ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2024-10. Proyecto – PARTE 3
Entrega: jueves, mayo 30 de 2024 6:00PM (por Bloque Neon)

0 OBJETIVOS

- Diseñar soluciones computacionales para problemas.
- Estimar costos de las soluciones planteadas.
- Implementar soluciones.

Se premiarán las mejores soluciones y se castigarán las peores, en cuanto a optimización, eficiencia en tiempo y espacio.

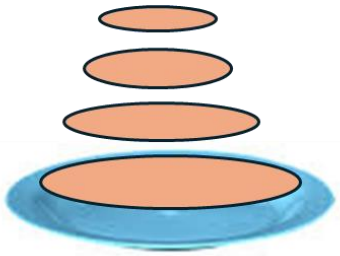
1 CONDICIONES GENERALES

El proyecto se divide en tres partes independientes entre sí. Este documento describe la PARTE III. Cada parte contiene un problema a resolver mediante soluciones implementadas en `Java` o `Python`.

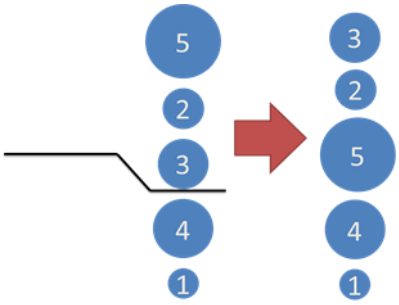
Para cada problema se pide:
- Descripción de la solución.
- Análisis temporal y espacial.
- Una implementación en Java o Python

2 DESCRIPCIÓN DEL PROBLEMA

Un mesero y un cocinero de un restaurante de desayunos se llevan terriblemente mal. Por políticas del restaurante los panqueques que se sirven a los clientes se deben servir como una pila ordenada:



El cocinero busca frecuentemente la forma de irritar al mesero y por eso le entrega los panqueques desordenados. El mesero debe entonces ordenarlos con la espátula usando “flips” para evitar tocar los panqueques y platos extras. En la operación de flip el mesero inserta la espátula en una posición y gira todos los panqueques. En la siguiente figura tenemos un ejemplo de “flip” para un stack de 5 panqueques ($n = 5$).



Problema

El mesero que conoce y le pide a un estudiante de DALGO que diseñe un algoritmo que le permita encontrar para una pila de tamaño n particular la secuencia de flips **más pequeña** (**mínima**) que se necesita para ordenar los panqueques. Se evaluará que tan cercana (o mejor) es su solución a la solución de referencia.

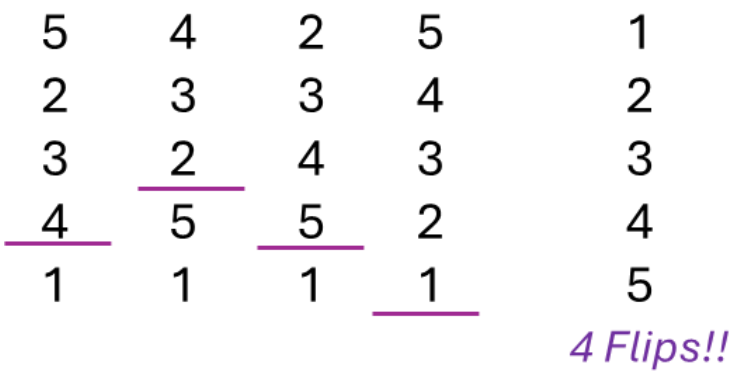
Ejemplo 1

Vamos a simular los órdenes de tamaño en una pila de panqueques usando números naturales no repetidos. Para una pila $n = 5$, 5 representa el pancake de mayor tamaño y 1 representa el más pequeño. Suponga por ejemplo la pila de entrada (1,4,3,2,5). Una forma de ordenarla es con la siguiente secuencia de flips:

5	1	4	2	3	1
2	4	1	3	2	2
3	3	3	1	1	3
4	2	2	4	4	4
1	5	5	5	5	5

5 Flips!!

Sin embargo, ¡esta solución NO ES EL MINIMO número de flips!. Para esta pila, el mínimo son 4 flips:



Por lo tanto, una posible solución sería la secuencia de flips en las posiciones (1,2,1,0):

3 ENTRADA Y SALIDA DE DATOS

En todas las soluciones que se presenten, la lectura de los datos de entrada se hace por la entrada estándar; así mismo, la escritura de los resultados se hace por la salida estándar.

Puede suponer que ninguna línea de entrada tiene espacios al principio o al final, y que los datos que se listan en cada línea están separados por exactamente un espacio.

A continuación, se establecen parámetros que definen su tamaño y formato de lectura de los datos, tanto de entrada como de salida.

Descripción de la entrada

- *Entrada*: La primera línea de entrada especifica el número de casos de prueba (i.e. pilas de panqueques) que contiene el archivo. El programa debe terminar su ejecución una vez termine de resolver la cantidad de casos de prueba dados por este número. Cada caso es representado por una línea con n números separados por espacio. Esta línea representa la pila de panqueques empezando desde la base hasta el panqueue en el tope de la pila $(5 \leq n \leq 1000)$.
- *Salida*: La posición de los flips realizado en orden de ejecución y separados por espacio. Si no es necesario ningún movimiento escriba la palabra “ORDENADO”.

Ejemplo de entrada / salida

Entrada	Salida
3 1 4 3 2 5 1 2 3 4 5 5 4 3 2 1	1 2 1 0 0 ORDENADO

Nota: Se van a diseñar casos de prueba para valores de n mucho más grandes y dentro de los valores establecidos en el enunciado. Los casos mostrados en este documento son demostrativos de la estructura de entrada/salida esperada.

4 COMPRENSIÓN DE PROBLEMAS ALGORITMICOS

A continuación, se presentan un conjunto de escenarios hipotéticos que cambian el problema original. Para cada escenario debe contestar[1]: (i) que nuevos retos presupone este nuevo escenario -si aplica-?, y (ii) que cambios -si aplica- le tendría que realizar a su solución para que se adapte a este nuevo escenario?

ESCENARIO 1: Suponga que existe un segundo movimiento "switch" que se puede ejecutar una única vez por pila. Este movimiento intercambia dos posiciones de la pila.

ESCENARIO 2: El cocinero ahora es el que tiene como amigo el estudiante de DALGO y le pide que le diseñe la pila de panqueques más complicada de ordenar.

Nota: Los escenarios son independientes entre sí.

5 ENTREGABLES

El proyecto puede desarrollarse por grupos de hasta dos estudiantes de la misma sección. La entrega se hace por bloque neon (una sola entrega por grupo de trabajo).

El grupo debe entregar, por bloque neon, un archivo de nombre `proyectoDalgoP3.zip`. Este archivo es una carpeta de nombre `proyectoDalgoP3`, comprimida en formato `.zip`, dentro de la cual hay archivos fuente de soluciones propuestas y archivos que documentan cada una de las soluciones.

5.1 Archivos fuente de soluciones propuestas

Todos los programas deben ser implementados en *Java* o en Python

Para el problema:

- Entregar un archivo de código fuente en *Java* (`.java`) o python (`.py`) con su código fuente de la solución que se presenta.
- Incluir como encabezado de cada archivo fuente un comentario que identifique el (los) autor(es) de la solución.
- Denominar `ProblemaP3.java` o `ProblemaP3.py` el archivo de la solución que se presente.

Nótese que, si bien puede utilizarse un *IDE* como *Eclipse* o Spyder durante el desarrollo del proyecto, la entrega requiere incluir solo un archivo por cada solución. El archivo debe poderse compilar y ejecutar independientemente (sin depender de ninguna estructura de directorios, librerías no estándar, etc.).

[1] NO tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

5.2 Archivos que documentan la solución propuesta

La solución al problema debe acompañarse de un archivo de máximo 3 páginas que la documente, con extensión `.pdf`. El nombre del archivo debe ser el mismo del código correspondiente (`ProblemaP3.pdf`).

Un archivo de documentación debe contener los siguientes elementos:

0 Identificación
Nombre de autor(es)
Identificación de autor(es)

1 Algoritmo de solución
Explicación del algoritmo elegido. Si hubo alternativas de implantación diferentes, explicar por qué se escogió la que se implementó. Generar al menos una gráfica que apoye la explicación del algoritmo implementado. No se debe copiar y pegar código fuente como parte de la explicación del algoritmo. Argumentar si el algoritmo planteado resuelve perfectamente el problema.

2 Análisis de complejidades espacial y temporal
Cálculo de complejidades y explicación de estas.

3 Respuestas a los escenarios de comprensión de problemas algorítmicos.
Respuesta a las preguntas establecidas en cada escenario. No tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

La nota del informe corresponde a un 50% de la nota total de la entrega del proyecto, solamente si se entrega el código fuente de la solución implementada. En caso de no entregar el código fuente, no se hará evaluación del informe y la nota del proyecto será cero.

Además de la pertinencia del texto como explicación de la solución implementada, se evaluará la calidad en la redacción del texto y en el diseño de las gráficas. Se evaluará también que la explicación del algoritmo integre los conceptos relacionados con las técnicas de diseño de algoritmos cubiertas en el curso.

Téngase en cuenta que los análisis de 2 tienen sentido en la medida que la explicación de 1 sea clara y correcta. No se está exigiendo formalismo a ultranza, pero sí que, como aplicación de lo estudiado en el curso, se pueda describir un algoritmo de manera correcta y comprensible.