Estructuras de datos y algoritmos

IIC 2133 - Sección 3

Prof. Eduardo Bustos

Sumario

US$ 71,383.86

US$ 71,383.86

Costo de 2 horas de una función recursiva
sin caso base en Google Cloud.[1]

[1] *"The story of how one little recursive function took down an entire business"*
https://www.youtube.com/watch?v=N6lYcXjd4pg

$$2^{20} \cdot 2^{14}$$

$$2^{20} \cdot 2^{14}$$

Número total de filas y columnas en una hoja de cálculo
(1.048.576 filas por 16.384 columnas)[2]

[2] *Especificaciones y límites de Excel* @ https://support.microsoft.com

Estructuras de datos y algoritmos

Estudio de algoritmos y su **implementación eficiente**
mediante estructuras de datos

Sumario

Objetivos del curso

1. Desarrollar el **pensamiento algorítmico**.
2. Estudiar y aplicar distintos tipos de **estructuras de datos**.
3. Estudiar y aplicar distintas **técnicas algorítmicas**.

Contenidos del curso

1. Estructuras fundamentales
   - Arreglos
   - Listas ligadas
   - Stacks
   - Heaps
   - Colas
   - Tablas de Hash
2. Árboles de búsqueda
   - Árboles binarios y balanceados
   - Árboles 2-3
   - Árboles B+
3. Algoritmos de ordenación
   - *selectionsort*
   - *insertionsort*
   - *heapsort*
   - *quicksort*
   - *mergesort*
   - *countingsort*
4. Técnicas algorítmicas
   - Dividir para conquistar
   - *Backtracking*
   - Programación dinámica
   - Algoritmos codiciosos
5. Algoritmos en grafos
   - Representación de grafos
   - Exploración
   - Ordenación topológica
   - Componentes fuertemente conectadas
   - Árboles de cobertura
   - Rutas más cortas

Metodología

Clases expositivas:	lunes y miércoles, módulo 4.
	sala B23
Ayudantías:	Viernes módulo 4.
	sala K201

El material de clases será subido a la página del curso en Github

Evaluación

1. Tareas de programación.

2. Interrogaciones escritas.

Tareas

- Cuatro tareas de programación en **C**.
  - No hay tareas optativas.
  - La tarea 0 sirve de introducción al lenguaje C.
- Las fechas de publicación serán informadas oportunamente.
- Plazo de ≈ 2 semanas por tarea.
- La corrección de las tareas se realizará de forma automatizada.

Tareas: política de atrasos

- Poseerán 2 **cupones de atraso** durante el semestre
- Cada cupón permite postergar el día de entrega de la tarea correspondiente 1 día calendario
- **Importante:** los cupones solo podrán ser usados en las tareas 1, 2 y 3 (no en la tarea 0)
- En caso de entregar una tarea con atraso, se hará un descuento:
  - 5 décimas por a lo más 3 horas de atraso
  - 2 puntos por a lo más 24 horas de atraso

  Tareas con más de 1 día de atraso se califican con nota 1.0

Interrogaciones

- Tres interrogaciones escritas (presenciales)

	Fecha	Hora
Interrogación 1	miércoles 7 de mayo	17:30
Interrogación 2	jueves 19 de junio	17:30
Examen	martes 8 de julio	8:20

- La inasistencia a la interrogación 1 o 2 debe ser **justificada ante la DIPRE** para optar a preguntas adicionales en el examen a modo de reemplazar su nota faltante
- No se podrá optar a preguntas recuperativas sin justificación
- La inasistencia al examen debe ser **justificada ante la DIPRE** para evaluar la aplicación de nota P

Es responsabilidad del estudiante cualquier tope de evaluaciones con otros cursos de la Escuela de Ingeniería

Evaluación

Nota **Tareas** (**NT**):

$$\mathbf{NT} \;=\; 0,2 \cdot T_0 \;+\; 0,3 \cdot T_1 \;+\; 0,25 \cdot T_2 \;+\; 0,25 \cdot T_3$$

Nota **Interrogaciones** (**NI**):

$$\mathbf{NI} \;=\; 0,3 \cdot I_1 \;+\; 0,3 \cdot I_2 \;+\; 0,4 \cdot Ex$$

Nota **Final** (**NF**):

$$\mathbf{NF} \;=\; \begin{cases} \dfrac{\mathbf{NI} + \mathbf{NT}}{2}, & \text{si } \mathbf{NI} \geq 3,7 \text{ y } \mathbf{NT} \geq 3,7 \\ \min\big(\mathbf{NF},\; 3,9\big), & \text{en otro caso} \end{cases}$$

Código de Honor

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

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

La copia y otros serán sancionados con nota final **NF** $= 1,1$ en el curso

Comunicación digital

- Anuncios . . .

Canvas del curso

- Clases, guía de instalación de C, enunciados de tareas y repositorios de entrega de tareas

Github del curso

- Preguntas sobre materia y tareas

Issues de Github

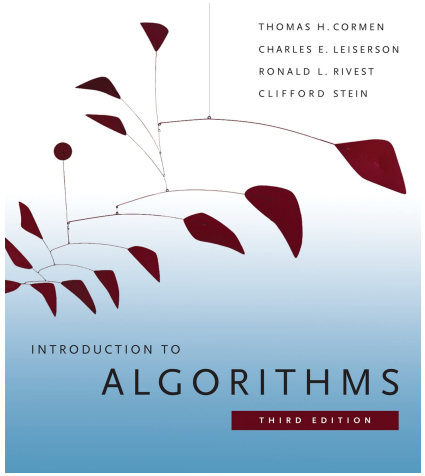
Ante problemas...

El equipo docente del curso estamos aquí para ayudarlos a aprender

Si durante el semestre se les presentan problemas, escríbannos cuanto antes.
A veces podremos ayudar

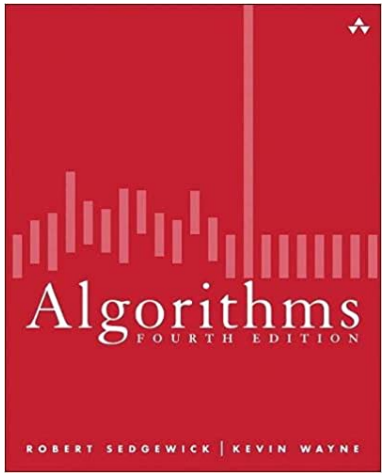
Comuníquense con nosotros!

Bibliografía del curso



Introduction to Algorithms

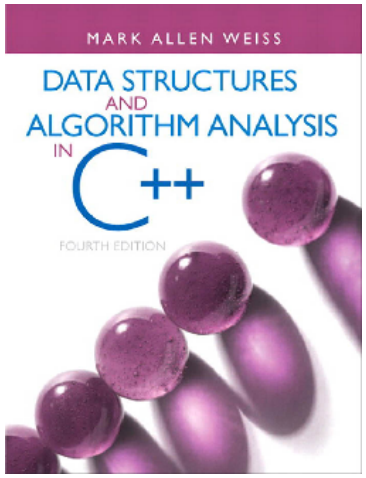
- T. Cormen, C. Leiserson, R. Rivest, C. Stein
- The MIT Press
- Tercera edición: 2009



Algorithms

- R. Sedgewick, K. Wayne
- Addison-Wesley
- Cuarta edición: 2011

Bibliografía del curso



"Data Structures and Algorithm Analysis in C++

- M. Weiss
- Pearson
- Cuarta edición: 2014.

Sumario

Herramientas matemáticas

Herramientas que necesitamos (las iremos refrescando)

- Exponentes
- Logaritmos
- Series
- Aritmética modular

Métodos de demostración

Métodos para demostrar que más utilizaremos

- Inducción (simple)
- Contradicción
- Demostración por contraejemplo

Nociones de programación

Usaremos regularmente la noción de **recursión**

Sumario

¿qué es un algoritmo?

Definición

Un **algoritmo** es una 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

Un algoritmo puede estar dado por cualquier lenguaje:

- Lenguaje de programación.
  - C, Python, Java, C++, etc
- Lenguaje natural.
- Pseudo-código.

Usaremos **pseudo-código** para describir algoritmos

Pseudocódigo

La notación que usaremos para el pseudocódigo incluye

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

Pseudocódigo: un ejemplo

Algoritmo simple para identificar si un número es primo

input : Número natural $n \geq 2$
output: Valor booleano

`isprime` *(n)*:
    for $i \in [\, 2 \ldots \sqrt{n}\,]$:
        if $n \mod i = 0$:
            return false
    return true

Pseudocódigo: un ejemplo

En este caso puede ser más claro usar lenguaje natural

input : Número natural $n \geq 2$
output: Valor booleano

`isprime` *(n)*:
    for $i \in [\, 2 \ldots \sqrt{n}\,]$:
        if *n es divisible por i* :
            return false
    return true

Implementación

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

- En clases veremos algoritmos de manera conceptual
- En las tareas tendrán que pensar en cómo implementarlos

No solo nos interesa que la implementación sea **correcta**... nos interesa estudiar **qué tan buena** es una solución a un problema

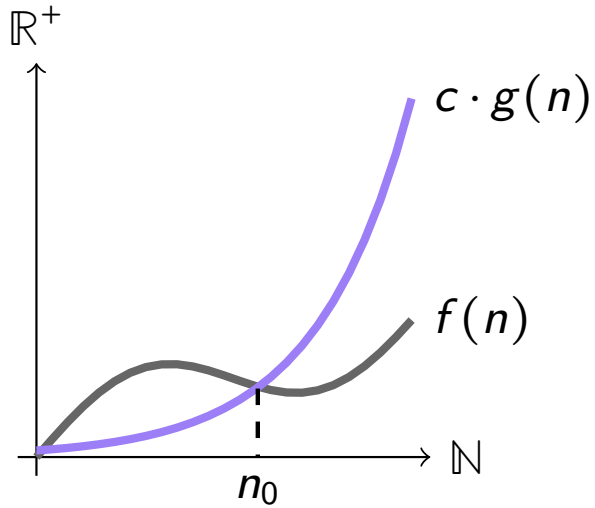
La principal herramienta que usaremos para esto
es la **complejidad computacional**

Complejidad: Notación $\mathcal{O}$

Definición

Para $g : \mathbb{N} \to \mathbb{R}^+$, se define el conjunto $\mathcal{O}(g)$ de las funciones $f : \mathbb{N} \to \mathbb{R}^+$ tales que **existen constantes** $c > 0$ y $n_0 \in \mathbb{N}$ tales que **para todo** $n \geq n_0$:

$$f(n) \leq c \cdot g(n)$$



Si $f \in \mathcal{O}(g)$, entonces f **crece más lento o igual** que g.

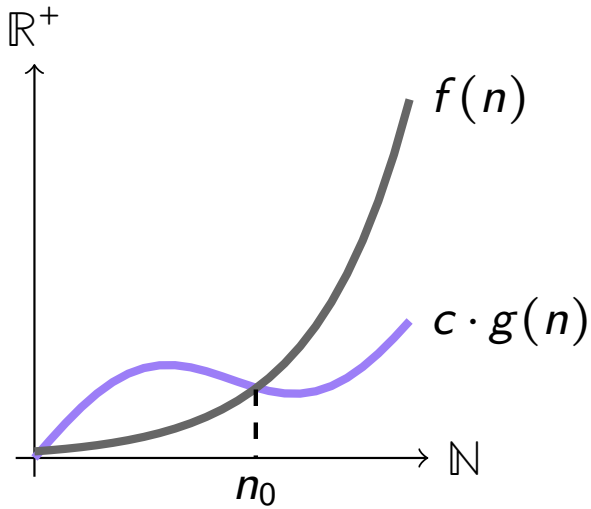
Complejidad: Notación Ω

Sean $f : \mathbb{N} \to \mathbb{R}^+$ y $g : \mathbb{N} \to \mathbb{R}^+$.

Definición

Para $g : \mathbb{N} \to \mathbb{R}^+$, se define el conjunto $\Omega(g)$ tal que

$$f \in \Omega(g) \iff g \in \mathcal{O}(f)$$



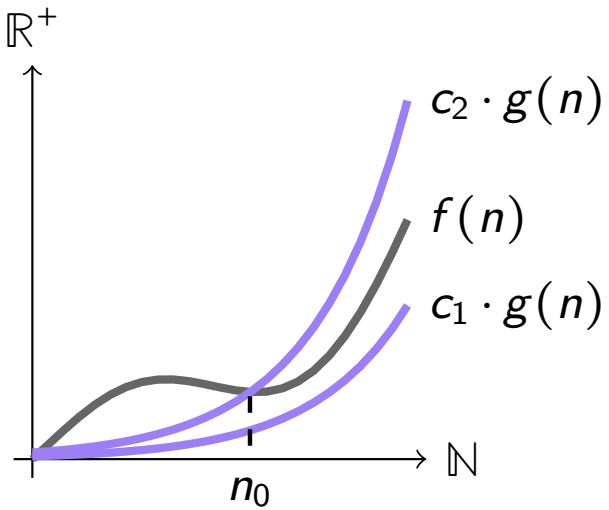
Si $f \in \Omega(g)$, entonces f **crece más rápido o igual** que g.

Complejidad: Notación Θ

Definición

Para $g : \mathbb{N} \to \mathbb{R}^+$, se define el conjunto de funciones $\Theta(g)$ tal que

$$f \in \Theta(g) \iff f \in \mathcal{O}(g) \wedge f \in \Omega(g)$$



Si $f \in \Theta(g)$, entonces f **crece igual** que g.

Complejidad: resumen de cálculo

Nos interesan dos tipos de complejidades para algoritmos como funciones del **tamaño del input** n (i.e. número de datos de entrada)

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

Si bien en general $T(n)$ es nuestra prioridad, no hay que olvidar que

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

Complejidad: resumen de cálculo

Para etapas sucesivas (una después de la otra) dentro de un algoritmo

- La complejidad de la secuencia de etapas es la **suma** de sus complejidades
- Consecuencia: un paso que se repite n veces, en general, contribuye con una complejidad que es n **veces** la complejidad del paso individual
- Al sumar, el término que **crece más rápido** es el que domina la complejidad

Sumario

Memoria RAM

Un **bit** es la unidad indivisible de información computacional que puede valer 0 o 1

Además, un **bit** es una celda física indivisible de almacenamiento en un computador

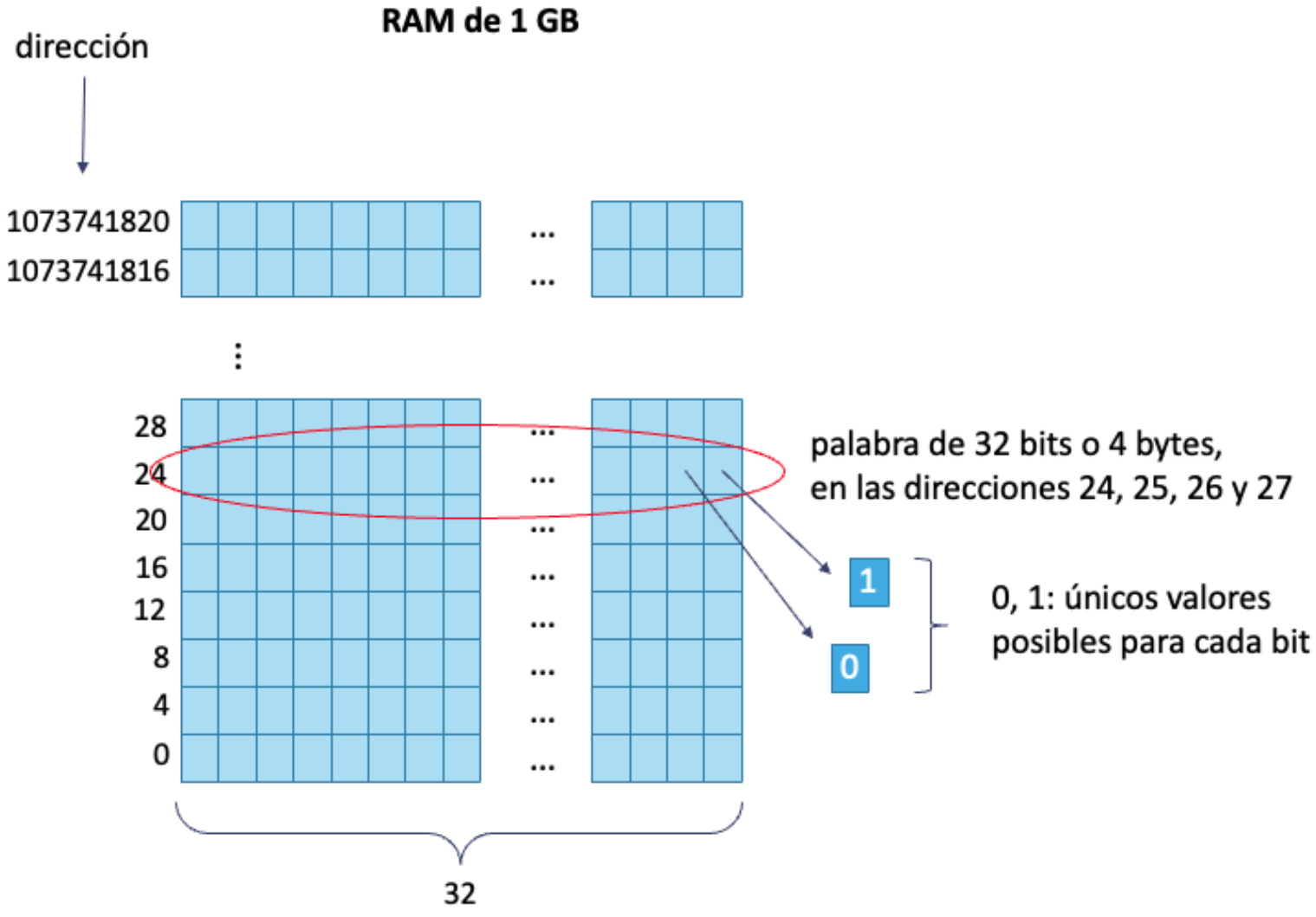
La memoria principal de un computador (RAM) puede ser imaginada como una gran tabla, arreglo o matriz de bits

- Cierto número de columnas
- Algunos miles de millones de filas

Cada fila de esta tabla tiene una dirección única

- Corresponde a su posición relativa dentro de la tabla
- Es un natural que parte en 0 (dirección de la primera fila) y aumenta de 4 en 4

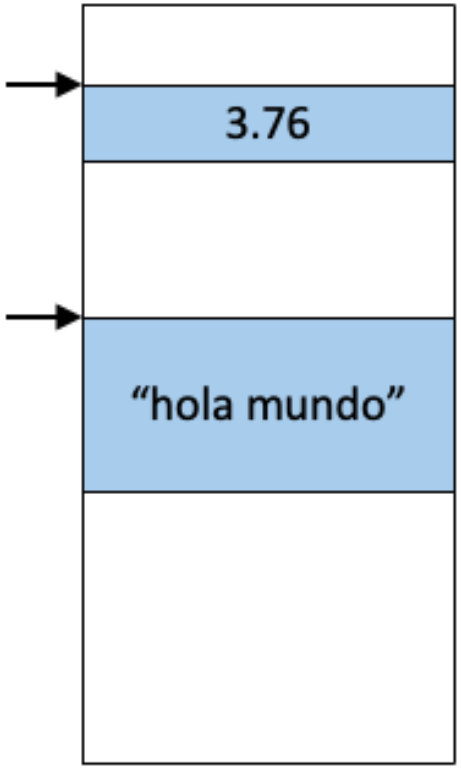
Memoria RAM


RAM de 1 GB
dirección
1073741820
1073741816
28
24
20
16
12
8
4
0
palabra de 32 bits o 4 bytes,
en las direcciones 24, 25, 26 y 27
1
0
0, 1: únicos valores
posibles para cada bit
32

Memoria RAM

Cada variable de un programa tiene

- Posición (dirección de memoria)
- Tamaño, en número de bytes
- Valor



Sumario

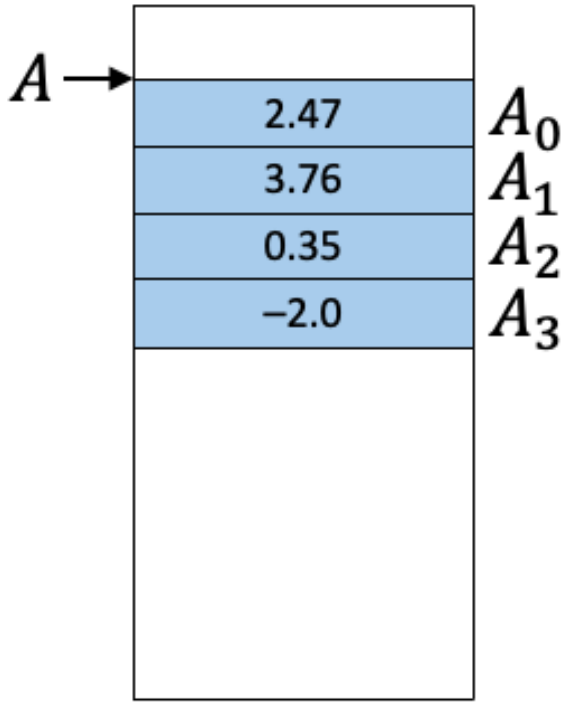
Arreglos

Un **arreglo** es una secuencia de celdas que tiene **largo fijo**

- Cada celda es del mismo tamaño
- Almacenan valores del mismo tipo

Se almacena en memoria de manera **contigua**

- Esto permite acceso por índice en tiempo $\mathcal{O}(1)$



Arreglos

	2.47	**3.76**	**0.35**	**−2.0**	

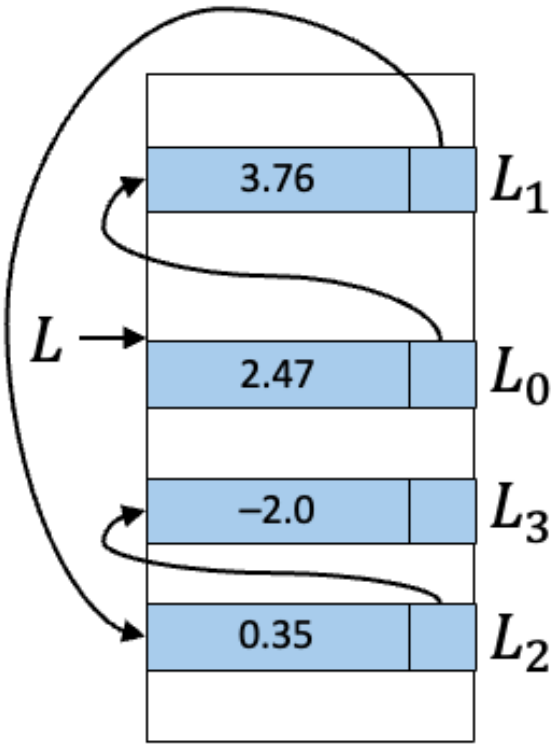
Listas ligadas

Una **lista ligada** es una secuencia de celdas que tiene **largo variable**

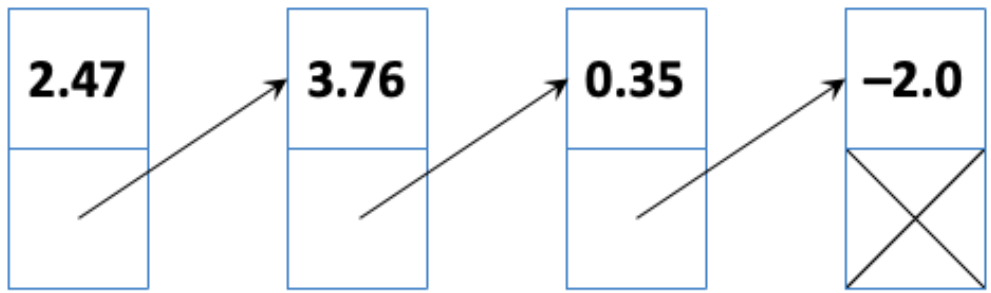
- Cada celda es del mismo tamaño

Se almacena en memoria de manera **aleatoria**, utilizando punteros

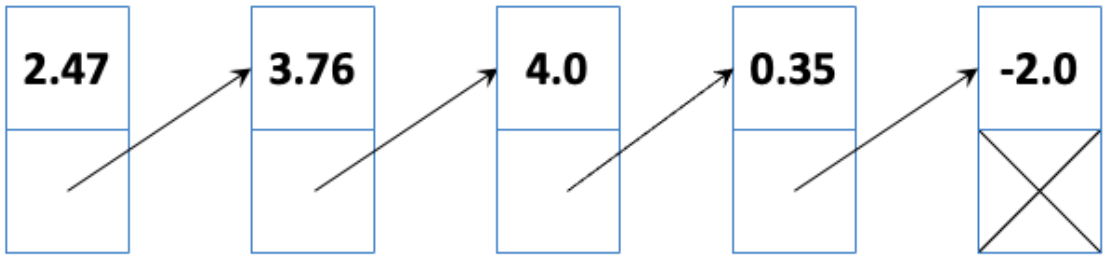
- Esto no permite acceso eficiente por índice



Listas ligadas



Listas ligadas



Outline

Esta semana: lenguaje C

Este miércoles y viernes estudiaremos C

Es una actividad **muy relevante** para las tareas del curso

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

Consejos para afrontar el curso

Este curso mezcla teoría y práctica

- No basta con revisar las diapositivas de clase para preparar las interrogaciones
- No basta con leer el enunciado de las tareas el día antes de la entrega

Esperamos que sean capaces de desarrollar habilidades que son muy útiles para su vida profesional, pero esto requiere dedicación

Consejos

- Luego de revisar las diapositivas, pónganse a prueba: ¿puedo crear un ejemplo propio para explicar este concepto?
- Abordar ejercicios de los libros, quizás viendo soluciones, para luego...
- Enfrentarse a problemas de la bibliografía sin ver soluciones.

El desafío es que **creen** sus propios algoritmos