Guía 4

Algoritmos y Estructuras de Datos II, DC, UBA.

Segundo cuatrimestre de 2020

Índice

1. Modalidad de trabajo

1.1. Clases

La materia se divide en clases teóricas, clases prácticas y clases de laboratorio. Como regla general, el material expositivo de las clases será grabado en diferido (no en vivo) y estará disponible en video en la sección **Clases** del sitio de la materia en el Campus[1]. Cualquier excepción a esta regla se informará con anticipación.

1.2. Vías de comunicación y espacios de consulta

Toda la información importante sobre la materia se comunicará a través de la lista de correo `algo2-alu@dc.uba.ar` que pueden leer y escribir todes les docentes y alumnes de la materia. Las consultas a docentes acerca de cuestiones administrativas o de índole personal se pueden hacer a través de la lista `algo2-doc@dc.uba.ar` que pueden leer todes les docentes de la materia, y a la que pueden escribir todes les alumnes.
Las consultas acerca de temas de la materia se responderán a través de los siguientes medios:

1. Para comunicación *asincrónica* (estilo foro) en la sección **Foro de Consultas** del sitio de la materia en el Campus se pueden dejar por escrito consultas que les docentes podrán leer y responder. Aconsejamos revisar el foro antes de formular una pregunta para ver si ya fue respondida con anterioridad.

2. Para comunicación *sincrónica* (estilo chat/videoconferencia), pondremos disponibles canales de comunicación (probablemente a través de Zoom). En general las consultas tendrán lugar en las siguientes franjas horarias:

   - **Laboratorio:** miércoles de 09:30 a 17:30 (turno mañana) y de 17:30 a 20:30 (turno noche).
   - **Práctica:** viernes de 11:00 a 14:00 (turno mañana) y de 17:00 a 20:00 (turno noche).

Tengan en cuenta que les docentes disponen de tiempo limitado para responder consultas asincrónicas, y que elaborar una respuesta por escrito puede consumir mucho tiempo y prestarse a malentendidos. Por eso aconsejamos **limitar el uso de la comunicación asincrónica** a dudas administrativas, y **aprovechar los espacios de consulta sincrónica** para resolver dudas de la materia.

1.3. Correlativas

Para poder cursar la materia se requiere tener aprobada la cursada (no así el final) de Algoritmos y Estructuras de Datos I antes del comienzo del cuatrimestre.

1.4. Guías de ejercitación y criterios de aprobación

La materia se dividirá en cuatro bloques. Cada bloque tendrá una **guía** como esta, que se encontrará disponible en la sección **Guías de ejercitación** del sitio de la materia en el Campus. Cada guía incluye:

1. **Ejercicios seleccionados (no se entregan).** Preguntas teóricas y ejercicios prácticos destinados a validar que hayan entendido el material presentado en las clases teóricas. Estas preguntas discutirán en los horarios de consulta y eventualmente se publicarán soluciones de algunas o varias de ellas.

2. **Ejercicios obligatorios del laboratorio (TPs y talleres)**. Los TPs son grupales (en grupos de 4 integrantes). Los talleres son individuales. Cada instancia se debe entregar antes de la fecha de finalización del bloque, y podrá ser aprobada o devuelta para incorporar correcciones. Se detallan en la Sección 3 (margen verde).

3. **Parciales.** Al final de cada bloque se publicará un parcial (individual). Tendrán tres días para resolver cada parcial. Cada ejercicio de cada parcial podrá ser aprobada o desaprobado y se recupera individualmente al final del cuatrimestre.

La forma de entrega de los ejercicios individuales y trabajos prácticos será informada oportunamente. Para aprobar los prácticos de la materia será necesario aprobar **todas** las instancias de evaluación (tras las instancias de recuperación correspondientes).

En caso de que la Facultad vuelva a funcionar de manera presencial, los mecanismos de evaluación podrían sufrir modificaciones.

[1] `https://campus.exactas.uba.ar/course/view.php?id=1846`

1.5. Cronograma tentativo de la materia

Se incluye un cronograma tentativo de la materia. Se publica con fines orientativos, pero está sujeto a todo tipo de cambios. Los colores corresponden a teórica (T), laboratorio (L) y práctica (P). Las clases están numeradas: por ejemplo el lunes 13/04 está programado que se publiquen las primeras dos clases teóricas (T01 y T02).

1. **Semana 1**
   - Lun 31/08 — Presentación de la materia. T01, T02: Especificación
   - Mié 02/09 — L01: Uso de clases (09:30-12:30 y 17:30-20:30)
   - Vie 04/09 — Consultas T01, T02 (11:00-12:00 y 17:00-18:00)
2. **Semana 2**
   - Lun 07/09 — P01: TADs y recursión (básico) ..... **Inicio del bloque 1**
   - Mié 09/09 — L02: Clases en C++ (básico), testing — **Presentación TP1** (09:30-12:30 y 17:30-20:30)
   - Vie 11/09 — Consultas P01 (11:00-14:00 y 17:00-20:00)
3. **Semana 3**
   - Lun 14/09 — T03: Complejidad — P02: TADs y recursión (avanzado)
   - Mié 16/09 — L03: Clases en C++ (09:30-12:30 y 17:30-20:30)
   - Vie 18/09 — Consultas T03 (11:00-12:00 y 17:00-18:00) — Consultas P02 (12:00-14:00 y 18:00-20:00)
4. **Semana 4**
   - Lun 21/09 — P03: TAD en dos niveles — T04: Diseño
   - Mié 23/09 — L04: Memoria dinámica (09:30-12:30 y 17:30-20:30)
   - Vie 25/09 — Consultas T04 (11:00-12:00 y 17:00-18:00) — Consultas P03 (12:00-14:00 y 18:00-20:00)
   - Dom 27/09 — **Entrega TP 1** – **Publicación Parcial 1**
5. **Semana 5**
   - Lun 28/09 — Consultas Parcial 1 (11:00-14:00 y 17:00-20:00)
   - Mar 29/09 — **Entrega Parcial 1** — P04: Notación "O", complejidad ..... **Inicio del bloque 2**
   - Mié 30/09 — L05: Listas enlazadas — **Presentación TP2** (09:30-12:30 y 17:30-20:30)
   - Vie 02/10 — Consultas P04 (11:00-14:00 y 17:00-20:00)
6. **Semana 6**
   - Lun 05/10 — T05: Diccionarios sobre ABBs y AVLs, P05: Rep y Abs
   - Mié 07/10 — L06: Templates y algoritmos genéricos (09:30-12:30 y 17:30-20:30) — **Devolución TP 1**
   - Vie 09/10 — Consultas T05 (11:00-12:00 y 17:00-18:00)
7. **Semana 7**
   - Lun 12/10 — **Feriado: Día del Respeto a la Diversidad Cultural**
   - Mié 14/10 — Consultas (09:30-12:30 y 17:30-20:30)
   - Vie 16/10 — Consultas pre-parcial (11:00-14:00 y 17:00-20:00)
   - Dom 18/10 — **Entrega TP 2** — **Publicación Parcial 2**
8. **Semana 8**
   - Lun 19/10 — Consultas Parcial 2 (11:00-14:00 y 17:00-20:00)
   - Mar 20/10 — **Entrega Parcial 2** — T06: Tries ..... **Inicio del bloque 3**
   - Mié 21/10 — L07: ABBs — **Presentación TP3** (09:30-12:30 y 17:30-20:30)
   - Vie 23/10 — Consultas T06 (11:00-12:00 y 17:00-18:00)

9. **Semana 9**
   - Lun 26/10 — T07: Hashing 1 y 2 — P06: Interfaces y módulos
   - Mié 28/10 — Consultas (09:30-12:30 y 17:30-20:30) — **Devolución TP 2**
   - Vie 30/10 — Consultas T07 (11:00-12:00 y 17:00-18:00) — Consultas P06 (12:00-14:00 y 18:00-20:00)
10. **Semana 10**
   - Lun 02/11 — T08: Colas de prioridad — P07: Elección de estructuras
   - Mié 04/11 — L08: Tries (09:30-12:30 y 17:30-20:30)
   - Vie 06/11 — Consultas T08 (11:00-12:00 y 17:00-18:00) — Consultas P07 (12:00-14:00 y 18:00-20:00)
11. **Semana 11**
   - Lun 09/11 — T09: Sorting básico — P08: Elección de estructuras avanzadas
   - Mié 11/11 — L09: Heaps — **Presentación TP4** (09:30-12:30 y 17:30-20:30)
   - Vie 13/11 — Consultas P08 y pre-parcial (11:00-14:00 y 17:00-20:00)
   - Dom 15/11 — **Entrega TP 3** — **Publicación Parcial 3**
12. **Semana 12**
   - Lun 16/11 — Consultas Parcial 3 (11:00-14:00 y 17:00-20:00)
   - Mar 17/11 — **Entrega Parcial 3** — P09: Sorting ..... **Inicio del bloque 4**
   - Mié 18/11 — Consultas (09:30-12:30 y 17:30-20:30)
   - Vie 20/11 — Consultas T09 (11:00-12:00 y 17:00-18:00) — Consultas P09 (12:00-14:00 y 18:00-20:00)
13. **Semana 13**
   - Lun 23/11 — **Feriado: Día de la Soberanía Nacional** — T10: Divide and conquer — P10: D&C
   - Mié 25/11 — L10: Desarrollo de iteradores (09:30-12:30 y 17:30-20:30) — **Devolución TP 3** — **Posible entrega TP 4 (fecha 1/3)**
   - Vie 27/11 — Consultas T10 (11:00-12:00 y 17:00-18:00) — Consultas P10 (12:00-14:00 y 18:00-20:00)
14. **Semana 14**
   - Lun 30/11 — Consultas pre-parcial (11:00-14:00 y 17:00-20:00)
   - Mié 02/12 — Consultas (09:30-12:30 y 17:30-20:30) — **Posible entrega TP 4 (fecha 2/3)**
   - Jue 03/12 — **Publicación Parcial 4**
   - Vie 04/12 — Consultas Parcial 4 (11:00-14:00 y 17:00-20:00)
   - Sáb 05/12 — **Entrega Parcial 4**
15. **Semana 15**
   - Lun 07/12 — **Feriado con fines turísticos** — T11: Sorting y diccionarios en memoria externa — T12: Splay trees y skip lists
   - Mié 09/12 — L12: Sorting (ex taller de sorting) (09:30-12:30 y 17:30-20:30) — **Límite entrega TP 4 (fecha 3/3)**
   - Vie 11/12 — Consultas T11, T12 (11:00-12:00 y 17:00-18:00) — **Límite entrega talleres**
16. **Semana 16**
   - Lun 14/12 — Consultas pre-recuperatorio (11:00-14:00 y 17:00-20:00)
   - Vie 18/12 — **Presentación Recuperatorio**[2]
17. **Semana 17**
   - Lun 21/12 — **Entrega Recuperatorio**

[2] Cada ejercicio de cada parcial se recupera de manera independiente.

2. Ejercicios seleccionados

Los **ejercicios seleccionados** son un conjunto de ejercicios *mínimos* destinados a que cada estudiante pueda realizar una autoevaluación sobre su progreso en el dominio de los contenidos que se presentan en la materia, tanto desde el aspecto conceptual (entendimiento de los temas) como en el aspecto procedimental (capacidad de aplicar los conocimientos para resolver problemas prácticos). Deberían servir como disparadores para repasar partes de las clases que no hayan quedado claras, referirse a bibliografía complementaria y formular consultas.

Corrección: se brindarán resoluciones de los ejercicios prácticos, y se comentará sobre estas resoluciones en las clases de consulta. La idea es que no acudan a la resolución brindada por la cátedra sin haber intentado resolverlos por su cuenta.

Alentamos que los piensen individualmente, los discutan con sus compañeres y los consulten con les docentes. **Advertencia:** la resolución de los ejercicios seleccionados en esta guía no sustituye la resolución de prácticas (guías de ejercicios) publicadas en el sitio de la materia en el Campus.

2.1. Quiz sobre las clases teóricas

2.1.1. Clase T09: Sorting

- Exhibir arreglos de entrada con los que se alcance el mejor caso de *selection sort* e *insertion sort*, que son $\Theta(n)$ en mejor caso. Exhibir arreglos de entrada con los que se alcance el peor caso de *quicksort*, que es $\Theta(n^2)$ en peor caso, suponiendo que se elige como pivote siempre el primer elemento del arreglo.
- Sabemos que *mergesort* y *heapsort* son $O(n \log n)$ en peor caso y *quicksort* es $O(n \log n)$ en caso promedio, si se trabaja sobre arreglos. ¿Cómo se verían afectadas las complejidades si se cambian los arreglos por listas enlazadas?
- Podemos ordenar una secuencia de 0s y 1s usando *counting sort*. Por ejemplo, dada la secuencia 01110110, podemos notar que hay tres 0s y cinco 1s, para producir a continuación 00011111 como salida. En general, el costo de ordenar con este método es $O(n)$ en peor caso. ¿Por qué esto no contradice la cota inferior de $\Omega(n \log n)$ en peor caso vista en la teórica?
- ¿Cuáles de los algoritmos de ordenamiento conocidos se pueden hacer *in-place*, es decir, sobreescribiendo el arreglo de entrada y sin usar memoria adicional (salvo variables locales)?
- Recordar que un algoritmo de ordenamiento se dice *estable* si los elementos que "empatan" se ubican en la salida en el mismo orden en el que se encontraban en la entrada. ¿Cuáles de los algoritmos de ordenamiento conocidos se pueden hacer estables?

2.1.2. Clase T10: Divide & Conquer

- Completar la siguiente tabla para los siguientes algoritmos que reciben como entrada un arreglo de tamaño n y proceden haciendo Divide & Conquer.

	Número de llamados recursivos	Costo de dividir	Costo de combinar	Costo total
Búsqueda binaria	1	...	...	...
Mergesort	...	$O(1)$	...	...
Quicksort	...	...	$O(1)$	...

2.2. Ordenamiento

Ejercicio 1:
Se tiene un arreglo de n números naturales que se quiere ordenar por frecuencia, y en caso de igual frecuencia, por su valor. Por ejemplo, a partir del arreglo [1, 3, 1, 7, 2, 7, 1, 7, 3] se quiere obtener [1, 1, 1, 7, 7, 7, 3, 3, 2]. Describa un algoritmo que realice el ordenamiento descrito, utilizando las estructuras de datos intermedias que considere necesarias. Calcule el orden de complejidad temporal del algoritmo propuesto.

Ejercicio 2:
Se desea ordenar los datos generados por un sensor industrial que monitorea la presencia de una sustancia en un proceso químico. Cada una de estas mediciones es un número entero positivo. Dada la naturaleza del proceso se sabe que, dada una secuencia de n mediciones, a lo sumo $\lfloor \sqrt{n} \rfloor$ valores están fuera del rango $[20, 40]$.

Proponer un algoritmo $O(n)$ que permita ordenar ascendentemente una secuencia de mediciones y justificar la complejidad del algoritmo propuesto.

Ejercicio 3:
Se tiene un arreglo $A[1 \dots n]$ de T, donde T son tuplas $\langle c_1 : nat \times c_2 : string[\ell] \rangle$ y los $string[\ell]$ son *strings* de longitud máxima ℓ. Si bien la comparación de dos *nat* toma $O(1)$, la comparación de dos $string[\ell]$ toma $O(\ell)$. Se desea ordenar A en base a la segunda componente y luego la primera.

1. Escriba un algoritmo que tenga complejidad temporal $O(n\ell + n\log(n))$ en el peor caso. Justifique la complejidad de su algoritmo.
2. Suponiendo que los naturales de la primer componente están acotados, adapte su algoritmo para que tenga complejidad temporal $O(n\ell)$ en el peor caso. Justifique la complejidad de su algoritmo.

Ejercicio 4:
Se tiene un arreglo de enteros no repetidos $A[1..n]$, tal que se sabe que para todo i hay a lo sumo i elementos mas chicos que $A[i]$ en todo el arreglo. Dar un algoritmo que ordene el arreglo en $O(n)$.

Ejercicio 5:

1. Dar un algoritmo que ordene un arreglo de n enteros positivos menores que n en tiempo $O(n)$.
2. Dar un algoritmo que ordene un arreglo de n enteros positivos menores que n^2 en tiempo $O(n)$. Pista: Usar varias llamadas al ítem anterior.
3. Dar un algoritmo que ordene un arreglo de n enteros positivos menores que n^k para una constante arbitraria pero fija k.
4. ¿Qué complejidad se obtiene si se generaliza la idea para arreglos de enteros no acotados?

2.3. Dividir y conquistar

Ejercicio 1:
Escriba un algoritmo con dividir y conquistar que determine si un arreglo de tamaño potencia de 2 es *más a la izquierda*, donde "más a la izquierda" significa que:

- La suma de los elementos de la mitad izquierda superan los de la mitad derecha.
- Cada una de las mitades es a su vez "más a la izquierda".

Por ejemplo, el arreglo [8, 6, 7, 4, 5, 1, 3, 2] es "más a la izquierda", pero [8, 4, 7, 6, 5, 1, 3, 2] no lo es.
Intente que su solución aproveche la técnica de modo que complejidad del algoritmo sea estrictamente menor a $O(n^2)$.

Ejercicio 2:
Suponga que se tiene un método *potencia* que, dada un matriz cuadrada A de orden 4×4 y un número n, computa la matriz A^n. Dada una matriz cuadrada A de orden 4×4 y un número natural n que es potencia de 2 (i.e., $n = 2^k$ para algun $k \geq 1$), desarrollar, utilizando la técnica de dividir y conquistar y el método *potencia*, un algoritmo que permita calcular

$$A^1 + A^2 + \ldots + A^n.$$

Calcule el número de veces que el algoritmo propuesto aplica el método *potencia*. Si no es estrictamente menor que $O(n)$, resuelva el ejercicio nuevamente.

Ejercicio 3:
Dado un árbol binario cualquiera, diseñar un algoritmo de dividir y conquistar que devuelva la máxima distancia entre dos nodos (es decir, máxima cantidad de ejes a atravesar). El algoritmo no debe hacer recorridos innecesarios sobre el árbol.

Ejercicio 4:
La cantidad de parejas en desorden de un arreglo $A[1 \dots n]$ es la cantidad de parejas de posiciones $1 \leq i < j \leq n$ tales que $A[i] > A[j]$. Dar un algoritmo que calcule la cantidad de parejas en desorden de un arreglo y cuya complejidad temporal sea estrictamente mejor que $O(n^2)$ en el peor caso. **Hint:** Considerar hacer una modificación de un algoritmo de sorting.

3. Ejercicios obligatorios del laboratorio

IMPORTANTE: Los ejercicios de esta sección se dividen en TPs (grupales, grupos de 4 integrantes) y talleres (individuales), y serán calificados. Cada instancia podrá estar aprobada, o será devuelta para incorporar correcciones. Para la resolución de los TPs se espera que trabajen grupalmente y consulten todas sus dudas. No está permitido compartir soluciones detalladas entre grupos de trabajo distintos. En el caso de los talleres, si bien se permite que discutan ideas grupalmente, cada entrega debe ser individual y todo el código entregado debe ser de producción propia.

3.1. TP 4 — fecha de entrega: coordinar con su corrector/a; última fecha: miércoles 9/12

El objetivo de este TP es implementar en C++ todos los módulos correspondientes al diseño presentado en el TP3. El código que entreguen debería respetar el diseño propuesto en el TP 3 de la manera más fiel posible. Obviamente se permite y se espera que corrijan todos los potenciales *bugs* que puedan llegar a encontrar en el diseño. Las implementaciones deben cumplir con las complejidades definidas en su solución del TP 3, incluyendo las restricciones de complejidad establecidas en el enunciado.

3.1.1. Código producido

La resolución debe tener un archivo `.h` y `.cpp` por cada módulo del TP 3 (o eventualmente un archivo `.hpp` si se trata de un módulo paramétrico que se implemente con `templates`). Estos archivos deberán ubicarse en el directorio `src`, respetando el esqueleto disponible en la página de la materia.

3.1.2. Código de la cátedra y tests

Como parte del enunciado, la cátedra provee un **esqueleto de TP4**. El esqueleto tiene la misma estructura de directorio que los talleres, con un directorio `src` para los archivos fuente y un directorio `tests` para el código de los tests. Proveemos un conjunto de casos de test que deberán ser pasados con éxito. Además, **deben escribir sus propios test de unidad** para cada una de las clases que implementen.
El archivo `src/aed2_Juego.{h,cpp}` provisto como parte del esqueleto del TP incluye la interfaz para el módulo JUEGO que se utiliza en los tests. Deben completar estos archivos agregando instancias de las clases diseñadas por ustedes en la parte privada de la clase `aed2_Juego`, e implementando los métodos de forma tal que utilicen la interfaz provista por sus propios módulos.
El archivo `src/Tipos.h` define algunos tipos auxiliares y renombres de tipos (como el tipo `Coordenada` y el tipo `PuntoCardinal`).
La adaptación de la interfaz de sus módulos a los requeridos en `aed2_Juego` puede conllevar operaciones con un costo no inmediato (ej. copiar un conjunto a una lista, recorrer un diccionario, etc.). Los requisitos de complejidad a cumplir aplican solamente a las funciones de la interfaz de los módulos. Los costos asociados a la traducción de su interfaz a la nuestra no tienen restricciones.

Importante: sugerimos no implementar la lógica del juego en la clase `aed2_Juego`, sino hacerlo en una clase independiente `Juego` que respete el diseño hecho por ustedes en el TP3. La clase `aed2_Juego` únicamente debe hacer las veces de "fachada" que delega todos los mensajes que recibe a la clase `Juego`. Así, la implementación de todos los métodos de la clase `aed2_Juego` debería ser breve (ej. una o dos líneas).

El esqueleto disponible en la página de la materia cuenta con funcionalidad para correr el juego con una interfaz gráfica (para poder jugarlo(!)). No es obligatorio que logren compilar el TP con interfaz gráfica, pero puede ser instructivo y divertido intentarlo.

3.1.3. Módulos básicos

Pueden utilizar las siguientes clases de la STL de C++ para los respectivos módulos básicos:

Módulo	Clase
Lista Enlazada	`std::list`
Pila	`std::stack`
Cola	`std::queue`
Vector	`std::vector`
Diccionario Lineal	`std::map`
Conjunto Lineal	`std::set`

Entrega

Para la entrega deben hacer `commit` y `push` de todo el esqueleto, incluyendo el código fuente (`*.cpp`, `*.h`, `*.hpp`), los tests, y el archivo `CMakeLists.txt` en el repositorio **grupal** en el directorio `tpg4/`. No incluir los archivos binarios generados por el compilador (`*.o`, `*.a`, `*.exe`) en el repositorio.

3.2. Taller: Diccionario sobre Trie

En este taller se debe implementar un diccionario sobre Trie para un tipo paramétrico `T`. El enunciado corresponde al del taller de diccionario sobre Trie en L08.

Entrega

La entrega deberá consistir en agregar, commitear y pushear a su repositorio de trabajo indivudal el contenido del directorio con la ejercitación del taller (que contiene el archivo `CMakeLists.txt` y los directorios `src` y `tests`) al directorio `g4/taller`. Lo importante es que dentro del último (`g4/taller`) puedan encontrarse los archivos fuente del taller: `string_map.h` y `string_map.hpp`.