategias de resolución de problemas.
* (2) El uso de algoritmos en la resolución de problemas.
* (2) Estrategias de implementación de algoritmos.
* (2) Estrategias de depuración.
* (2) Posiciones ganadoras y perdedoras.
* (2) Notación *O* grande.
* (2) Propiedades de algoritmos(Eficiencia, correcto).

## Estructuras primitivas de Datos

* (0) Datos primitivos (booleanos, enteros con y sin signos, caracteres).
* (0) Arreglos.
* (0) Cadenas y procesamientos de cadenas.
* (1) Estrategias de implementación en árboles y grafos.
* (2) Alocación estática y en pila.
* (2) Estrategias para elegir la estructura de datos adecuada.
* (3) Representación de datos en la memoria.
* (4) Implementación linkeada.
* (4) Implementación de números reales.
* (4) Implementación de punto flotante.
* (4) Apuntadores y referencias.
* (4) Usar fracciones para realizar cálculos exactos.

## Recursión

* (0) El concepto de recursión.
* (0) Funciones recursivas matemáticas.
* (0) Procedimientos simples de recursión.
* (2) Implementación de recursión.
* (3) Estrategias de Divide and Conquer
* (4) Backtracking recursivo

## Problemas interactivos

Entender e implementar estos problemas es (1).

## Algoritmos y complejidad

* (2) Análisis de complejidad asintótica superior de algoritmos. Notación Big O.
* (2) Clases de complejidad estándar: Constante, logarítmico, lineal, O(NlogN), cuadrática, cúbica, exponencial.
* (2) Intercambio entre tiempo y espacio en algoritmos.
* (4) Análisis empírico de complejidad.
* (3) Identificar diferencias entre mejor caso, peor caso y caso promedio.
* (3) Notación little o, Omega, y Theta.
* (3) Cambiar parámetros para reducir tiempo, gasto de memoria, u otras medidas de rendimiento.
* (4) Usar recurrencias para analizar algoritmos recursivos.

## Estrategias de algoritmos

* (2) Estrategias de ciclos simples.
* (2) Algoritmos de fuerza bruta(búsqueda exhaustiva).
* (2) Algoritmos glotones.
* (2) Búsqueda binaria
* (2) Divide y vencerás.
* (2) Backtracking.
* (2) Búsquedas en amplitud, profundidad y priorizadas.
* (2) Algoritmos minimax.
* (2) Máscara de bits.
* (2) Binary-lifting.
* (2) Algoritmos fuera de línea.
* (3) Recta que barre (sweep-line).
* (3) Programación dinámica.
* (3) Heurísticas.
* (3) Algoritmos aleatorios.
* (4) Encontrar buenos parámetros para aprendizaje máquina.
* (4) Algoritmos de aproximación discreta.
* (4) Algoritmos de clustering
* (4) Minimizar funciones multivariables usando métodos numéricos.

## Algoritmos

* (2) Algoritmo de Euclides.
* (2) Criba de Eratóstenes.
* (2) Test de primalidad en O(sqrt(n)).
* (2) Factorizacion.
* (2) Manipulación simple de arreglos.
* (2) Algoritmos simples de cadenas(Búsqueda de subcadenas sencilla ).
* (2) Búsqueda binaria.
* (2) Algoritmos de ordenamiento NlogN y lineales (cubetas).
* (2) Recorridos ordenados de árboles. (pre-order, in-order, post-order).
* (2) DFS, BFS.
* (2) Unión pertenencia.
* (2) Algoritmo de Dijkstra.
* (3) Orden topológico.
* (3) Árboles mínimos de expansión.
* (3) Teoría de juegos, posiciones ganadoras y perdedoras. Algoritmo de Min-Max.
* (3) Encontrar componentes conexos,y cerraduras transitivas.
* (3) Algoritmos de Bellman-Ford, Floyd-Warshall.
* (4) Pareo máximo en grafos bipartitos.
* (4) Puentes, codos en grafos no dirigidos.
* (4) Conversión radix.
* (4) Ciclos/caminos eulerianos.
* (4) Flujo máximos
* (4) Cobertura de vértices

## Estructuras de datos avanzadas

* (1) Pilas y colas.
* (2) Representaciones de grafos (listas de adyacencia, matriz de adyacencia).
* (2) Estructura de union-find.
* (2) Estructuras incluídas en la stl (set, multiset, map, unordered\_map).
* (3) Árboles de búsqueda binaria en segmentos incluyendo árboles de Fenwick, y segment tree.
* (3) Resolver el problema del menor ancestro en común.
* (3) Algoritmos en strings y estructuras de datos
* (4) Arboles balanceados de búsqueda binaria.
* (4) Árboles binarios aumentados.
* (4) Crear estructuras de datos persistentes.
* (4) Anidar estructuras de datos
* (4) Tries(Diccionario).
* (4) Heavy-Light decomposition
* (4) Estructuras de datos para cambiar dinámicamente árboles.
* (4) Heap binarios.
* (4) Tablas de hash.

## Algoritmos distribuidos

Todo está en (4).

## Autómatas y gramática

Todo esta en (4)