Técnicas de Diseño de Algoritmos (Ex Algoritmos y Estructuras de Datos III)

Segundo cuatrimestre 2025

(bienvenidos!)

Programa

1. Repaso de Complejidad Computacional parte 1, Dividir y Conquistar
2. Repaso de Complejidad Computacional parte 2, fuerza Bruta y backtracking
3. Programación Dinámica
4. Algoritmos Golosos
5. Introducción a la teoría de grafos y algoritmos sobre grafos
6. Problema de árbol generador mínimo
7. Problema de camino mínimo
8. Problemas de flujo en redes

Bibliografía

1. Jon Kleinberg and Eva Tardos, *Algorithm Design*, Pearson Education Limited, 2005.
2. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, *Introduction To Algorithms (Fourth Edition)*, MIT Press, 2022.

Régimen de cursada 2do cuatrimestre 2025

- Cursada:
  1. Lunes: clases teóricas (salvo 10/11)
  2. Miércoles: clases prácticas
- Evaluaciones:
  1. Dos parciales (fechas tentativas).
     - 1er parcial (29/09)
     - 2do parcial (17/11)
     - recuperatorio 1er parcial (01/12)
     - recuperatorio 2do parcial (10/12)
  2. 3 Talleres individuales

Régimen de cursada 2do cuatrimestre 2025 (cont.)

- Promoción. Se promocionan si aprueban tanto los parciales como los Talleres. La nota de la promoción es el promedio de las notas de los parciales. Es posible mejorar dicha calificación con las siguientes opciones.
  - Se puede sumar hasta dos puntos bonus para la nota de promoción a través de cuestionarios sorpresivos (quices) en 4 a 5 de las clases teóricas. En algún momento de la clase teórica, los alumnos presentes deberán firmar una planilla de control y se publicará el link del cuestionario. Las respuestas deben ser enviadas al finalizar la clase.
  - En base a los resultados de los talleres individuales, se puede subir también la nota.

Cambios de turnos/comisiones e inscripciones fuera de término

Por favor informen su situación y datos hasta el jueves 21/8 a través de (i) Consultas Administrativas (Campus) (ii) Mails a Mariano Martín González González (marianogonzalez@dc.uba.ar) si no tienen acceso al campus.

- Los pedidos de cambio de turno y/o cambiar de comisión de TDA a AED 3 o viceversa serían aceptados sin inconveniente.
- Los pedidos de nuevas inscripciones van a ser considerados una vez que se hayan recepcionados todos.

Complejidad computacional: Las reglas del juego

- En el contexto de la teoría de complejidad computacional, llamamos problema a la descripción de los datos de entrada y la respuesta a proporcionar para cada dato de entrada.
- Una instancia de un problema es un juego válido de datos de entrada.
- Ejemplo:
  1. **Entrada:** Un número n entero no negativo.
  2. **Salida:** ¿El número n es primo?
- En este ejemplo, una instancia está dada por un número entero no negativo.

Complejidad computacional: Las reglas del juego

- Suponemos una Máquina RAM (*random access memory*).
  1. La memoria está dada por una sucesión de celdas numeradas. Cada celda puede almacenar un valor de b bits.
  2. Supondremos habitualmente que el tamaño b en bits de cada celda está fijo, y suponemos que todos los datos individuales que maneja el algoritmo se pueden almacenar con b bits.
  3. Se tiene un programa imperativo no almacenado en memoria, compuesto por asignaciones y las estructuras de control habituales.
  4. Las asignaciones pueden acceder a celdas de memoria y realizar las operaciones estándar sobre los tipos de datos primitivos habituales.

Complejidad computacional: Las reglas del juego

- Cada instrucción tiene un tiempo de ejecución asociado.
  1. El acceso a cualquier celda de memoria, tanto para lectura como para escritura, es $O(1)$.
  2. Las asignaciones y el manejo de las estructuras de control se realiza en $O(1)$.
  3. Las operaciones entre valores lógicos son $O(1)$.

- Las operaciones entre enteros/reales dependen de b:
  1. Las sumas y restas son $O(b)$.
  2. Las multiplicaciones y divisiones son $O(b \log b)$.

$\Rightarrow$ Si b está fijo, estas operaciones son $O(1)$. En cambio, si no se puede suponer esto, entonces hay que contemplar que el costo de estas operaciones depende de b.

Complejidad computacional: Las reglas del juego

- **Tiempo de ejecución de un algoritmo A:**
  $T_A(I) =$ suma de los tiempos de ejecución de las instrucciones realizadas por el algoritmo con la *instancia* I.
- Dada una instancia I, definimos $|I|$ como la cantidad de bits necesarios para almacenar los datos de entrada de I.
  1. Si b está fijo y la entrada ocupa n celdas de memoria, entonces $|I| = bn = O(n)$.
- **Complejidad de un algoritmo A:**
  $f_A(n) = \max_{I:|I|=n} T_A(I)$.

Repaso: Notación O

Dadas dos funciones $f, g : \mathbb{N} \to \mathbb{R}$, decimos que:

- $f(n) = O(g(n))$ si existen $c \in \mathbb{R}_+$ y $n_0 \in \mathbb{N}$ tales que $f(n) \leq c\, g(n)$ para todo $n \geq n_0$.
- $f(n) = \Omega(g(n))$ si existen $c \in \mathbb{R}_+$ y $n_0 \in \mathbb{N}$ tales que $f(n) \geq c\, g(n)$ para todo $n \geq n_0$.
- $f(n) = \Theta(g(n))$ si $f = O(g(n))$ y $f = \Omega(g(n))$.

Repaso: Notación O

- Si un algoritmo es $O(n)$, se dice lineal.
- Si un algoritmo es $O(n^2)$, se dice cuadrático.
- Si un algoritmo es $O(n^3)$, se dice cúbico.
- Si un algoritmo es $O(n^k)$, $k \in \mathbb{N}$, se dice polinomial.
- Si un algoritmo es $O(\log n)$, se dice logarítmico.
- Si un algoritmo es $O(d^n)$, $d \in \mathbb{R}_{>1}$, se dice exponencial.

- Cualquier función exponencial es *peor* que cualquier función polinomial: Si $k \in \mathbb{R}_{>1}$ y $d \in \mathbb{N}$ entonces k^n no es $O(n^d)$.
- La función logarítmica es *mejor* que la función lineal (no importa la base), es decir $\log n$ es $O(n)$ pero no a la inversa.