Estructuras de Datos y Algoritmos – IIC2133

Outline

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

Outline

- Programa del curso
- 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, otros árboles de búsqueda balanceados

Algoritmos de ordenación: *insertionsort*, *heapsort, quicksort*, análisis de desempeño

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

Grafos: representación, exploración, ordenación topológica, árboles de cobertura mínimos, rutas más cortas, flujo máximo en redes

Las clases

Nos importa que aprovechen las clases, por lo que hemos preparado un sistema de instancias de aprendizaje activo

Podrán reconocer las distintas instancias por sus íconos:

Den ideas

Nos importa que puedan deducir los contenidos del curso

No tengas miedo de decir lo que piensas, nada es obvio

Cuando veas la ampolleta, es el momento de dar ideas

Discute con tus compañeros

Nos importa que puedas discutir ideas con otras personas

¿Cuál crees que es la respuesta y por qué?

Cuando veas los globos de texto, prepara tu argumento

Metodología

Lunes: Clases magistrales con el profesor (yo)

Miércoles: Clases de aprendizaje activo con los ayudantes Vicente y Antonio en dos salas

Viernes: Ayudantías o talleres de C según corresponda

Las tareas

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

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

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

Evaluaciones escritas

Las interrogaciones se separarán en dos mitades cada una y serán en horario se clases, en días miércoles. Cada mitad consistirá en dos preguntas:

$I1_1$: 28/8 $I1_2$: 11/9 $I2_1$: 2/10 $I2_2$: 16/10 $I3_1$: 30/10 $I3_2$: 20/11

Al final del semestre habrá un examen: 27/11

La **nota de interrogaciones** $(\boldsymbol{NI})$ es el promedio de las mejores 10 preguntas de las ies

La **nota de evaluaciones escritas** $(\boldsymbol{NE})$ se calcula así:

$$\boldsymbol{NE} = \begin{cases} \dfrac{\boldsymbol{Ex} + \boldsymbol{NI}}{\boldsymbol{2}}, & \boldsymbol{Ex} \geq \boldsymbol{NI} \\ \dfrac{\boldsymbol{Ex} + \boldsymbol{2NI}}{\boldsymbol{3}}, & \boldsymbol{else} \end{cases}$$

La **nota final (NF)** se calcula así

$$\boldsymbol{NF} = \begin{cases} \dfrac{NE + NT}{2}, & si\ NE \geq 4 \\ \min\left(NE, \dfrac{NE + NT}{2}\right), & en\ otro\ caso \end{cases}$$

Código de Honor

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

http://www.uc.cl/codigodehonor/el-codigo

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

GitHub: Plataforma oficial del curso

En el repositorio del curso en GitHub 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

Outline

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

Algoritmos

Nuestra primera ampolleta

¿Qué es un algoritmo?

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

$$isprime(x):$$

$$\quad for\ i \in [2..x-1]:$$

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

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

$$\quad return\ true$$

En este caso puede ser más claro usar lenguaje natural:

$isprime(x)$:

    $for\ i \in [2..x-1]$:

        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 como implementarlos

Buenos algoritmos, mejores algoritmos

¿Qué hace que una solución a un problema sea buena?

¿Cuál será la mejor?

Complejidad

$f(x) \in O\big(g(x)\big)$:

$$\exists\, x_0, k > 0$$

$$f(x) < k \cdot g(x), \qquad \forall\, x\ > x_0$$

$f(x) \in \Omega\big(g(x)\big)$:

$$g(x) \in O\big(f(x)\big)$$

$f(x) \in \Theta\big(g(x)\big)$:

$$f(x) \in O\big(g(x)\big) \qquad y \qquad f(x) \in \Omega\big(g(x)\big)$$

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 x veces (p.ej., un *loop*), entonces (a veces) se puede **multiplicar** su complejidad por x, 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
- Algoritmos y notación
- Memoria de un computador
- Estructuras básicas

Memoria RAM: Experimento

Abre una consola de Python en tu computador

Ejecuta el siguiente código:

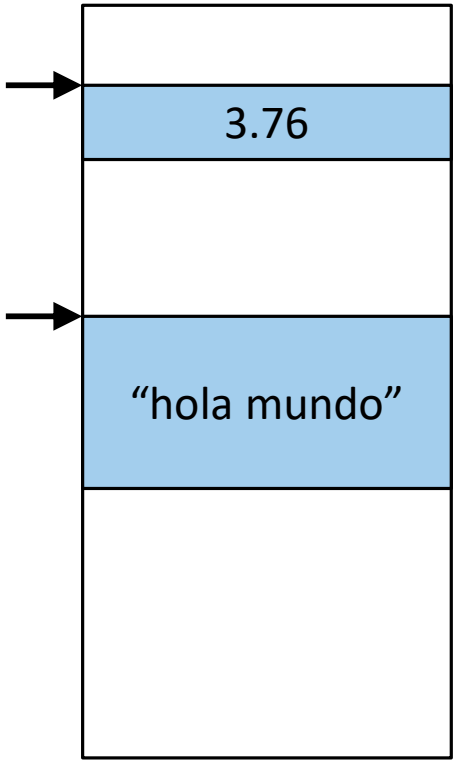
```
a = object()
print(a)
```

¿Qué significa lo que aparece en consola?

Memoria RAM

Cada variable de un programa tiene:

- Posición en memoria
- Tamaño, en bytes
- Valor



Outline

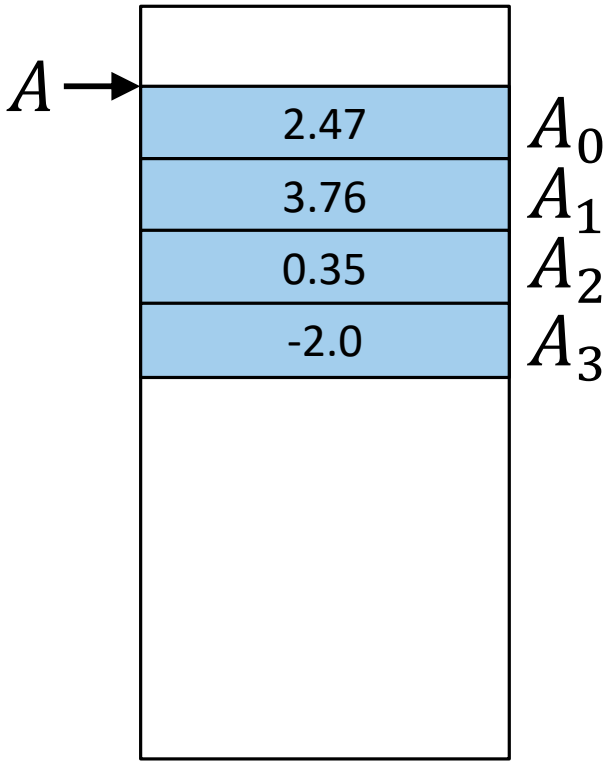
- Programa del curso
- Algoritmos y notación
- Memoria de un computador
- Estructuras básicas

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

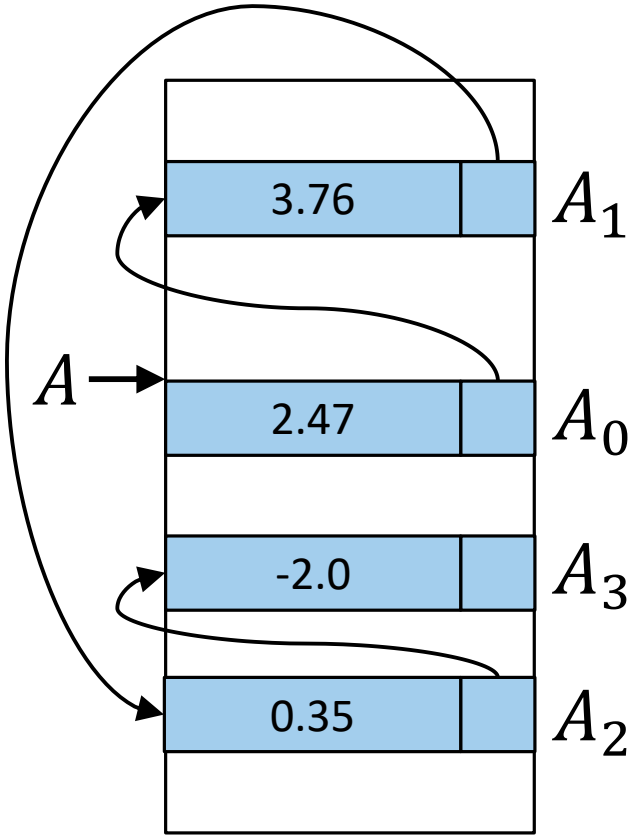
	2.47	**3.76**	**0.35**	**-2.0**	

Listas ligadas

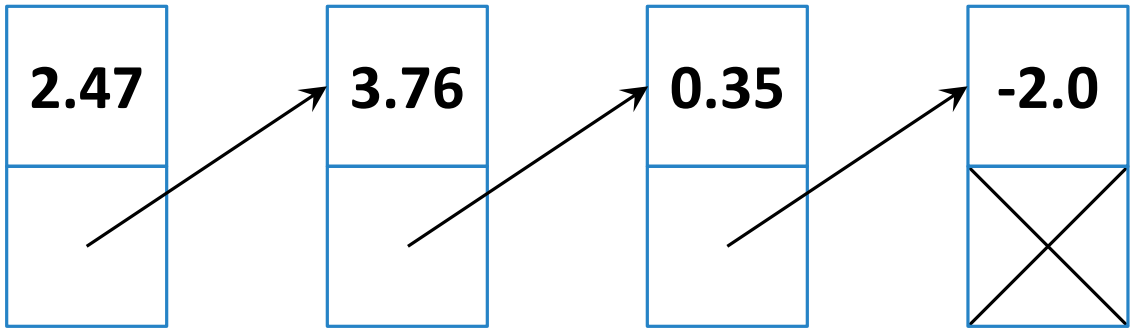
Secuencia de **largo variable** de celdas del mismo tamaño

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

No permite acceso eficiente por índice



Lista ligada: Ejemplo abstracto



Este miércoles y viernes

Este miércoles y viernes serán los primeros talleres introductorios a **C**

y muy relevantes para las tareas del curso

Necesitan lo siguiente:

- Haber leído y seguido la guía de instalación de **C**
- Traer tu computador: si no tienes, puedes trabajar con un compañero