Problema de Josephus

Taller de Algoritmos y Estructuras de Datos II

UNIVERSIDAD SIGLO 21

Problema de Josephus

Contexto

En una simulación, la computadora emula el funcionamiento de un sistema real y toma datos estadísticos.

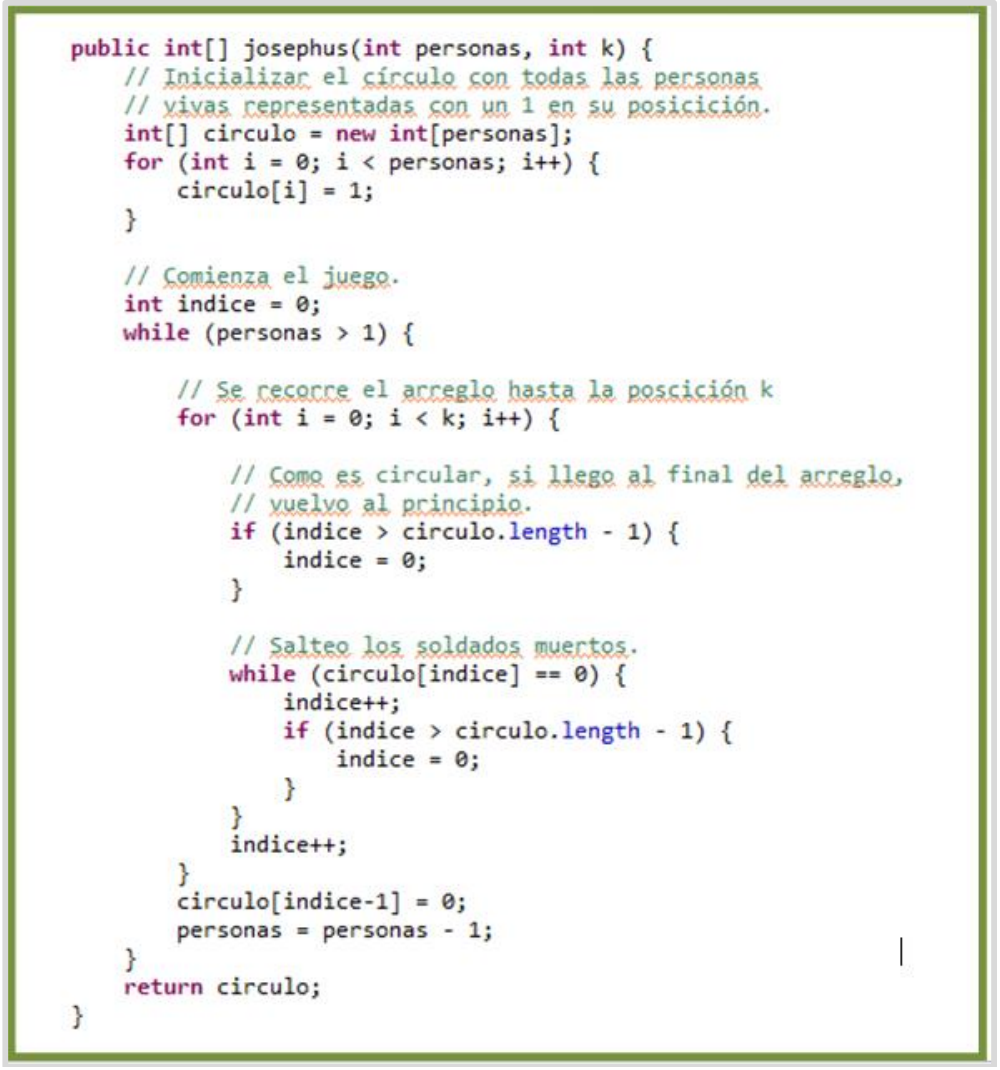
El problema de Josephus es un problema teórico cuyo nombre se debe a Flavio Josefo, un historiador que vivió en el siglo I. Según lo que cuenta Josefo, él y 40 soldados se encontraban atrapados en una cueva, rodeados de romanos, y prefirieron suicidarse antes que ser capturados. Para ello, decidieron que la suerte debía decir quién mataba a quién. Los últimos que quedaron fueron él y otro hombre. Entonces, convenció al otro hombre de que debían entregarse a los romanos en lugar de matarse entre ellos. Josefo atribuyó su supervivencia a la suerte o al destino.

El problema plantea lo siguiente:

> Hay n personas paradas en un círculo esperando a ser ejecutadas. Después de que ejecutan a la primera persona, saltean a k-1 personas y la persona número k es ejecutada. Entonces nuevamente, saltean a k-1 personas y la persona número k es ejecutada. La eliminación continúa alrededor del círculo (que se hace cada vez más pequeño a medida que más personas son eliminadas del mismo) hasta que sólo queda la última, que es liberada. ("Problema de Flavio Josefo", s. f., http://es.wikipedia.org/wiki/Problema_de_Flavio_Josefo).

El objetivo es escoger el lugar inicial en el círculo para sobrevivir (es el último que queda), dados *n* y *k*. El siguiente algoritmo en Java resuelve el problema de Josephus:

Figura 1:



```
public int[] josephus(int personas, int k) {
    // Inicializar el círculo con todas las personas
    // vivas representadas con un 1 en su posicición.
    int[] circulo = new int[personas];
    for (int i = 0; i < personas; i++) {
        circulo[i] = 1;
    }

    // Comienza el juego.
    int indice = 0;
    while (personas > 1) {

        // Se recorre el arreglo hasta la poscición k
        for (int i = 0; i < k; i++) {

            // Como es circular, si llego al final del arreglo,
            // vuelvo al principio.
            if (indice > circulo.length - 1) {
                indice = 0;
            }

            // Salteo los soldados muertos.
            while (circulo[indice] == 0) {
                indice++;
                if (indice > circulo.length - 1) {
                    indice = 0;
                }
            }
            indice++;
        }
        circulo[indice-1] = 0;
        personas = personas - 1;
    }
    return circulo;
}
```

El método Josephus recibe como parámetros la cantidad de personas y la constante *k*, que indica la cantidad posiciones que hay que dejar hasta eliminar a una persona.

En la primera parte del algoritmo, se crea un arreglo con la cantidad de personas y se inicializa cada posición del arreglo (posición en la que se encuentra una persona) con un 1 que indica que la persona está viva.

Luego se inicializa el índice en 0 para indicar el comienzo del juego. Se itera hasta que quede solo una persona viva (persona > 1). Por cada eliminación, se itera desde 0 hasta la constante *k* para saber quién es el próximo que se debe eliminar. En este paso, hay que tener en cuenta que a las posiciones con personas eliminadas (circulo[índice] == 0) hay que saltearlas, ya que no las tenemos que contar. Luego de contar hasta *k*, marcamos la posición con un 0 indicando que esa persona se elimina y restamos la cantidad de

personas en una unidad para seguir con el juego. En cada incremento del índice, tenemos que verificar que este no supere el tamaño del arreglo; si esto ocurre, debemos reiniciar el índice a 0, ya que consideramos que las personas están ubicadas en círculo y que, después de pasar por la última persona, la cuenta sigue con el primero (arreglo circular). También es posible implementarlo utilizando una lista doble vinculada circular.

Finalmente, el algoritmo retorna el arreglo con la única persona que sobrevivió. Con esto podemos saber que, si hay 20 personas y la constante *k* es 45, la posición en la que deberíamos colocarnos para sobrevivir es en la 19.

Figura 2:

```
public static void main(String[] args) {
    Main main = new Main();

    int[] circulo = main.josephus(20, 45);
    for (int i = 0; i < circulo.length; i++) {
        System.out.print(i + 1 + "\t");
        System.out.println(circulo[i]);
    }
}
```

```
Console
<terminated> Main (6) [Java Application]
1       0
2       0
3       0
4       0
5       0
6       0
7       0
8       0
9       0
10      0
11      0
12      0
13      0
14      0
15      0
16      0
17      0
18      0
19      1
20      0
```

Referencias

Weiss, M. A. (2000). *Estructuras de Datos en Java*. Buenos Aires, AR: Adisson Weisley.

Cormen, T., Leiserson, C., Rivest, R. y Stein, C. (2002). *Introduction to Algorithms* (2nd ed.). McGraw Hill.

Sznajdleder, P. A. (2012). *Algoritmos a fondo con implementación en C y JAVA*. Buenos Aires: Alfaomega.