

Matemáticas Discretas

Proyecto 2 (Recursividad)

Prof: Dr. Fabián Riquelme Csori <fabian.riquelme@uv.cl>
Ayudante: Ricardo Díaz González <ricardo.diazgo@alumnos.uv.cl>
2023-II

1. Objetivos

- Aplicar conceptos teóricos de recursividad.
- Implementar algoritmos recursivos considerando el paradigma de Programación Orientada a Objetos (POO).
- Desarrollar competencias de trabajo en equipo.

2. Descripción

- El proyecto se realiza en **grupos de 3-4 integrantes**, organizados por preferencia.
- Se debe entregar un Google Colab con la solución ejecutable, sin errores, acompañada de código bien documentado y ajustado a los requerimientos (ver sección 2.5).
- No se debe entregar informe ni presentación.
- Eventualmente, el profesor podría pedir que un grupo le explique la entrega en reunión, de forma de validar el trabajo realizado y la distribución de tareas.

2.1. Algoritmos recursivos

En ciencias de la computación, el principio de recursividad o recursión se puede entender como una función que se llama a sí misma. La recursividad es una forma poderosa, elegante y natural de resolver una amplia gama de problemas, a través de la estrategia “divide y vencerás”, en que la solución global depende de soluciones a instancias más pequeñas del mismo problema. Cada subproblema se descompone hasta el punto en que los subproblemas se pueden resolver de forma directa. Finalmente, las soluciones de los subproblemas se vuelven a combinar para obtener la solución del problema original. Dicho de otra forma, en toda función recursiva debe haber algún caso en que la función deje de llamarse a sí misma (caso base); de lo contrario, la recursión se invocaría a sí misma sin detenerse.

Las matemáticas detrás de las definiciones recursivas son fácilmente comprensibles, y los algoritmos para implementarlas resultan comprensibles y relativamente fáciles de definir. Sin embargo, desde el punto de vista de la ejecución del programa, estos pueden ser ineficientes,

debido a la gran cantidad de llamadas recursivas que pueden ocurrir. En palabras simples, el algoritmo recursivo puede terminar ejecutando más instrucciones que su versión iterativa. Además, e incluso abusando un poco del vocabulario (estos conceptos los verán en profundidad en el curso de Sistemas Operativos), por cada llamada recursiva se debe crear espacio de memoria para almacenar las variables locales de la función, las que se liberarán una vez que se alcance el “caso base”.

En este trabajo visualizaremos esta complejidad de los algoritmos recursivos en el diseño y el proceso de anidar y desanidar muñecas Matrioshka.

2.2. Muñecas Matrioshka

Las muñecas Matrioshka (ver Figura 1), también conocidas como muñecas rusas, son un tipo de muñecas de madera tradicionales de Rusia y Ucrania que han fascinado a muchas personas alrededor del mundo. Estas llevan consigo un significado cultural y simbólico profundo. Se consideran un símbolo de maternidad, fertilidad y protección familiar, reflejando así la unidad familiar y la conexión entre generaciones.

Las muñecas se caracterizan por su diseño distintivo, que consiste en una serie de muñecas más pequeñas contenidas dentro de una muñeca más grande. La más grande encaja en su interior una serie de muñecas más pequeñas, que a su vez pueden contener otras aún más pequeñas, formando una serie de “capas” o “anidamientos” que pueden variar en número. Este diseño anidado es un claro ejemplo de la aplicación de la recursividad en el ámbito de la artesanía.

El proceso de *anidar* las muñecas Matrioshka implica colocar una muñeca dentro de otra de manera cuidadosa y precisa, de forma que cada una se ajusta perfectamente dentro de la siguiente más grande. Así, finalmente se tiene a las muñecas completamente contenidas dentro de la más grande. Por el contrario, el proceso de *desanidar* las muñecas consiste en quitar una muñeca a la vez para revelar la siguiente más pequeña en su interior, y esto se repite hasta llegar a la más pequeña.



Figura 1: Set de $N = 7$ muñecas Matrioshka.

2.3. Trabajo a realizar

El trabajo a desarrollar consiste en considerar el paradigma de Programación Orientada a Objetos (POO) para implementar y desarrollar algoritmos que simulan el proceso de *anidar* y *desanidar* las muñecas Matrioshka. Para esto, se debe tener en cuenta los siguientes aspectos:

- Cada muñeca tiene un tamaño y puede contener otra de menor tamaño en su interior (por tanto, esta última puede tener a su vez otras muñecas en su interior).
- La cantidad de muñecas estará dada por un número entero positivo N ingresado por el usuario como *input* único (este valor influye en el tamaño de las muñecas, explicado en el siguiente punto).
- El tamaño de cada muñeca se encuentra dentro del rango $[1, N]$, siendo N el *input* ingresado por el usuario. La muñeca más pequeña será de tamaño 1 y la más grande será de tamaño N . Cada muñeca posee un tamaño único, distinto al de las demás. Por ejemplo, para $N = 3$ (es decir, tres muñecas), existirá exactamente una muñeca de tamaño 1, una de tamaño 2, y otra de tamaño 3.
- A nivel de implementación, una vez ingresado el *input* N del usuario, las muñecas deben ser almacenadas *aleatoriamente* dentro de un *array*. Este arreglo será la estructura de datos con la que deberán funcionar los algoritmos de *anidación* y *desanidación* (procurar hacer una copia del *array* creado, ya que en ambos algoritmos se debe trabajar con la misma estructura).
- Para determinar la forma correcta en que se anidan unas dentro de otras, una muñeca de tamaño i puede colocarse solamente dentro de una de tamaño $i + 1$. Por ejemplo, si se quiere colocar una muñeca de tamaño 2 dentro de una de tamaño 4, antes debe ponerse directamente en el interior de la muñeca de tamaño 3, y luego esta última dentro de la de tamaño 4.

Dicho lo anterior, se pide diseñar e implementar en lenguaje de programación Python 3.x lo siguiente:

- R1. Un algoritmo **recursivo** que simule el proceso de *anidar* las muñecas Matrioshka. Despliegue en pantalla el orden de las muñecas en el *array* (utilizando su tamaño) en cada instancia y el tiempo de ejecución requerido para anidarlas.
- R2. Un algoritmo **iterativo** que simule el proceso de *anidar* las muñecas Matrioshka. Despliegue en pantalla el orden de las muñecas en el *array* (utilizando su tamaño) en cada instancia y el tiempo de ejecución requerido para anidarlas.
- R3. Un algoritmo **recursivo** que simule el proceso de *desanidar* muñecas Matrioshka, luego de anidarlas. Este debe recibir como parámetro la muñeca de mayor tamaño que contiene a las demás. Despliegue en pantalla la extracción de cada muñeca (utilizando su tamaño).

Además, debe responder, en el mismo Colab:

- P1. ¿Cuál de los dos algoritmos implementados es más eficiente en tiempo de ejecución? ¿Por qué? Justifique su respuesta considerando los resultados obtenidos en los requerimientos. Opcionalmente, puede complementar su respuesta utilizando recursos externos (páginas web, libros, etc.), explicitando dichas referencias.

2.4. *Input - Outputs*

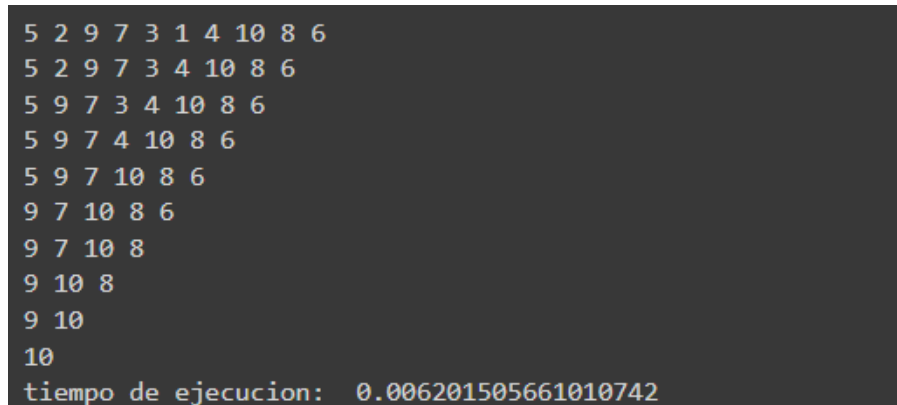
A continuación se muestran ejemplos de *input* único y *outputs* esperados:

- Ejemplo de *input* único:



Figura 2: Ejemplo de *input* único

- Ejemplo de *output* para R1 y R2:



```
5 2 9 7 3 1 4 10 8 6
5 2 9 7 3 4 10 8 6
5 9 7 3 4 10 8 6
5 9 7 4 10 8 6
5 9 7 10 8 6
9 7 10 8 6
9 7 10 8
9 10 8
9 10
10
tiempo de ejecucion: 0.006201505661010742
```

Figura 3: Ejemplo de *output* de R1 y R2, para $N = 10$

Tener en cuenta que el tiempo de ejecución puede variar en cada máquina.

- Ejemplo de *output* para R3:

```
10
9
8
7
6
5
4
3
2
1
```

Figura 4: Ejemplo de *output* de R3, para $N = 10$

Todos los integrantes del grupo deben comprender y poder explicar en su totalidad el código, por lo que deben repartirse trabajo y comprender tanto la teoría como la práctica. **La copia entre equipos de trabajo está estrictamente prohibida y será amonestada con nota 1.0 para cada grupo que la realice.**

2.5. Entrega

Su proyecto debe ser creado en un Google Colab, a partir de la duplicación del código en este enlace. Al inicio debe indicar los nombres completos de los integrantes del grupo. El proyecto debe ser descargado desde Google Colab en formato `.ipynb` y subido al Aula Virtual, en el recurso respectivo, con el siguiente formato:

Proyecto2-PrimerApellido1-PrimerApellido2-PrimerApellido3.ipynb

Ejemplo de formato: Proyecto2-Bustamante-Diaz-Elgueta-Ponce.ipynb

Además, junto con la entrega del proyecto, cada estudiante debe responder esta encuesta, indicando el factor de participación (FP) para cada compañero/a de grupo (ver Sección 3.2).

Fecha de Entrega: Martes 7 de noviembre de 2023, 13:30 hrs.

3. Sistema de evaluación

Este proyecto vale un 10 % de la nota final del curso. La nota del Proyecto (NP) estará dada por una nota grupal (NG) resultante del algoritmo implementado, y un factor de participación (FP) entregado por sus pares de forma anónima:

$$NP = NG \cdot FP$$

3.1. Nota grupal (NG)

Criterio	Puntos
Calidad del código. El código implementado se ejecuta sin errores; es claro, está bien documentado y se ajusta con las condiciones de entrega.	20 pts
Correctitud y completitud	
Correctitud y completitud del requerimiento R1	30 pts
Correctitud y completitud del requerimiento R2	20 pts
Correctitud y completitud del requerimiento R3	20 pts
Respuesta y justificación	
Respuesta y justificación de la pregunta P1	10 pts
Total	100 pts

3.2. Factor de participación (FP)

Nivel de trabajo	Factor
Excelente. Mi compañero/a fue indispensable para realizar del trabajo. Sin él/ella no se podría haber realizado (solo puede evaluar a máximo 1 integrante de esta manera).	1.1
Bueno. Mi compañero/a se desempeñó de manera satisfactoria y responsable en el equipo.	1.0
Insuficiente. Mi compañero/a no participó lo suficientemente en el trabajo. Su desempeño no fue el ideal esperado.	0.8
Deficiente. Mi compañero/a no fue un aporte al equipo. El equipo habría funcionado mejor sin él/ella.	0.6

La nota FP para un/a estudiante corresponde al promedio de los factores ingresados por sus compañeros/as. Eventualmente, también pueden incidir las respuestas a preguntas realizadas por el profesor, en caso de que el grupo se entrevistó con éste.