INFORME COMPARATIVO ENTRE EL MÉTODO ITERATIVO Y RECURSIVO

KATHERIN BARRERA

DAVID CASALLAS

VALERIA FLOREZ

UNIVERSIDAD DEL NORTE

ALGORITMIA Y PROGRAMACIÓN II_202430_2135

SEGUNDO SEMESTRE

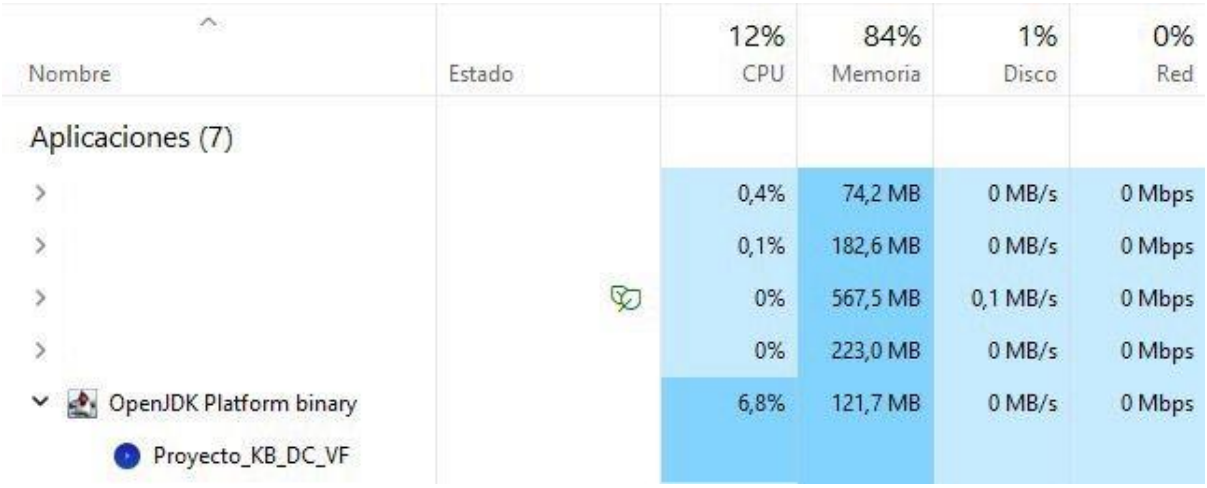
BARRANQUILLA

2024

En el siguiente informe se mostrará algunas comparaciones entre dos métodos (iterativo y recursivo), donde se evidenciarán diferencias entre la carga computacional, la redundancia, la complejidad de la solución y por último la elegancia de código.

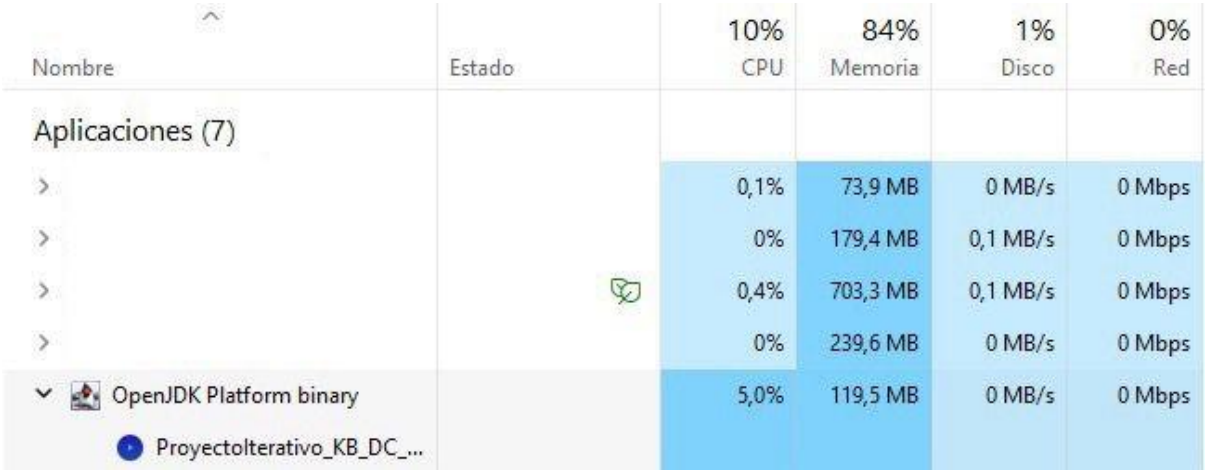
CARGA COMPUTACIONAL:

En esta parte se comparan ambos códigos con el administrador de tareas de uno de nuestros computadores, ya que con esta opción podemos visualizar en el computador cuanta memoria y CPU consumen los códigos.



Nombre	Estado	12% CPU	84% Memoria	1% Disco	0% Red
Aplicaciones (7)					
>		0,4%	74,2 MB	0 MB/s	0 Mbps
>		0,1%	182,6 MB	0 MB/s	0 Mbps
>		0%	567,5 MB	0,1 MB/s	0 Mbps
>		0%	223,0 MB	0 MB/s	0 Mbps
OpenJDK Platform binary		6,8%	121,7 MB	0 MB/s	0 Mbps
Proyecto_KB_DC_VF					

En esta imagen estudiamos la carga de CPU y la memoria que usa el **método recursivo** a la hora de generar la matriz y estallar una flor de loto.



Nombre	Estado	10% CPU	84% Memoria	1% Disco	0% Red
Aplicaciones (7)					
>		0,1%	73,9 MB	0 MB/s	0 Mbps
>		0%	179,4 MB	0,1 MB/s	0 Mbps
>		0,4%	703,3 MB	0,1 MB/s	0 Mbps
>		0%	239,6 MB	0 MB/s	0 Mbps
OpenJDK Platform binary		5,0%	119,5 MB	0 MB/s	0 Mbps
ProyectoIterativo_KB_DC_...					

Con el **método iterativo** se estudió el proceso de la misma manera que con el método recursivo y estos fueron los resultados.

Al ver estas dos imágenes podemos observar como el método recursivo consume bastante memoria, pero no tanta CPU, caso diferente con el iterativo que ocurre lo contrario. Esto es debido a que los métodos recursivos usan “pilas de llamadas”, esto quiere decir que dejan espacios reservados en la memoria para q cuando llegue el caso base llene los espacios reservados, cosa que el iterativo no realiza ya que este método no deja reservado si no que mientras guarda espacio en la memoria va resolviendo el problema.

REDUNDANCIA:

En esta parte podemos evidenciar cuantas veces se repiten los ciclos (en el caso iterativo), o se llama el método así mismo (en el caso recursivo), donde comparamos ambos códigos y podemos evidenciar algo particular.

```java
void explotarBurbujaRecursivo(int x, int y, int valorInicial) {
  if (x < 0 || x >= M || y < 0 || y >= N || matriz[x][y] != valorInicial) return;

  matriz[x][y]--;

  explotarBurbujaRecursivo(x - 1, y, valorInicial); // Arriba
  explotarBurbujaRecursivo(x + 1, y, valorInicial); // Abajo
  explotarBurbujaRecursivo(x, y - 1, valorInicial); // Izquierda
  explotarBurbujaRecursivo(x, y + 1, valorInicial); // Derecha
}
```

(imagen método recursivo)

```java
void explotarBurbujaIterativo(int x, int y) {
  int[][] cola = new int[M * N][2];
  int inicio = 0;
  int fin = 0;

  cola[fin][0] = x;
  cola[fin][1] = y;
  fin++;

  int valorInicial = matriz[x][y];
  matriz[x][y]--;

  while (inicio < fin) {
    int cx = cola[inicio][0];
    int cy = cola[inicio][1];
    inicio++;

    int[][] direcciones = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
    for (int[] dir : direcciones) {
      int nx = cx + dir[0];
      int ny = cy + dir[1];
      if (nx >= 0 && nx < M && ny >= 0 && ny < N && matriz[nx][ny] == valorInicial) {
        matriz[nx][ny]--;
        cola[fin][0] = nx;
        cola[fin][1] = ny;
        fin++;
      }
    }
  }
}
```

(imagen método iterativo)

Viendo las imágenes se aprecia que el método iterativo funciona gracias a muchos condicionales, lo que en la recursiva no se ve, ya que acá solo cambiamos los parámetros para que así se vean las posiciones alrededor de la seleccionada.

En este caso ambos tienen redundancia ya que no contamos con una herramienta de optimización para que el método no recorra partes que ya había recorrido posteriormente. En

el iterativo podemos ver que hay redundancia, ya que cada vez que vuelve a ingresar al ciclo (en este caso un “while”), repite el mismo procedimiento una y otra vez, recorriendo espacios que ya recorrió anteriormente. Lo mismo pasa con recursivo, ya que a pesar de que se ve más simple, con cada vez que se vuelve a llamar vuelve a recorrer espacios que ya recorrió en llamamientos anteriores. Lo anterior nos lleva a que la redundancia del método iterativo es más extensa o pesada que el recursivo debido a las condiciones y procedimientos que presenta.

COMPLEJIDAD DEL CÓDIGO:

En la complejidad de los métodos, el método recursivo es mucho más rápido de ver y hacer ya que los pasos son muchos más sencillos y eficientes. A la hora de realizarlo solo con el cambio de los parámetros “x” y “y”, ya sea sumándolos o restándolos, podíamos cambiar de posición fácilmente, en cambio en el método iterativo es necesario el uso de condicionales e incluso crear otra matriz que nos ayude de auxiliar para poder analizar las posiciones cercanas a las que usa el usuario. Al programar es mucho más complicado, ya que al hacer un bosquejo o con el simple hecho de tener la idea, parece fácil de realizar, pero cuando se va a programar se complica algo ya que los requiere de parámetros difíciles de descifrar.

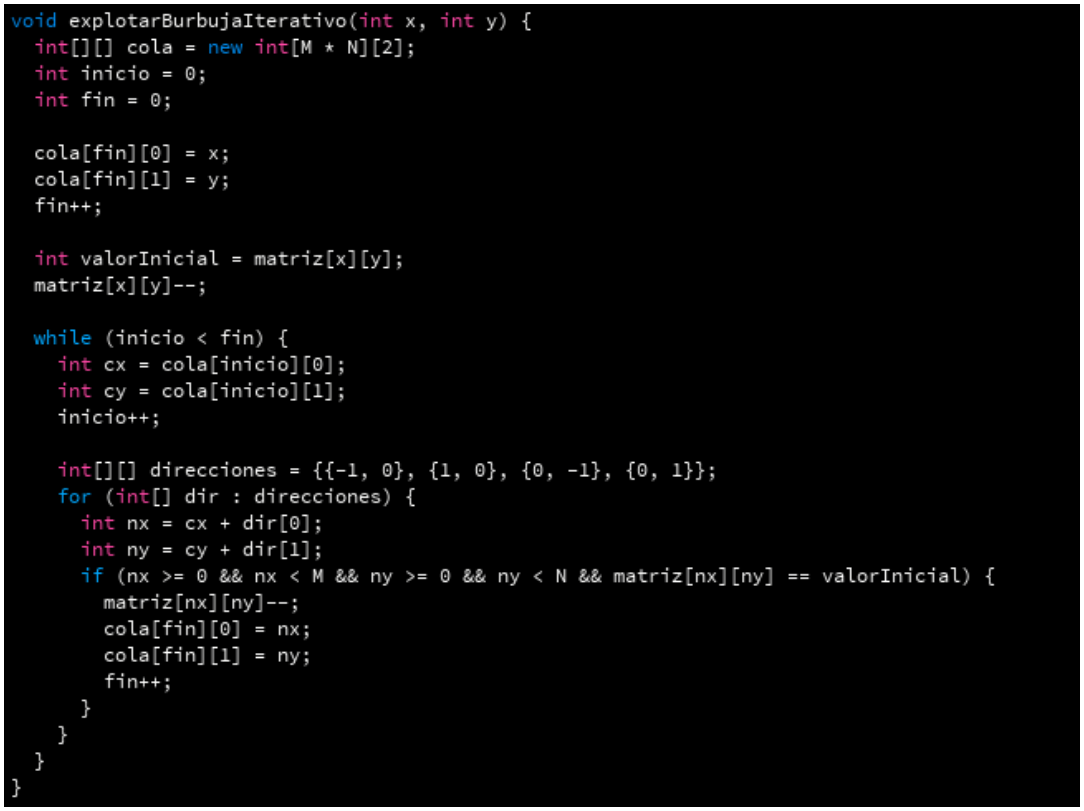
LEGIBILIDAD Y ELEGANCIA DEL CÓDIGO:

```
void explotarBurbujaRecursivo(int x, int y, int valorInicial) {
  if (x < 0 || x >= M || y < 0 || y >= N || matriz[x][y] != valorInicial) return;

  matriz[x][y]--;

  explotarBurbujaRecursivo(x - 1, y, valorInicial); // Arriba
  explotarBurbujaRecursivo(x + 1, y, valorInicial); // Abajo
  explotarBurbujaRecursivo(x, y - 1, valorInicial); // Izquierda
  explotarBurbujaRecursivo(x, y + 1, valorInicial); // Derecha
}
```

(imagen método recursivo)



```
void explotarBurbujaIterativo(int x, int y) {
  int[][] cola = new int[M * N][2];
  int inicio = 0;
  int fin = 0;

  cola[fin][0] = x;
  cola[fin][1] = y;
  fin++;

  int valorInicial = matriz[x][y];
  matriz[x][y]--;

  while (inicio < fin) {
    int cx = cola[inicio][0];
    int cy = cola[inicio][1];
    inicio++;

    int[][] direcciones = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
    for (int[] dir : direcciones) {
      int nx = cx + dir[0];
      int ny = cy + dir[1];
      if (nx >= 0 && nx < M && ny >= 0 && ny < N && matriz[nx][ny] == valorInicial) {
        matriz[nx][ny]--;
        cola[fin][0] = nx;
        cola[fin][1] = ny;
        fin++;
      }
    }
  }
}
```

(imagen método iterativo)

Con solo ver las imágenes es más fácil de entender el método recursivo, además es más agradable a la vista porque son pocas líneas de código, en cambio el iterativo es mucho más largo y para una persona familiarizada con la programación no le resulta agradable porque no parece elegante ni es eficiente por lo complejo y extenso que es.

En conclusión, ambos métodos tienen sus pros y contras ilustrados en la siguiente tabla para entenderlo mejor:

METODO ITERATIVO	METODO RECURSIVO
- Es menos eficiente a la hora de mostrar la información. - No usa tanta memoria a la hora de ejecutar el programa. - Su legibilidad no es tan buena debido a lo extenso que es y no es tan agradable visualmente. - Hace comparaciones hechas anteriormente.	- Es más eficiente a la hora de mostrar la información. - Usa más memoria debido a que deja reservas en la memoria. - Su legibilidad es muy buena ya que tiene pocas líneas de código y es agradable visualmente. - Hace comparaciones ya realizadas anteriormente.