

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2283 — Diseño y Análisis de Algoritmos 2025-2

Examen

12 de diciembre de 2025

Evaluación: 3 preguntas, cada una con subpreguntas. Cada pregunta vale 6 puntos (+1 punto base). La nota es el promedio de las 3 preguntas.

Forma de Entrega: Además de entregar las hojas con sus respuestas, debe subir una versión digital a Canvas. Al final de la prueba, tiene 10 minutos para subir la imagen de sus respuestas, **en formato vertical**, **cada pregunta por separado**. Es su responsabilidad asegurarse de que todas sus respuestas se suban correctamente al sistema.

Dispositivo: Para digitalizar la prueba se permite el uso de UN dispositivo (teléfono, tablet) el cual debe estar guardado y en silencio durante el desarrollo.

Torpedo: Se permite una hoja tamaño oficio escrita a mano por usted, usando ambos lados. También debe subirse una versión digital a Canvas.

Política de Salidas: Si necesita salir de la sala al baño, debe dejar el celular adelante en la mesa con los ayudantes.

Se sugiere leer todas las preguntas antes de iniciar las respuestas.

Tiempo: 2 horas.

1. Algoritmos Aleatorizados

Sean A y B dos algoritmos aleatorizados de tipo Monte Carlo, los cuales resuelven el mismo problema de decisión (es decir, responden true o false ante una entrada dada). El algoritmo A tiene probabilidad $p > 0$ de fallar cuando responde false, y su respuesta es siempre correcta cuando responde true. Por otro lado, el algoritmo B tiene probabilidad $q > 0$ de fallar cuando responde true, y su respuesta es siempre correcta cuando responde false.

1. **[3.0 pts.]** Muestre cómo combinar A y B en un único algoritmo que permita resolver el problema, de tal manera que la probabilidad de error del nuevo algoritmo sea estrictamente menor a $\min\{p, q\}$ (es decir, la probabilidad de error debe ser estrictamente menor a la de los dos algoritmos originales). Si T_A es el tiempo de ejecución del algoritmo A, y T_B el tiempo de ejecución de B, el tiempo de ejecución del algoritmo resultante debería ser a lo más $T_A + T_B$.

   Solución:

Algoritmo 1: AB(X)

```
1  begin
2      if p < q then
3          if not B(X) then
4              return false
5          else
6              return A(X)
7      else
8          if A(X) then
9              return true
10         else
11             return B(X)
```

Rúbrica: si ponen sólo el caso en que primero usa A y luego B en caso de que A respondió false, el puntaje no debe superar los 0.3 puntos, ya que esa solución sería la misma de las guías. El algoritmo sólo podía hacer una invocación a A y una a B, de acuerdo al enunciado. No podían repetir las invocaciones a ninguno de ellos. En esos casos, aplicar descuentos con criterio.

2. **[1.5 pts.]** ¿Cuál es la probabilidad de fallar que tiene el algoritmo propuesto? ¿En qué casos falla?

   Solución:

   - Si $p < q$, el algoritmo puede fallar al responder false (porque A puede fallar en ese caso). La probabilidad de fallo es $(1 - q) \cdot p = p - pq$.

- Si $q \leq p$, el algoritmo puede fallar al responder true (porque B puede fallar en ese caso). La probabilidad de fallo es $(1-p) \cdot q = q - pq$.

3. [**1.5 pts.**] Demuestre formalmente que la probabilidad de error del algoritmo propuesto es estrictamente menor a $\min\{p, q\}$.

   Solución: Note que la probabilidad de error dada en el punto anterior puede ser expresada como $\min\{p, q\} - pq$. Dado que $p > 0$ y $q > 0$, $pq > 0$ y por lo tanto $\min\{p, q\} - pq < \min\{p, q\}$.

2. Algoritmos en Teoría de Números

Usted forma parte de un equipo forense de informática que se encuentra auditando un sistema de autenticación, el cual genera tokens de autenticación de sesión. Para eso, emplea un generador de números pseudo aleatorios que produce una secuencia de valores $X_0, X_1, \ldots$, en el rango $[0, n-1]$ de la siguiente manera:

$$X_i = (a \cdot X_{i-1} + c) \mod n, \quad i > 0$$

en donde a es un multiplicador tal que $0 < a < n$, c es un incremento tal que $0 \leq c < n$, y X_0 es un valor dado. Note que cada X_i se representa usando $\lceil \lg_2 n \rceil$ bits.
Su equipo forense ha interceptado en el tráfico de la red un token X_i, para $i > 0$, y además han descubierto los valores de a, c, y n. Para validar una vulnerabilidad de predicción de sesiones, su objetivo es recuperar el token previamente generado X_{i-1}.

a) **[2.5 pts.]** Diseñe un algoritmo eficiente, usando peseudocódigo o el lenguaje de su preferencia, que determine X_{i-1} usando los datos proporcionados por el equipo forense. Si lo necesita, puede usar cualquiera de los algoritmos vistos en clase como subrutina.

Solución:
Primero debemos manipular la ecuación. Todo número es congruente a su propio resto, por lo tanto tenemos:

$$\begin{aligned} X_i &= (a \cdot X_{i-1} + c) \mod n \\ X_i &\equiv (a \cdot X_{i-1} + c) \mod n \\ X_i - c &\equiv (a \cdot X_{i-1} + c - c) \mod n \\ X_i - c &\equiv (a \cdot X_{i-1}) \mod n \end{aligned}$$

Por simetría de la congruencia,

$$a \cdot X_{i-1} \equiv (X_i - c) \mod n$$

Aquí "el truco", debemos deshacernos del a para poder despejar X_{i-1}, aquí podemos calcular el inverso modular de a (lo llamaremos d) para luego multiplicar en ambos lados.

Entonces, con Euclides extendido podemos calcular $a \cdot d \equiv 1 \mod n$.

Finalmente, tenemos:

$$\begin{aligned} d \cdot a \cdot X_{i-1} &\equiv d \cdot (X_i - c) \mod n \\ X_{i-1} &\equiv d \cdot (X_i - c) \mod n \end{aligned}$$

El algoritmo quedaría de la siguiente forma:

Algoritmo 2: TokenAnterior$(curr, a, c, n)$

```
1 d ← Inverso(a, n) // Euclides extendido
2 prev ← d · (curr − c)  mod n
3 return prev
```

Rúbrica: [0.5 puntos], relaciona correctamente la ecuación con la relación de congruencia, **[1.0 punto]** por identificar la utilización del inverso modular de a, **[1.0 punto]** por la utilización y de forma correcta del algoritmo Euclides extendido.

Aquellas soluciones que asumían conocer X_0 e iteraban hasta encontrar X_i, ***reciben 0 pts****, no hay nada que se pueda evaluar de aritmética modular.*

b) **[2.5 pts.]** Demuestre que el algoritmo diseñado es correcto (puede ser respondido en la parte a) si lo desea).
Solución: Respondido en el item anterior.

Rúbrica: Se puede demostrar de diferentes formas la correctitud de forma directa del algoritmo, la idea es que utilizarán bien las propiedades de aritmética modular. Se sugiere descontar **[0.7 puntos]** por propiedad mal aplicada, siempre y cuando el item a) resuelva el problema.

c) **[0.5 pts.]** Indique cual es la condición que debe cumplir la constante a para que el algoritmo pueda responder el valor de X_{i-1}.
Solución: a debe tener inversa en módulo n, es decir $MCD(a, n) = 1$.
Rúbrica: basta con que digan que a debe tener inversa en módulo n, **[0.25 puntos]** de descuento si no menciona n

d) **[0.5 pts.]** Indique usando notación asintótica el tiempo de ejecución de su algoritmo. Justifique brevemente su respuesta.
Solución: el tiempo de ejecución esta dado por Euclides extendido, por lo que el tiempo es $O(\lg n)$.

3. Preguntas Conceptuales

Responda las siguientes preguntas usando respuestas cortas, pero formales y precisas.

1. **[2.0 pts.]** Sea $a \cdot x + b \cdot y = c$ una recta en el plano, con $a, b, c \in \mathbb{Z}$, $a \neq 0$ y $b \neq 0$.

   a) Suponiendo que $\mathsf{MCD}(a, b) \mid c$, indique (mediante una fórmula) todos los puntos (x, y) de la recta tal que $x, y \in \mathbb{Z}$ y se cumple que $x \geq 0$, $y \geq 0$.
   Solución: la idea es modelar la recta como una ecuación diofántica. Dado que $\mathsf{MCD}(a, b) \mid c$, entonces se usa el algoritmo de Euclides extendido para obtener (g, x_g, y_g), en donde $g = \mathsf{MCD}(a, b)$. Luego, definimos $x_0 = x_g \frac{c}{g}$ e $y_0 = y_g \frac{c}{g}$. Luego, para $k \in \mathbb{Z}$, los puntos (x, y) de la recta que tienen ambas coordenadas enteras se obtienen mediante la ecuación vista en clases $x = x_0 + k\frac{b}{g}$ e $y = y_0 - k\frac{a}{g}$. Luego, queremos $x \geq 0$ e $y \geq 0$. Por lo tanto, $x = x_0 + k\frac{b}{g} \geq 0$ implica $k \geq -x_0\frac{g}{b}$. Haciendo lo mismo para y queda $k \leq y_0\frac{g}{a}$. Por lo tanto, $-x_0\frac{g}{b} \leq k \leq y_0\frac{g}{a}$.

   Rúbrica: distribuir los 2.0 puntos de manera uniforme entre las partes de la solución. Notar que esta pregunta se lleva todos los puntos, ya que decidimos eliminar la parte b, leer más abajo). Las partes a considerar en la evaluación son: usar Euclides extendido, obtener x_0 e y_0, definir las soluciones generales x e y para $k \geq 0$, acotar las soluciones a $x \geq 0$ e $y \geq 0$.

   b) Argumente por qué la cantidad de puntos que cumplen las condiciones del punto anterior siempre es finita.
   Solución: Como puede verse en el resultado anterior, k queda acotada en un intervalo finito, por lo tanto la cantidad de soluciones que pueden obtenerse a partir de esa fórmula es finita. Una forma geométrica de verlo es que al cumplirse $a \neq 0$ y $b \neq 0$, la recta nunca es paralela a alguno de los ejes (x o y). Por lo tanto, siempre corta a ambos ejes en los puntos $(c/a, 0)$ y $(0, c/b)$. Los puntos (x, y) con coordenadas enteras ≥ 0 que se buscan son los que están contenidos en el segmento de recta entre los puntos $(c/a, 0)$ y $(0, c/b)$.

   Rúbrica: Decidimos eliminar esta pregunta, ya que faltaron algunas restricciones para a y b para poder haber demostrado lo que se pedía. De todas maneras, si no hicieron la primera parte pero sí la segunda, se consideraba (Nota post evaluación: no hubo casos de ese tipo). Lo importante es que muestren haber entendido el problema y que muestren dominio de la materia.

2. **[2.0 pts.]** Sean $a, b, n \in \mathbb{Z}$ tal que su representación binaria usa L bits. ¿Por qué para calcular a^b mod n de manera eficiente no se calcula primero a^b usando el siguiente algoritmo rápido:

$$a^b = \begin{cases} 1, & b = 0 \\ (a^{b/2})^2, & b > 0, b \text{ es par} \\ a(a^{(b-1)/2})^2, & b > 0, b \text{ es impar} \end{cases}$$

para, una vez obtenido ese valor, aplicarle $\bmod n$? ¿Cómo resuelve eso el algoritmo $\mathsf{EXP}(a, b, n)$ estudiado en clases?
Solución: Dado que las representaciones binarias de a y b usand L bits, se pueden representar números enteros $\leq 2^L - 1$. Así, a^b necesita $\lg_2 a^b = b \lg_2 2 \leq 2^L \cdot L$ bits, lo cual es exponencial en L y por lo tanto ineficiente ya que más allá de que el algoritmo indicado arriba produce una cantidad $O(L)$ de invocaciones, debe multiplicar números de tamaño exponencial en L, haciendo que el tiempo de ejecución se vuelva al menos exponencial. El algoritmo estudiado en clases resuelve el problema calculando $a^b \bmod n = (a^{b/2} \bmod n)^2 \bmod n$, por lo tanto siempre manipula enteros de L bits. Por lo tanto, el tiempo es $O(L)$.

Rúbrica: distribuir los puntos con criterio.

3. **[2.0 pts.]** Dado el siguiente algoritmo aleatorizado estudiado en clases para el chequeo de primalidad de un número n:

TestPrimalidad3(n, k)
sea $a_1, \ldots, a_k$ una secuencia de números elegidos de
manera uniforme e independiente desde $\{1, \ldots, n-1\}$
for $i := 1$ **to** k **do**
if MCD$(a_i, n) > 1$ **then return** COMPUESTO
else
if EXP$(a_i, n-1, n) \neq 1$
then return COMPUESTO
return PRIMO

a) ¿En qué casos el algoritmo falla en responder correctamente?
Solución: el algoritmo falla al responder que el número es primo, siendo que es compuesto. El algoritmo siempre es correcto cuando responde COMPUESTO.

b) ¿Qué condiciones cumplen los números (si es que existen) que harán fallar el test **siempre**, en todas las invocaciones que se hagan con ese número, sin importar los a_i que se elijan? Argumente su respuesta.
Solución: El algoritmo falla con números compuestos a los que marca como primos. No hay otro caso de error. Sin embargo, no existe ningún número compuesto que haga fallar el algoritmo en todos los casos. El algoritmo falla si n es un número de Charmichael y los a_i elegidos son coprimos de n (en ese caso, $\mathsf{MCD}(a_i, n) = 1$). Sin embargo, para que un número de Carmichael haga fallar siempre al algoritmo, debería ser coprimo con todo número a_i posible elegido. Pero eso es imposible, ya que un número de Carmichael es un número compuesto, lo que significa que existe al menos un divisor $1 < d < n$. Eso implica que si un a_i es ese divisor, el test $\mathsf{MCD}(a_i, n) > 1$ va a dar verdadero y responderá que es compuesto, lo que es correcto.

Glosario de Propiedades Útiles

Media y varianza de una variable aleatoria X. Dada una variable aleatoria X, definimos su *media* (o esperanza) como

$$E(X) = \sum_{x \in A} x \cdot \Pr\{X = x\},$$

en donde A es el conjunto de valores que puede tomar X. También definimos el k-ésimo momento de X, para $k \geq 1$ como

$$E(X^k) = \sum_{x \in A} x^k \cdot \Pr\{X = x\}.$$

Luego, se define la varianza de X como:

$$\mathrm{Var}(X) = E((X - E(X))^2) = E(X^2) - E(X)^2.$$

Linealidad de la Media y la Varianza. Sean $V_1, \ldots, V_n$ variables aleatorias idénticamente distribuidas. Entonces se cumple:

$$E\left(\sum_{i=1}^{n} V_i\right) = \sum_{i=1}^{n} E(V_i).$$

Si además se cumple que las variables $V_1, \ldots, V_n$ son independientes, se tiene

$$\mathrm{Var}\left(\sum_{i=1}^{n} V_i\right) = \sum_{i=1}^{n} \mathrm{Var}(V_i).$$

Distribución Geométrica. Si X es una variable aleatoria usada para medir el número total de intentos necesarios para obtener el primer éxito en una sucesión de eventos con distribución Bernoulli, cada uno de ellos con probabilidad de éxito p, entonces X tiene distribución geométrica y $\Pr\{X = i\} = (1-p)^{i-1} \cdot p$. Además,

$$E(X) = 1/p$$

y

$$\mathrm{Var}(X) = \frac{1-p}{p^2}.$$

Desigualdad de Markov. Sea X una variable aleatoria no negativa. Para cada $a \in \mathbb{R}^+$ se tiene que

$$\Pr\{X \geq a\} \leq \frac{E(X)}{a}.$$

Desigualdad de Chebyshev. Sea X una variable aleatoria no negativa. Para cada $a \in \mathbb{R}^+$ se tiene que

$$\Pr\{|X - E(X)| \geq a\} \leq \frac{\mathrm{Var}(X)}{a^2}.$$

Teorema de Bayes. Sean A y B eventos, entonces la probabilidad condicional del evento A dado que ocurrió el evento B está dada por:

$$\Pr\{A|B\} = \frac{\Pr\{B|A\} \cdot \Pr\{A\}}{\Pr\{B\}}.$$

Ley de Probabilidad Total. Sean A y B eventos. Entonces:

$$\Pr\{A\} = \Pr\{A|B\} \cdot \Pr\{B\} + \Pr\{A|\overline{B}\} \cdot \Pr\{\overline{B}\}.$$

Propiedades de aritmética modular:

1. $a \equiv b \bmod n$ si y sólo si $a \bmod n = b \bmod n$
2. $a \equiv (a \bmod n) \bmod n$
3. Si $a \equiv b \bmod n$ y $c \equiv d \bmod n$, entonces:

$$\begin{aligned} (a + c) &\equiv (b + d) \bmod n \\ (a \cdot c) &\equiv (b \cdot d) \bmod n \end{aligned}$$

4. Si $a \equiv b \bmod n$, entonces para todo $c \in \mathbb{Z}$ se cumple $a + c \equiv b + c \bmod n$.
5. Si $a \equiv b \bmod n$, entonces para todo entero positivo c se cumple $a \cdot c \equiv b \cdot c \bmod n$.
6. $(a \cdot b) \bmod n = (a \bmod n)(b \bmod n) \bmod n$

Teorema de Fermat:

1. Sea p un número primo. $\forall a \in \{0, \ldots, p-1\}$, entonces $a^p \equiv a \bmod p$.
2. Sea p un número primo. $\forall a \in \{1, \ldots, p-1\}$, entonces $a^{p-1} \equiv 1 \bmod p$.