Guía 2

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 T03: Complejidad

- En 1969, el matemático alemán Volker Strassen propuso un algoritmo que calculaba productos de matrices de $n \times n$ con complejidad temporal $O(n^{2.807})$. Un procesador típico hoy en día puede llegar a ser 100.000 veces más rápido que un procesador típico de 1969. ¿Cómo se ve afectada la complejidad del algoritmo por esta diferencia?
- El algoritmo $\mathcal{A}$ recibe como entrada una lista de n números y la procesa en tiempo lineal. Más precisamente, su complejidad temporal, tanto en mejor como en peor caso, es $\Theta(n)$. Roque ejecuta el algoritmo $\mathcal{A}$ sobre una lista de 100 elementos y observa que tarda 100 segundos (con un margen de error de 1 décima de segundo). ¿Qué puede decirse sobre lo que observaría Roque si ejecutara el mismo algoritmo sobre una lista de 200 elementos?
- El algoritmo $\mathcal{B}$ recibe una lista de números y la procesa en tiempo constante, es decir, en $O(1)$. Roque corre el algoritmo $\mathcal{B}$ sobre la misma lista de 100 elementos. ¿Qué puede decirse sobre lo que observa Roque?
- El algoritmo $\mathcal{C}$ recibe un árbol de n nodos y lo procesa de alguna manera. ¿Cuáles de las siguientes combinaciones son posibles y cuáles imposibles?
  1. La complejidad temporal de $\mathcal{C}$ es $O(1)$ en mejor caso y $O(n)$ en peor caso.
  2. La complejidad temporal de $\mathcal{C}$ es $O(n^2)$ en mejor caso y $\Omega(n)$ en peor caso.
  3. La complejidad temporal de $\mathcal{C}$ es $\Omega(2^n)$ en mejor caso y $O(n)$ en peor caso.

2.1.2. Clase T04: Diseño

- Recordemos que el **principio de abstracción** afirma que el usuario de un módulo sólo puede interactuar con él a través de su interfaz ("parte pública"), en tanto que no debe acceder a su representación ("parte privada"). ¿Para qué puede servir considerar diferentes representaciones de un mismo módulo, si el usuario nunca va a poder acceder a ellas?
- Supongamos que el módulo Lista está representado sobre un arreglo. El usuario del módulo, ¿puede usar el hecho de que la complejidad temporal de acceder al i-ésimo elemento de la lista es $O(1)$?
- Supongamos que el módulo Conjunto(Nat) se representa sobre una lista sin repetidos. Roque propone cambiar la representación para usar un arreglo ordenado de menor a mayor. ¿Qué impacto tiene este cambio en la interfaz del módulo? ¿Qué impacto tiene este cambio en la representación y algoritmos del módulo? ¿Qué impacto tiene este cambio en los demás módulos (usuarios del módulo Conjunto(Nat))? ¿Cómo se modifica el invariante de representación? ¿Cómo se modifica la función de abstracción?
- ¿Cuáles de las siguientes situaciones son normales y cuáles necesariamente implican un error de programación?
  - El programador del módulo rompe el invariante de representación de una instancia en una función **privada** y **no** lo reestablece al final de la función.

- Igual que arriba, pero **sí** reestablece el invariante.
- El programador del módulo rompe el invariante de representación de una instancia en una función **pública** (de la interfaz) y **no** lo reestablece al final de la función.
- Igual que arriba, pero **sí** reestablece el invariante.

¿Cómo podría hacer el usuario de un módulo para romper el invariante de representación de la instancia?

2.1.3. Clase T05: Diccionarios sobre ABBs y AVLs

- Si n es la cantidad de nodos de un árbol **perfectamente balanceado** y h es la altura del árbol. ¿Cuáles de las siguientes afirmaciones son ciertas?:

$$h \in O(n) \qquad h \in \Omega(n) \qquad h \in O(\log n) \qquad h \in \Omega(\log n)$$

- Igual que arriba, pero si el árbol es un árbol **arbitrario**, no necesariamente balanceado.
- Igual que arriba, pero si el árbol cumple con el invariante de balanceo de un **AVL**.

2.2. Complejidad

Ejercicio 1

Sean $f, g : \mathbb{N} \to \mathbb{N}$, y supongamos que está definido el límite $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = \ell \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$. Probar que:

a) $0 < \ell < +\infty$ si y sólo si $f \in \Theta(g)$.
b) $\ell = +\infty$ si y sólo si $f \in \Omega(g)$ y $f \notin O(g)$.
c) $\ell = 0$ si y sólo si $f \in O(g)$ y $f \notin \Omega(g)$.

Recordar las definiciones de límite:

- $\lim_{n\to+\infty} a_n = \ell \in \mathbb{R}$ si $\forall\, \varepsilon > 0.\, \exists n_0 \in \mathbb{N}.\, \forall n > n_0.\, |a_n - \ell| < \varepsilon$.
- $\lim_{n\to+\infty} a_n = +\infty$ si $\forall\, M > 0.\, \exists n_0 \in \mathbb{N}.\, \forall n > n_0.\, a_n > M$.

Ejercicio 2

Determinar el orden de complejidad temporal de peor caso de los siguientes algoritmos, asumiendo que todas las operaciones sobre arreglos y matrices toman tiempo $O(1)$.
La complejidad se debe calcular en función de una medida de los parámetros de entrada, por ejemplo, la cantidad de elementos en el caso de los arreglos y matrices y el valor en el caso de parámetros naturales.

Sumatoria, que calcula la sumatoria de un arreglo de enteros:

```
1: function Sumatoria(arreglo A)
2:     int i, total;
3:     total := 0;
4:     for i := 0 ... Long(A) - 1 do
5:         total := total + A[i];
6:     end for
7: end function
```

SumatoriaLenta, que calcula la sumatoria de n, definida como la suma de todos los enteros entre 1 y n, de forma poco eficiente:

```
1: function SumatoriaLenta(natural N)
2:     int i, total;
3:     total := 0;
4:     for i := 1 ... n do
5:         for j := 1 ... i do
6:             total := total + 1;
7:         end for
8:     end for
9: end function
```

InsertionSort, que ordena un arreglo pasado como parámetro:

```
1: function InsertionSort(arreglo A)
2:     int i, j, valor;
3:     for i := 0 ... Long(A) - 1 do
4:         valor := A[i];
5:         j := i - 1;
6:         while j ≥ 0 ∧ a[j] > valor do
7:             A[j+1] := A[j];
8:             j := j - 1;
9:         end while
10:        A[j+1] := valor;
11:    end for
12: end function
```

BúsquedaBinaria, que determina si un elemento se encuentra en un arreglo, que debe estar ordenado:

```
1: function BúsquedaBinaria(arreglo A, elem valor)
2:     int izq := 0, der := Long(A) - 1;
3:     while izq < der do
4:         int medio := (izq + der) / 2;
5:         if valor < A[medio] then
6:             der := medio;
7:         else
8:             izq := medio;
9:         end if
10:    end while
11:    return A[izq] = valor;
12: end function
```

ProductoMat, que dadas dos matrices A (de $p \times q$) y B (de $q \times r$) devuelve su producto AB (de $p \times r$):

```
1: function ProductoMat(matriz A, matriz B)
2:     int fil, col, val, colAFilB;
3:     matriz res(Filas(A), Columnas(B));
4:     for fil := 0 ... Filas(A) - 1 do
5:         for col := 0 ... Columnas(B) - 1 do
6:             val := 0;
7:             for colAFilB := 0 ... Columnas(A) - 1 do
8:                 val := val + (A[fil][colAFilB] * B[colAFilB][col]);
9:             end for
10:            res[fil][col] := val;
11:        end for
12:    end for
13:    return res;
14: end function
```

Ejercicio 3

Para cada una de las siguientes afirmaciones, decida si son verdaderas o falsas y justifique su decisión.

- $O(n^2) \cap \Omega(n) = \Theta(n^2)$
- $\Theta(n) \cup \Theta(n \log n) = \Omega(n \log n) \cap O(n)$
- Existe una $f : \mathbb{N} \to \mathbb{N}$ tal que para toda $g : \mathbb{N} \to \mathbb{N}$ se cumple que $g \in O(f)$.
- Sean $f, g : \mathbb{N} \to \mathbb{N}$, entonces se cumple que $O(f) \subseteq O(g)$ o $O(g) \subseteq O(f)$ (es decir, el orden sobre funciones dado por la inclusión de la O es total).

2.3. Invariante de representación y función de abstracción

Ejercicio 1

Se propone la siguiente estructura para representar a los polinomios que tienen a lo sumo grado n (ver TAD en Práctica 1):

polinomio **se representa con** estr, donde estr es tupla

⟨*grado*: nat,
coef: array[0 . . . n] de nat⟩

Se pide:

1. Definir el invariante de representación y la función de abstracción.
2. Escribir la interface completa y el algoritmo para la función evaluar.

Ejercicio 2

Los palíndromos son aquellas palabras que pueden leerse al derecho o al revés. El siguiente TAD describe a los palíndromos:

TAD PALÍNDROMO(α)

observadores básicos
ver : palindromo(α) $\longrightarrow$ secu(α)

generadores
medio : α $\longrightarrow$ palindromo(α)
medioDoble : α $\longrightarrow$ palindromo(α)
agregar : $\alpha \times$ palindromo(α) $\longrightarrow$ palindromo(α)

axiomas
ver(medio(a)) $\equiv$ a • <>
ver(medioDoble(a)) $\equiv$ a • a • <>
ver(agregar(a,p)) $\equiv$ a • (ver(p) $\circ$ a)

Fin TAD

Se propone la siguiente estructura de representación:

palindromo **se representa con** estr, donde estr es tupla

⟨*long*: nat,
palabra: secu(α)⟩

dónde *palabra* representa el palíndromo completo.
Se pide:

- Definir el invariante de representación y la función de abstracción.
- Escribir la interface y el algoritmo para la función ver.
- Rehacer los ítems anteriores si el campo *palabra* en lugar de la palabra completa guardamos sólo la mitad inicial de la palabra (redondeando hacia arriba).

Ejercicio 3

El salón Alta Fiesta se encuentra últimamente en dificultades para coordinar el desarrollo de las reuniones que se realizan en sus facilidades. Hoy en día las fiestas se desarrollan de una manera muy particular. Los invitados llegan en grupos, por lo general numerosos, aunque también a veces de una sola persona. Que un invitado llegue sin un regalo está considerado una falta grave a las buenas costumbres. Tanto es así que a dichos individuos no se les permite el ingreso a las fiestas. Lo que sí se permite es que los invitados hagan regalos en grupo: los invitados que llegan en grupo traen un único regalo de parte de todos. Como es habitual, sólo se permite la entrada a la fiesta a aquellas personas que han sido invitadas.
Al ingresar un grupo de invitados a la fiesta, éste se identifica con un nombre: por ejemplo "Los amigos de la secundaria" o "La familia de la novia". Este nombre se usa por ejemplo para los juegos grupales que van a hacer los animadores durante la fiesta. Igualmente, dado que el comportamiento de las personas en masa no siempre es civilizado, se quiere poder saber de manera eficiente cuál es el nombre del grupo más numeroso para poder seguirle el rastro.
Además, se desea tener un registro de todos los regalos, junto con el grupo de personas que lo hicieron, y se desea conocer en todo momento qué personas se encuentran ya en la fiesta.
Se nos ha encomendado realizar un sistema que permite llevar el control del estado de la fiesta en un momento dado. La cátedra ha realizado la especificación del sistema y ha armado una estructura de representación para el diseño del mismo. Notar que esta es una de las posibles resoluciones del enunciado, puede haber otras dependiendo de la interpretación.

TAD AltaFiesta

observadores básicos

invitadosPendientes : AltaFiesta $\longrightarrow$ conj(Persona)
regalos : AltaFiesta $\longrightarrow$ conj(Regalo)
personasPorRegalo : AltaFiesta a × Regalo r $\longrightarrow$ conj(Persona) {r ∈ regalos(a)}
grupoMasNumeroso : AltaFiesta $\longrightarrow$ Grupo

generadores

iniciarFiesta : conj(Persona) *invitados* $\longrightarrow$ AltaFiesta
lleganInvitados : AltaFiesta a × conj(Persona) c × Grupo g × Regalo r $\longrightarrow$ AltaFiesta
{c ⊆ invitadosPendientes(a) ∧ r ∉ regalos(a)}

axiomas

...

Fin TAD

altafiesta **se representa con** estr, donde estr es tupla

⟨*invitados*: conj(persona),
presentes: cola(persona),
grupoDe: dicc(grupo, conj(persona)),
regaloDeGrupo: dicc(grupo, regalo),
grupoMasNumeroso: grupo⟩

grupo, persona y regalo **son** string

Informalmente, esta representación cumple las siguiente propiedades:

- En *invitados* están todos los invitados a la fiesta, incluyendo también a aquellos que ya llegaron.
- En *presentes* están los invitados que ya llegaron a la fiesta.
- En *grupoDe* se encuentra, para cada identificador de grupo i, las personas que al llegar agrupadas se identificaron como i.

- En *regaloDeGrupo* se encuentra qué regalo trajo cada grupo.
- *grupoMasNumeroso* contiene al identificador del grupo de más personas. En caso de empate, contiene al lexicográficamente menor de los empatados (se asume que la función $<_{string}$ está definida).

Se pide:

1. Realizar el invariante de representación del módulo. Escribirlo en lenguaje formal y en castellano. Para que quede claro cuáles enunciados informales hacen referencia a cuáles enunciados formales se recomienda numerar cada punto del invariante para luego poder hacer referencia al mismo.
2. Escribir la función de abstracción. De considerarse necesario, explicarla en castellano.
3. Escribir una versión imperativa de la función llegaGrupo marcando claramente los puntos de su programa en que alguna parte del invariante de representación se rompe, indicando a qué parte se refiere, y también aquellos puntos donde éste se reestablece.

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 2 — fecha de entrega: domingo 18 de octubre

En este trabajo se extenderá la especificación del el juego de ingenio Sokoban con las siguientes modificaciones:

- **Bombas.** Las reglas del juego son las mismas que en el TP 1, con el añadido de la siguiente regla: al principio de un nivel, el jugador cuenta con un cierto número n de bombas. En cualquier momento del juego, el jugador puede arrojar una bomba en la posición donde se encuentra actualmente. Esto tiene el efecto de destruir todas las paredes que se encuentren en la misma fila o en la misma columna que la posición donde se encuentra actualmente el jugador. Las posiciones donde había paredes quedan vacías (sin cajas ni depósitos). Cuando el jugador arroja una bomba, se decrementa en 1 el número de bombas disponibles. Para poder arrojar una bomba, el jugador debe tener al menos una bomba en su poder.

  Notas: las bombas sólo destruyen paredes: no destruyen cajas ni depósitos. Si hay una caja entre la bomba y una pared, la pared se destruye y la caja queda intacta (a pesar de que esto pueda ser contraintuitivo).

- **Deshacer movimientos.** El jugador tiene la opción hacer *undo*, es decir, deshacer la movimiento anterior. El mecanismo de *undo* debe deshacer por completo el último movimiento. Es decir: si el jugador se había movido, debe devolvérselo a la posición anterior. Si había empujado una caja, la caja debe volver a la posición donde se encontraba. Si el jugador arrojó una bomba, las paredes que se hubieran destruido deben volver a encontrarse en el mapa, y el número de bombas debe volver a incrementarse en 1. El jugador puede hacer *undo* un número arbitrario de veces, de tal modo que si hace *undo* n veces se deshagan los últimos n movimientos[3].

  Notas: si el jugador no hizo ningún movimiento, no se puede aplicar la operación de *undo*.

- **Pasar de nivel.** Por último, el juego cuenta con un repositorio de niveles. Cada nivel incluye todos los datos necesarios para jugar (mapa, cantidad de bombas, ubicación inicial del jugador, etc.). El jugador empieza en alguno de todos esos niveles. Cuando el jugador completa un nivel, es decir, cuando ubica las cajas sobre sendos depósitos, ese nivel queda resuelto y el jugador pasa *automáticamente* a alguno de los niveles que no haya sido resuelto aún. Pueden tomar una decisión sobre qué ocurre cuando el jugador completa todos los niveles. El mecanismo para elegir el nivel en el que empieza el jugador y el nivel al que pasa el jugador cuando gana un nivel **no está determinado aún**; se elegirá en el momento del diseño. Tener en cuenta que dar un método concreto sería sobreespecificar.

  Notas: En el repositorio de niveles no puede haber niveles repetidos. Una vez que el jugador pasa de nivel, no puede aplicar *undo* más allá del inicio del nuevo nivel. Es decir, cuando el jugador pasa de nivel se "resetea" el historial de últimos movimientos.

  Sugerencia: consideren especificar el mecanismo de pasaje de nivel en un TAD "superior" que haga uso del TAD en el que se especifica el mecanismo del juego.

Se pide especificar todos estos comportamientos, extendiendo o modificando la especificación del TP 1.

Entrega

Para la entrega deben hacer `commit` y `push` de un único documento digital en formato `pdf` en el repositorio **grupal** en el directorio `tpg2/`. El documento debe incluir la especificación completa del enunciado presentado usando el lenguaje de especificación con TADs de la materia. Se recomienda el uso de los paquetes de LaTeX de la cátedra para lograr una mejor visualización del informe.

[3] Notar que el comando *undo* no cuenta como un movimiento.

3.2. Taller: Lista Doblemente Enlazada

En este taller se debe implementar una lista doblemente enlazada para `int`. El enunciado corresponde al del taller de listas enlazadas presentado en L05.

Entrega

Para la entrega deben hacer `commit` y `push` de los archivos fuente (`.h`, `.cpp`) necesarios en el repositorio **individual** en el directorio `g2/taller/src/`. **Respetar los nombres originales de los archivos**. No es necesario que incluyan los tests en el directorio `g1/taller/tests/` pero pueden hacerlo. No incluir archivos binarios en el repositorio (`*.o`, `*.a`, `*.exe`, etc.).