



**Universidad Tecnológica  
del Norte de Guanajuato**  
Organismo Público Descentralizado del Gobierno del Estado  
**“Educación y progreso para la vida”**

## TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

Licenciatura en Ingeniería en Tecnologías de la Información e Innovación Digital  
(DSM)

### ESTRUCTURA DE DATOS

#### UNIDAD II

R1

Grupo: GTID0141

Alumno: Avalos Melado Rey Gerardo

No.Control: 1224100509

Docente: Gabriel Barron Rodriguez

# Colas

## 1. Operaciones básicas de una cola

```
class Cola {  
    private int frente; i  
    private int fin; j  
    private String[] datos;
```

```
public Cola (int capacidad) {  
    datos = new String [capacidad];  
    frente = 0;  
    fin = -1;
```

```
3  
public boolean estaVacia() {  
    return frente > fin;
```

```
3  
public boolean estaLlena() {  
    return fin == datos.length - 1;
```

```
3  
public void enqueue (String elemento) {  
    if (estaLlena ()) {  
        System.out.println ("cola llena");  
    } else {  
        fin = fin + 1;  
        datos [fin] = elemento;
```

```
3  
public String dequeue () {  
    if (estaVacia ()) {  
        return null;  
    } else {  
        String temp = datos [frente];  
        frente = frente + 1;  
        return temp;
```

Simplifica estas operaciones.

Enqueue(A), Enqueue(B), Enqueue(C), Dequeue(), Enqueue(D)

Enqueue(E), Dequeue().

Inicio = frente = 0, fm = -1 (Cola vacía).

1. Enqueue(A)  $\rightarrow$  fm = 0  $\rightarrow$  [A]

2. Enqueue(B)  $\rightarrow$  fm = 1  $\rightarrow$  [A,B]

3. Enqueue(C)  $\rightarrow$  fm = 2  $\rightarrow$  [A,B,C]

4. Dequeue()  $\rightarrow$  Sacar A, frente = 1  $\rightarrow$  [B,C]

5. Enqueue(D)  $\rightarrow$  fm = 3  $\rightarrow$  [B,C,D]

6. Enqueue(E)  $\rightarrow$  fm = 4  $\rightarrow$  [B,C,D,E]

7. Dequeue()  $\rightarrow$  Sacar B, frente = 2  $\rightarrow$  [C,D,E]

Preguntas:

a) Estado final de la cola = CDE

b) Elemento en frente = C

c) Elemento en fm = E

2 - Cola circular

```
class ColaCircular {
```

```
private int frente;
```

```
private int fm;
```

```
private String [] datos;
```

```
public ColaCircular (int capacidad) {
```

```
datos = new String [capacidad];
```

```
frente = 0;
```

```
fm = 0;
```

3

```
public boolean colVacia () {
```

```
return frente == fm;
```

}

```
public boolean estllena () {
```

```
return (fm + 1) % datos.length == frente;
```

}

return temp;

3

3

Actividad:

Simula en papel con capacidad 5:

1. Dequeue(), 2. Enqueue(A), 3. Enqueue(B)
1. Dequeue()  $\rightarrow$  devuelva null (vacía), frente = 0, fn = 0.
2. Enqueue(A)  $\rightarrow$  datos [0] = A, fn = (0 + 1) / 5 = 1  $\rightarrow$  frente = 0, fn = 1.
3. Enqueue(B)  $\rightarrow$  datos [1] = B, fn = (1 + 1) / 5 = 2  $\rightarrow$  frente = 0, fn = 2.

Preguntas:

- a) Nueva posición al frente = 0
- b) Nueva posición de fn = 2
- c) Si queremos otro enqueue, se inserta en índice 2  
(datos [2]) y fn pasa a 3.

3- Verificar uso correcto de colas (término + razoamiento)

- a) Personas esperando turno en ventanilla  
Si, es FIFO: la primera persona en llegar es la primera en ser atendida
- b) Impresión de documentos: Si, los trabajos se suelen procesar por orden de llegada (FIFO).
- c) Deshacer acciones (undo): UNDO necesita LIFO
- d) Reservaciones de asientos: una reservación no siempre no siempre se maneja estrictamente FIFO.

#### 4- Simulación de cda de procesos

clase proceso {

private String nombre;

public Proceso (String n) { nombre = n; }

public String getNombre() { return nombre; }

public void setNombre (String n) { nombre = n; }

}

class ColaProcesos {

private int frente;

private int fin;

private Proceso [] procesos;

public ColaProcesos (int capacidad) {

procesos = new Proceso [capacidad];

frente = 0;

fin = -1;

}

public void enqueue (Proceso p) {

if (fin == procesos.length - 1) {

System.out.println ("cola llena");

return;

}

fin = fin + 1;

procesos [fin] = p;

}

public Proceso dequeue () {

if (frente > fin) {

return null;

}

Proceso temp = procesos [frente];

✓ frente = frente + 1;

return temp;

3

Achividad:

Dado el orden de llegada:

Proceso Tiempo

P <sub>1</sub>	0
P <sub>2</sub>	1
P <sub>3</sub>	2
P <sub>4</sub>	4

Simula en papel:

- Dibuja la cola en  $t = 0, 1, 2, 3, 4$

- Identifica que proceso es el tercero en ejecutarse

$t = 0$ : Llega P<sub>1</sub>  $\rightarrow$  Cola: [P<sub>1</sub>]  $\rightarrow$  CPU ejecuta P<sub>1</sub> (se quita) Ejecutando: P<sub>1</sub>

$t = 1$ : Llega P<sub>2</sub>  $\rightarrow$  Cola: [P<sub>2</sub>]  $\rightarrow$  CPU ejecuta P<sub>2</sub>, Ejecutando: P<sub>2</sub>

$t = 2$ : Llega P<sub>3</sub>  $\rightarrow$  Cola: [P<sub>3</sub>]  $\rightarrow$  CPU ejecuta P<sub>3</sub>, Ejecutando: P<sub>3</sub>

$t = 3$ : No llegan nuevos  $\rightarrow$  Cola vacía  $\rightarrow$  CPU idle

$t = 4$ : Llega P<sub>4</sub>  $\rightarrow$  Cola [P<sub>4</sub>]  $\rightarrow$  CPU ejecuta P<sub>4</sub> ejecutando P<sub>4</sub>

Identificando que proceso es el tercero en ejecutarse: P<sub>3</sub>

5) BFS usando una cola

```
class ColaInt {  
    private int[] datos;  
    private int frente, fin;  
    public ColaInt (int capacidad) {  
        datos = new int[capacidad];  
        frente = 0;  
        fin = -1;  
    }
```

3  

```
public void enqueue (int x) {  
    fin = fin + 1;  
    datos [fin] = x;
```

3  

```
public int dequeue () {  
    int temp = datos [frente];  
    frente = frente + 1;  
    return temp;
```

public boolean colision(C) {  
return frente > fin;

}

3

Atributos: BFS;

Grado:

A-B-C-D

19. Dados números: A=0, B=1, C=2, D=3.

1- Simular BFS, mirando en A

2- Dibujar el estadio de la cola en cada iteración

3- Escibir el orden de visita: 0, 1, 2, 3

Iniciar: encolar A  $\rightarrow$  cola: [0]

Proceso 0: visitados {0}; encolar vecinos de 0  $\rightarrow$  encolar 1 [cola: {1}]

Proceso 1: visitados {0,1}; encolar vecinos no visitados  $\rightarrow$  encolar 2 [cola: {2}]

Proceso 2: visitados {0,1,2}; encolar 3  $\rightarrow$  cola [3]

Proceso 3: visitados {0,1,2,3}; cola vacía  $\rightarrow$  fin

6- Cola en un banco

Completa manualmente el simulador en papel (Tab)

class BancoCola {

private int frente, fin;

private String[] clientes = new String[10];

public BancoCola() {

frente = 0;

fin = -1;

3

public void encolar (String c) {

fin = fin + 1;

clientes [fin] = c;

3

```
public String dequeue() {  
    String temp = clients[front];  
    front = front + 1;  
    return temp;
```

{

}

Síntaxis:

- |                    |  |
|--------------------|--|
| 1. Enqueue (Carla) | 1. Enqueue (Carla) $\rightarrow f_{in} = 0$ : [Carla]                    |
| 2. Dequeue ()      | 2. Dequeue () $\rightarrow$ saca Carla, frente = 1 $\rightarrow []$      |
| 3. Enqueue (Pedro) | 3. Enqueue (Pedro) $\rightarrow f_{in} = 1$ [Pedro]                      |
| 4. Enqueue (Sofia) | 4. Enqueue (Sofia) $\rightarrow f_{in} = 2$ [Pedro, Sofia]               |
| 5. 2x Dequeue ()   | 5. Dequeue () $\rightarrow$ saca Pedro, frente = 2 $\rightarrow$ [Sofia] |
|                    | 6. Dequeue () $\rightarrow$ saca Sofia, frente = 3 $\rightarrow []$      |

Preguntas:

- Cola final: vacía
- Quién queda al frente: Nadie
- Total personas: 0