ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2024-10. Proyecto – PARTE 1
Entrega: Lunes, 11 de marzo 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 eficiencia en tiempo y espacio.

1 CONDICIONES GENERALES

El proyecto se divide en tres partes independientes entre sí. Este documento describe la PARTE I. 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

A Torres de lego

Un erudito amante de los legos frecuentemente se le ve construyendo torres con fichas. En un espacio lineal construye n torres de fichas contiguas, donde la torre $i-esima$ está compuesta por f_i fichas de igual tamaño apiladas una sobre la otra (ver Fig 1). Con el animo de entrenar a su estudiante, el erudito le pide que reorganice las torres de manera descendiente de tal manera que para todo i se cumpla que $f_i \geq f_{i+1}$. El erudito le pone como restricción al estudiante que solo realice movimientos de una (1) ficha entre las torres adyacentes más cercanas.

Problema

Dada una disposición inicial de torres y fichas que cumplen el enunciado, calcular el mínimo número de movimientos requeridos para reorganizar las fichas de manera que se cumplan las condiciones dadas por el erudito.

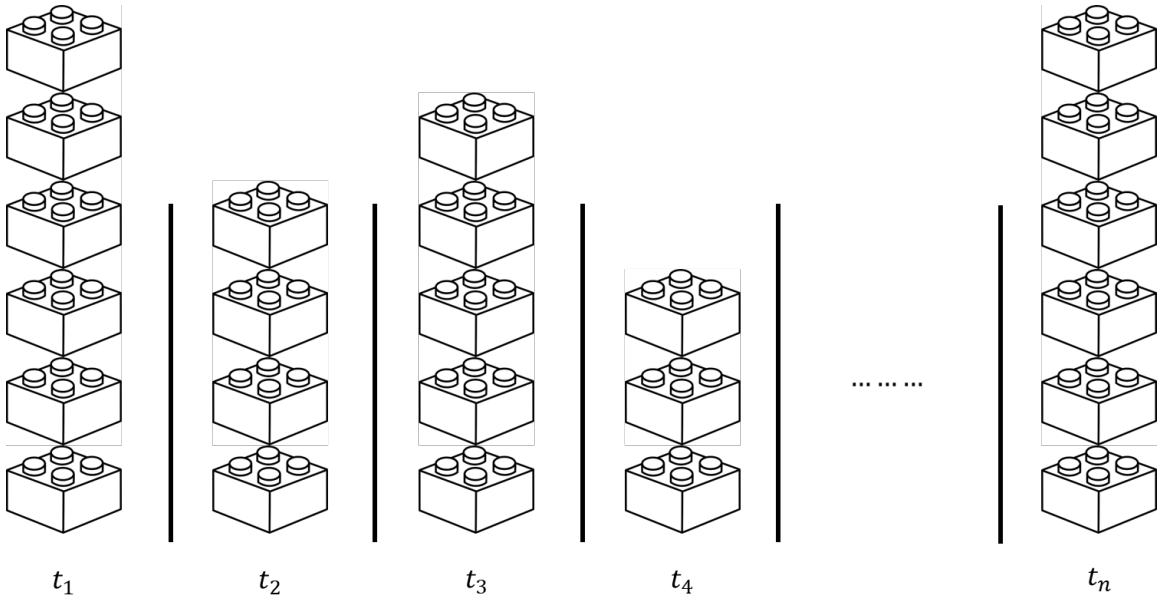

Fig 1. n torres de lego cada una con un número f_i de fichas.

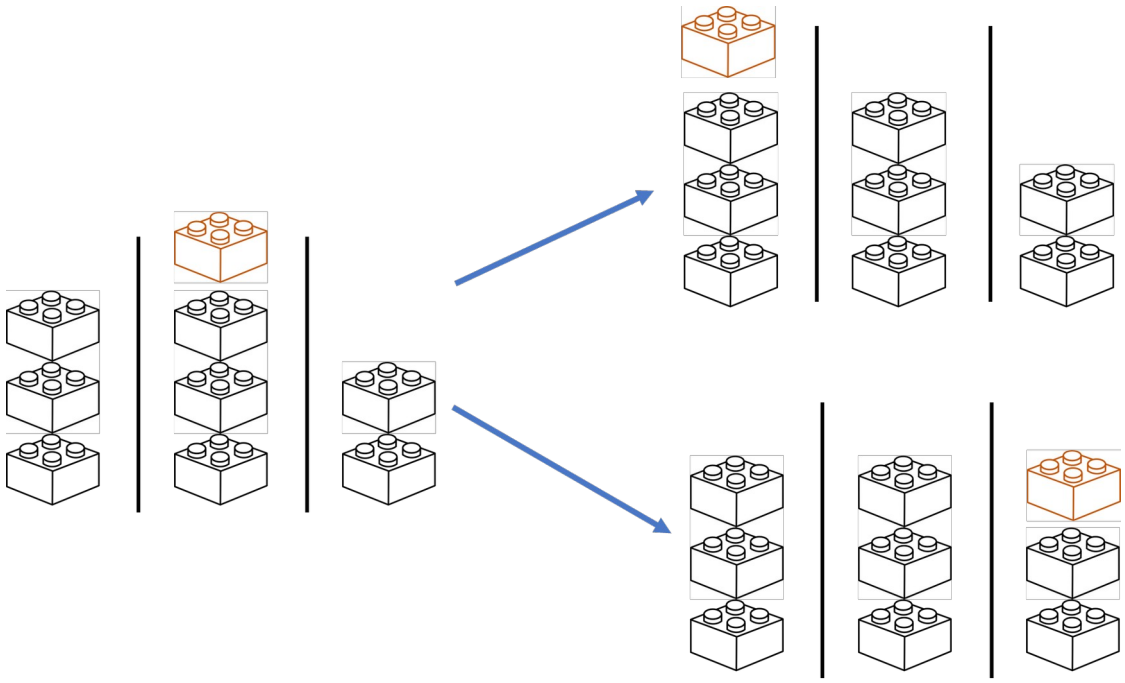

Fig 2. Posibles resultados de mover la ficha naranja en una distribución de torres arbitraria. A diferencia del ejemplo, si la ficha es de la primera o última torre solo existe una posibilidad de movimiento.

Ejemplo 1: A modo de ejemplo suponga la distribución de torres de la Figura 3.

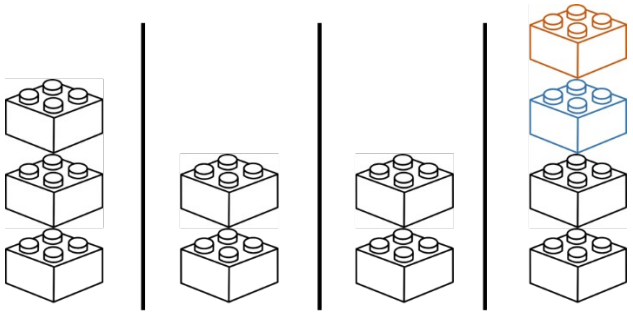

Fig 3. Distribución de torres inicial para el ejemplo 1.

La solución para este ejemplo es 3. Que corresponde a los 2 movimientos de la ficha naranja (ver Fig 4.) y un movimiento adicional de la ficha azul (ver Fig 5).

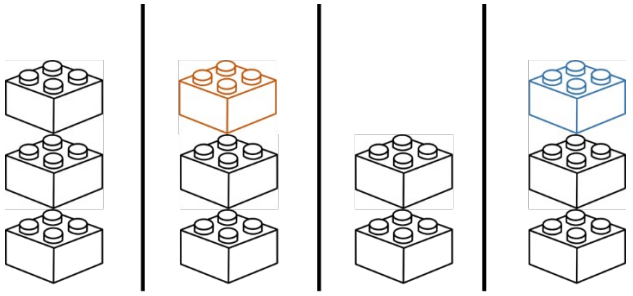

Fig 4. Dos movimientos de la ficha naranja.

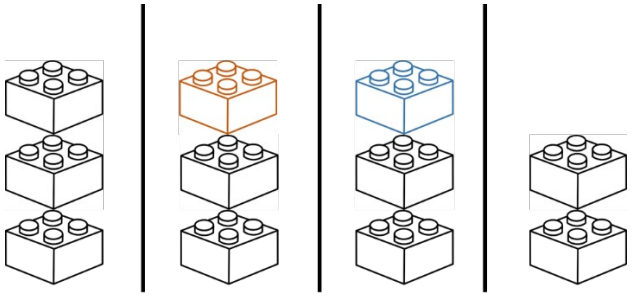

Fig 5. Un movimiento de la ficha azul.

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

La primera línea de entrada especifica el número de casos de prueba 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 está representado en una línea con la especificación del valor de n, seguido de la especificación de los f_i.

$$1 \leq n \leq 10^3$$

$$0 \leq \sum_{i=1}^{n} f_i \leq 10^3$$

Descripción de la salida

Para cada caso de prueba, imprimir el mínimo número de movimientos requerido para cumplir con la especificación del erudito amante de los legos.

Ejemplo de entrada / salida

Entrada	Salida
7 7 0 0 0 0 0 0 1 20 32 11 7 5 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 1 2 6 6 23 29 24 36 38 14 7 7 7 2 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 1 0 0 0 2 9 3 5 14 19 23 32 4 3 2 2 4 4 3 0 1 1	6 1 536 2 857 3 1

Nota: Se van a diseñar casos de prueba para valores de n, y f_i muchos 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: En un movimiento se puede mover mas de una ficha.

ESCENARIO 2: Se pueden mover fichas entre torres no necesariamente contiguas, pero si dentro de un radio de *r* torres.

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 `proyectoDalgoP1.zip`. Este archivo es una carpeta de nombre `proyectoDalgoP1`, 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 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 `ProblemaP1.java` o `ProblemaP1.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.).

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 (`ProblemaP1.pdf`).

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

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.