Estructuras de Datos y Algoritmos – IIC2133

Outline

- Programa del curso
- Prerrequisitos
- Algoritmos y notación
- Memoria de un computador
- Estructuras básicas

Outline

- Programa del curso
- Prerrequisitos
- Algoritmos y notación
- Memoria de un computador
- Estructuras básicas

Contenidos

Estructuras fundamentales: arreglos, listas ligadas, stacks, colas, tablas de hash, colas priorizadas

Árboles de búsqueda: árboles binarios, árboles binarios balanceados, árboles 2-3

Algoritmos de ordenación: *insertionsort*, *heapsort, quicksort*, *mergesort*, *countingsort*, análisis de desempeño, propiedades de ordenación

Técnicas algorítmicas: dividir para conquistar, *backtracking*, programación dinámica, algoritmos codiciosos

Grafos: representación, exploración, ordenación topológica, componentes fuertemente conectadas, árboles de cobertura mínimos, rutas más cortas

Metodología

Lunes y Miércoles: Clases expositivas online

Viernes: Ayudantías o talleres

Las clases

Las clases serán por videoconferencia (Zoom)

Tendremos espacios donde se discuta la materia y se resuelvan dudas:

- en particular, las clases de los miércoles 30 sept. y 11 nov. van a ser repaso de la materia para la I1 e I2, respectivamente

Haremos un break de 5 minutos a los 30 – 40 minutos de clase

Las clases

Para terminar la clase puntualmente, avísenme cuando falten 3 – 5 minutos (para las 3.20)

Todo el material de clases (y la clase misma) será subido a la página del curso en Github

Aún estamos aprendiendo a hacer clases online: cualquier sugerencia (herramienta, metodología, etc.) es bienvenida

Las tareas

Durante el semestre habrá 5 tareas de programación en **C**

La **nota de tareas** $(\boldsymbol{NT})$ se calcula de la siguiente manera:

$$\boldsymbol{NT} = \frac{T_0 + T_1 + T_2 + T_3 + T_4}{5}$$

Las interrogaciones

Tres interrogaciones —dos en horario de clases y la tercera en el horario del examen:

I1: lunes 5 octubre I2: lunes 16 noviembre I3: viernes 11 diciembre

Cada interrogación tendrá 4 preguntas, de las cuales ustedes deberán contestar tres

No hay examen

La **nota de interrogaciones** $(\boldsymbol{NI})$ se calcula así:

$$\boldsymbol{NI} = \frac{I1 + I2 + I3}{3}$$

La **nota final, *NF*,** se calcula así

$$\boldsymbol{NF} = \begin{cases} \dfrac{NI + NT}{2}, & si\ NI \geq 3.7\ y\ NT \geq 3.7 \\ \min\left(3.9, \dfrac{NI + NT}{2}\right), & en\ otro\ caso \end{cases}$$

Código de Honor

Este curso suscribe el **Código de Honor** de la universidad

http://www.uc.cl/codigo-de-honor/

Copias y otros serán sancionados con nota final ***NF*** = 1.1 en el curso

GitHub: plataforma oficial del curso

En el GitHub del curso podrán encontrar:

- Guías de instalación de C y algunas librerías
- El foro para dudas de tareas, materia, etc.
- Los enunciados de las tareas y las diapositivas de clases
- Sus propios repositorios para entregar las tareas

Deben contestar la encuesta en el **SIDING** para poder acceder

Discord: punto de encuentro

Debido a la cuarentena, ya no existe la posibilidad de acercarse a los ayudantes en busca de ayuda

El punto de encuentro entre ayudantes y estudiantes será el servidor de **Discord** del curso

Problemas

El equipo docente del curso estamos aquí para ayudarlos a aprender

Si durante el semestre se les presentan problemas, a veces podemos ayudar

Comuníquense con nosotros: `yadran@ing.puc.cl`

Outline

- Programa del curso
- Prerrequisitos
- Algoritmos y notación
- Memoria de un computador
- Estructuras básicas

Matemáticas

- Exponentes
- Logaritmos
- Series
- Aritmética modular

Demostraciones

- Por inducción
- Por contradicción
- Por contraejemplo

Algoritmos y programación

- Recursión

Outline

- Programa del curso
- Prerrequisitos
- Algoritmos y notación
- Memoria de un computador
- Estructuras básicas

Algoritmo: Definición

"Secuencia ordenada de pasos que permite hacer un cálculo o hallar la solución de un tipo de problemas"

Los algoritmos son **independientes** de los lenguajes de programación

Usaremos **pseudocódigo** para describirlos

Notación para pseudocódigo

- Control de flujo: ***if***, ***else***, ***while***, ***for***
- Lenguaje matemático (operaciones lógicas, de conjuntos, vectoriales, etc.)
- Atributos, métodos y subíndices
- Lenguaje natural, si es más claro que usando lo anterior

Algoritmo simple para identificar si un número es primo

$$isprime(x):$$

$$\quad for\ i \in [2..\sqrt{x}]:$$

$$\quad\quad if\ x \bmod i = 0:$$

$$\quad\quad\quad return\ false$$

$$\quad return\ true$$

En este caso puede ser más claro usar lenguaje natural:

$isprime(x)$:

    $for\ i \in [2..\sqrt{x}]$:

        if x es divisible por i:

            $return\ false$

    $return\ true$

Algoritmos y su implementación

Para un **algoritmo** no existe una única **implementación**

En clases veremos los algoritmos de manera conceptual

En las tareas tendrán que pensar cómo implementarlos

Buenos algoritmos, mejores algoritmos

Además de que sea correcta

… en este curso nos interesa estudiar qué tan buena es una solución a un problema

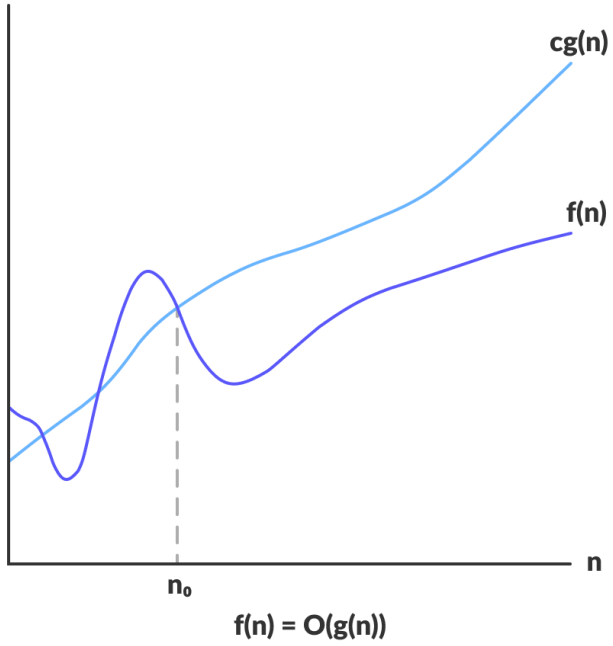
La principal herramienta que usamos para esto es la de **complejidad computacional**

Complejidad

$f(n) \in O(g(n))$:

$$\exists\, n_0, c > 0$$

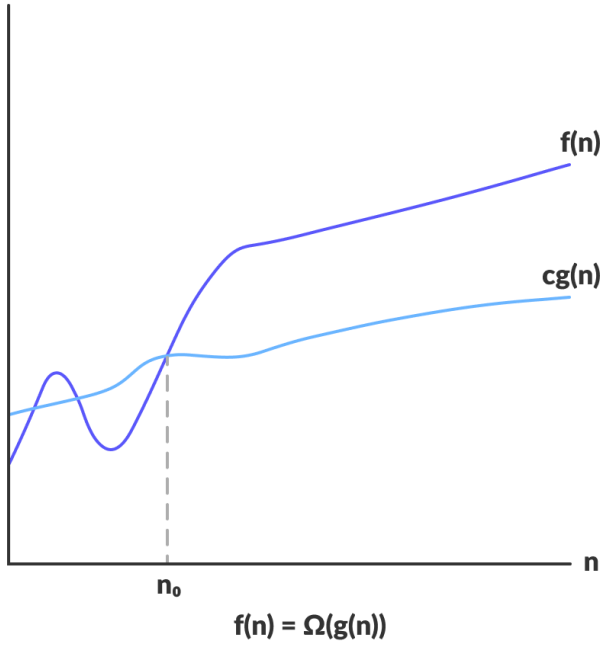
$$f(n) < c \cdot g(n), \qquad \forall\, n > n_0$$



Complejidad

$f(n) \in \Omega(g(n))$:

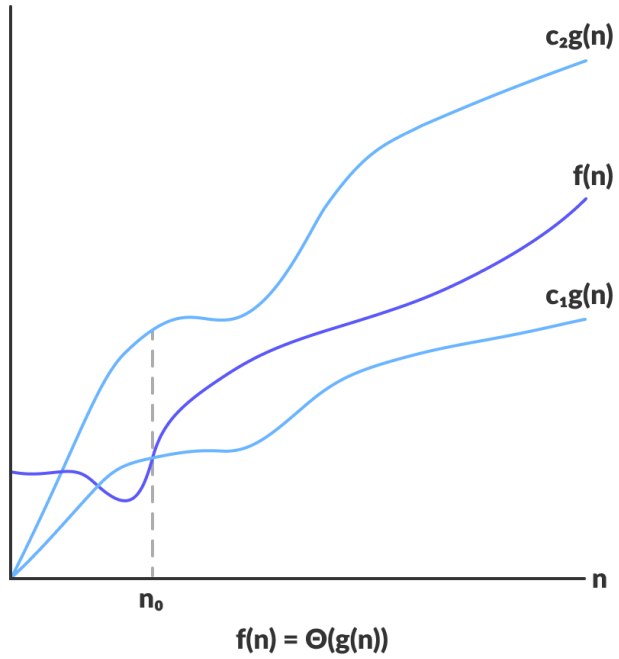
$$g(n) \in O(f(n))$$



Complejidad

$f(n) \in \Theta(g(n))$:

$$f(n) \in O(g(n)) \quad \wedge \quad f(n) \in \Omega(g(n))$$



Complejidad: resumen de cálculo

- Si un algoritmo tiene varias partes que se ejecutan una después de otra (secuencialmente), su complejidad es la **suma** de las complejidades de cada parte
- Por lo tanto, si una parte se repite n veces (p.ej., un *loop*), entonces (a veces) se puede **multiplicar** su complejidad por n, y luego sumar al resto del algoritmo
- Al final, sólo queda el término que **crece más rápido**

Complejidad de tiempo y memoria

Nos interesan dos tipos de complejidades para algoritmos:

- Complejidad de tiempo: $T(n)$
- Complejidad de memoria **adicional**: $M(n)$

Ambos son relativos al **tamaño del input** (i.e., el número de datos de entrada), n

Si bien $T(n)$ es nuestra prioridad, nunca olvidar que

$$T(n) \in \Omega\big(M(n)\big)$$

Outline

- Programa del curso
- Prerrequisitos
- Algoritmos y notación
- Memoria de un computador
- Estructuras básicas

Memoria RAM: Experimento

Abre una consola de Python en tu computador

Ejecuta el siguiente código:

```
a = object()
print(a)
```

¿Qué significa lo que aparece en consola?

Memoria RAM

La memoria principal (RAM) de un computador puede ser imaginada como una gran tabla, arreglo o matriz de bits:

- 32 columnas
- algunos miles de millones de filas

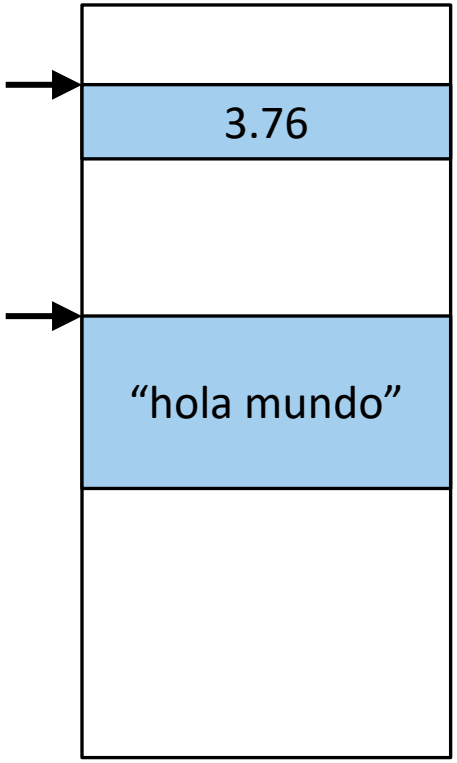
Cada fila tiene una dirección única:

- su posición relativa dentro de la tabla
- un número narural que parte en 0 (la dirección de la primera fila) y aumenta de 4 en 4

Memoria RAM

Cada variable de un programa tiene:

- posición (dirección) en memoria
- tamaño, en número de bytes
- valor



Outline

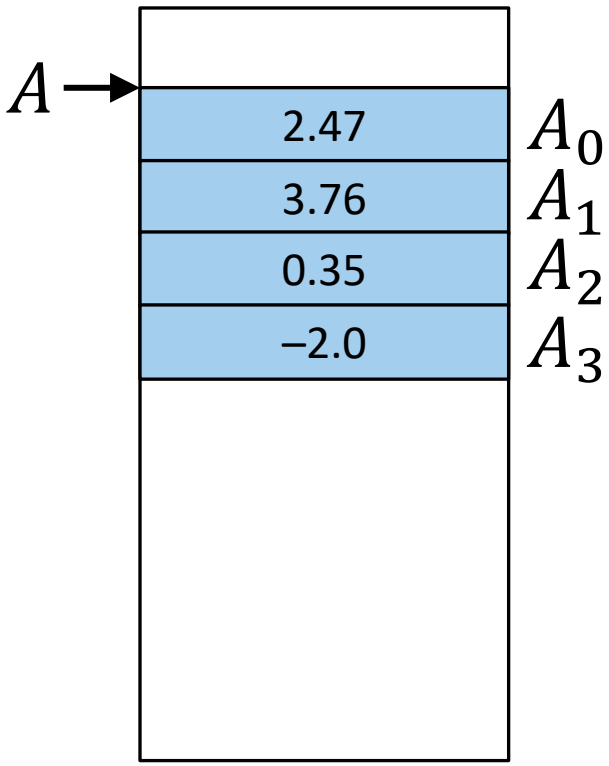
- Programa del curso
- Prerrequisitos
- Algoritmos y notación
- Memoria de un computador
- Estructuras básicas

Arreglos

Secuencia de **largo fijo** de celdas del mismo tamaño

Se almacena de manera **contigua** en memoria

Permite acceso por índice en O(1)



Arreglo: Ejemplo abstracto

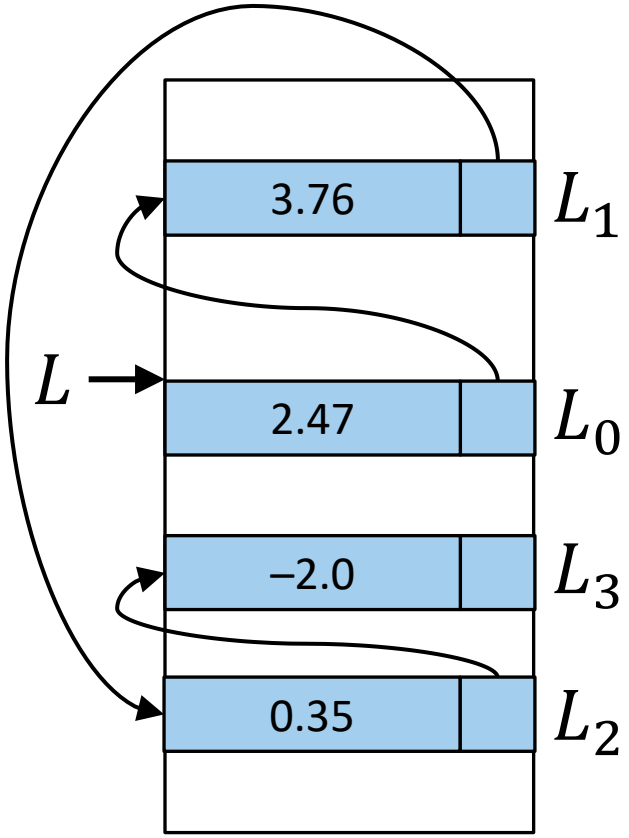
	2.47	**3.76**	**0.35**	**–2.0**	

Listas ligadas

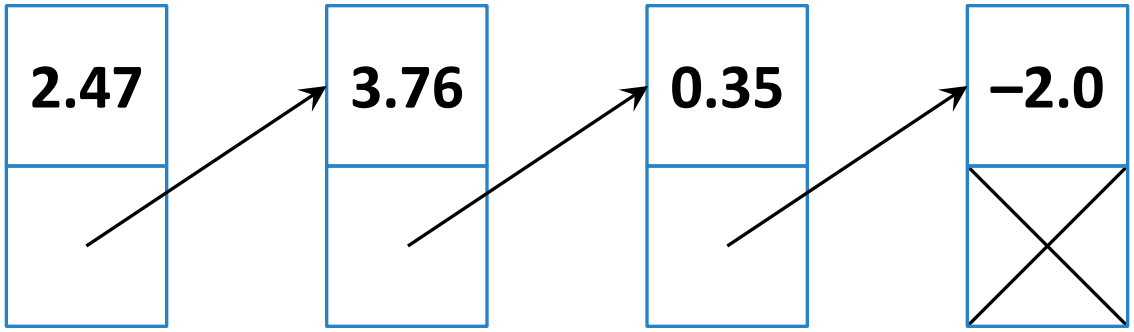
Secuencia de **largo variable** de celdas del mismo tamaño

Se almacena de manera **aleatoria** en memoria conectada mediante punteros

No permite acceso eficiente por índice



Lista ligada: Ejemplo abstracto



Lenguaje C

Este miércoles y viernes estudiaremos **C**

Esta actividad será muy relevante para las tareas del curso

Les llegará un correo con las instrucciones que deberán seguir para acceder al material

Bibliografía

T. Cormen, C. Leiserson, R. Rivest, C. Stein, “Introduction to Algorithms” 3rd ed., The MIT Press 2009

R. Sedgewick, K. Wayne, “Algorithms” 4th ed., Addison-Wesley 2011

M. Weiss, “Data Structures and Algorithm Analysis in C++”4th ed., Pearson 2014