


GUÍA PRÁCTICA

1. DATOS GENERALES	
Asignatura: Programación Avanzada	Código de la Asignatura: SIS-04212
Carrera: Ingeniería de Sistemas	
Curso: A	Semestre: Cuarto
Contenido Analítico: <ul style="list-style-type: none"> Introducción Sintaxis de Java 	Unidad Temática: Análisis y Diseño de Algoritmos
Docente: Msc. Víctor Rodríguez Estévez	Email: vrodrigueze@doc.emi.edu.bo
Bibliografía a seguir: <ul style="list-style-type: none"> How to Program in Java (Deitel Deitel) 	
Práctica: 1	Título: Introducción al Análisis y Diseño de Algoritmos en Java
Material de Apoyo: Diapositivas, lecturas	Carga horaria: 6

2. OBJETIVO
<ul style="list-style-type: none"> Repasar conocimientos previos de programación (estructuras de control) Repasar conocimientos previos de Estructuras de Datos (Vectores, listas, archivos de texto) Utilizar la sintaxis de java para implementar los algoritmos Utilizar herramientas del IDE para el análisis y la implementación de código Utilizar pruebas unitarias para facilitar en testeo de código.

3. SOFTWARE, SIMULADORES Y/O EQUIPOS	
Detalle	Cantidad
Java SDK 1.8	30
IntelliJ IdeA Community	30

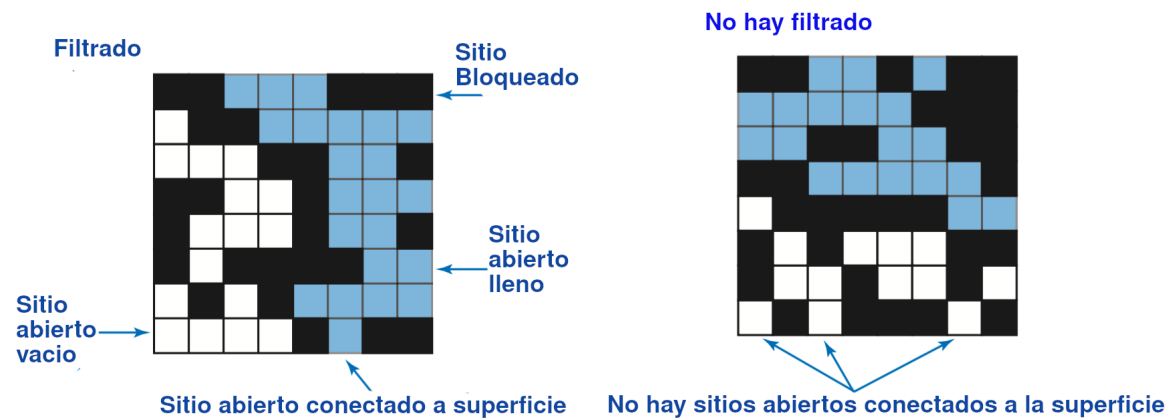
1 Introducción

Dada una superficie porosa en un tanque con agua ¿bajo qué condiciones el agua podrá drenar? Básicamente podemos definir un proceso abstracto, es decir una filtración.

Modelamos un sistema de filtrado utilizando una matriz de sitios $n \times n$. Cada sitio está abierto o bloqueado. Un sitio completo es un sitio abierto que se puede conectar a un sitio abierto en la fila superior a través de una cadena de sitios abiertos vecinos (izquierda, derecha, arriba, abajo).

Decimos que el sistema filtra si hay un sitio completo en la fila inferior, en otras palabras, un sistema se filtra si llenamos todos los sitios abiertos conectados a la fila superior y ese proceso llena algún

sitio abierto en la fila inferior. Los sitios abiertos corresponden a espacios vacíos a través del cual puede fluir el agua, de modo que un sistema que se filtra permite que el agua llene los sitios abiertos, fluyendo de arriba hacia abajo).

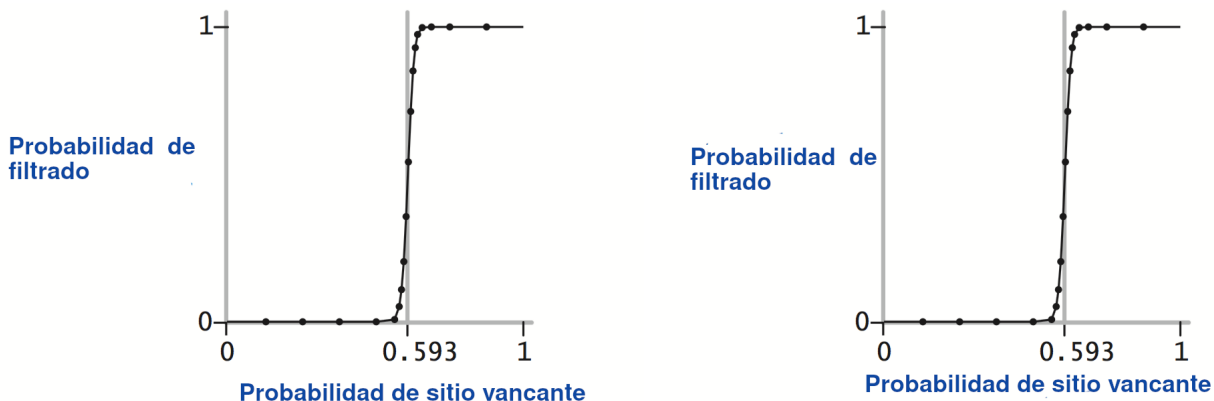


2 Problemas

Unos investigadores están interesados en la siguiente pregunta: si los sitios se configuran de forma independiente para estar abiertos con probabilidad P (y por lo tanto bloqueados tienen probabilidad $1 - p$), ¿cuál es la probabilidad de que el sistema se filtre?

Cuando P es igual a 0, el sistema no se filtra; cuando P es igual a 1, el sistema se filtra.

Los gráficos a continuación muestran la probabilidad de vacancia del sitio p frente a la probabilidad de percolación para una cuadrícula aleatoria de 20 por 20 (izquierda) y una cuadrícula aleatoria de 100 por 100 (derecha).



Cuando n es lo suficientemente grande, hay un valor de umbral p^* tal que cuando $p < p^*$ una cuadrícula aleatoria n por n casi nunca se filtra, y cuando $p > p^*$, una cuadrícula aleatoria n por n casi siempre se filtra. Aún no se ha derivado ninguna solución matemática para determinar el umbral de filtración p^* . Tu tarea es escribir un programa de computadora para estimar p^* .

Tipo de datos para representar filtrado: Para modelar un sistema de filtrado, crea un tipo de datos Filtracion con la siguiente API:

```
public class Filtracion{
```

```
// Crear una grilla n*n, con todos los sitios bloqueados inicialmente
public Filtracion(int n)

// abrir el sitio (fila , columna) si no esta abierto ya.
public void abrir(int fila , int columna)

// esta el sitio (fila , columna) abierto?
public boolean esta_abierto(int fila , int columna)

Esta lleno el sitio (fila , col)?
public boolean esta_lleno(int fila , int columna)

// devuelve el numero de sitios abiertos
public int get_num_abiertos()

// el sistema se filtra?
public boolean filtra()

// probar
public static void main(String [] args)
}
```

Por convención, los índices de fila y columna son números enteros entre 1 y n , donde (1, 1) es el sitio superior izquierdo: Ejecuta una excepción `IllegalArgumentException` si algún argumento está fuera de su rango prescrito. Lanza una `IllegalArgumentException` en el constructor si $n \leq 0$.

Requisitos de desempeño. El constructor debe tomar tiempo proporcional a n^2 ; todos los métodos de instancia deben tomar un tiempo constante más un número constante de llamadas a `union()` y `find()`.

3 Simulación del Monte Carlo

Para estimar el umbral de filtración, considere el siguiente experimento computacional:

- Inicialice todos los sitios que se bloquearán.
- Repita lo siguiente hasta que el sistema se filtre:
 - Elija un sitio uniformemente al azar entre todos los sitios bloqueados.
 - Abre el sitio.
- La fracción de sitios que se abren cuando el sistema se filtra proporciona una estimación del umbral de filtración.

Por ejemplo, si los sitios se abren en una área de 20 por 20 según las instantáneas a continuación, nuestra estimación del umbral de filtración es $204/400 = 0,51$ porque el sistema se filtra cuando se abre el sitio 204.

Repitiendo este experimento de cálculo T veces y promediando los resultados, obtenemos una estimación más precisa del umbral de filtración. Sea x_T la fracción de sitios abiertos en el experimento computacional t . La media muestral \bar{x} proporciona una estimación del umbral de filtración; la desviación estándar de la muestra s ; mide la nitidez del umbral.

- $\bar{x} = \frac{x_1 + x_2 + x_3 + \dots + x_T}{T}$
- $s^2 = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + (x_3 - \bar{x})^2 + \dots + (x_T - \bar{x})^2}{T - 1}$

Suponiendo que T es lo suficientemente grande (digamos, al menos 30), lo siguiente proporciona un intervalo de confianza del 95 % para el umbral de filtración:

$$\left[\bar{x} - \frac{1.96s}{\sqrt{T}}, \bar{x} + \frac{1.96s}{\sqrt{T}}\right]$$

Para realizar esta serie de experimentos computacionales, cree un tipo de datos Experimento con la siguiente API:

```
public class Experimento {  
  
    // realizar ensayos independientes en una cuadrícula n-by-n  
    public Experimento(int n, int trials)  
  
    // media muestral del umbral de filtración  
    public double get_media()  
  
    // desviación estándar de la muestra del umbral de filtración  
    public double get_std()  
  
    // punto final bajo del intervalo de confianza del 95%  
    public double confidenceLo()  
  
    // punto final alto del intervalo de confianza del 95%  
    public double confidenceHi()  
  
    // pruebas  
    public static void main(String[] args)  
  
}
```

"Reconocer lo que no se ha hecho bien o lo que falta". Si has puntuado 3 o menos, indícame qué crees que tengo que cambiar en el diseño de la práctica para mejorar.