Diseño y Análisis de Algoritmos

Proyecto # 2: Tito el corrupto

Laura Victoria Riera Pérez
Marié del Valle Reyes

Cuarto año. Ciencias de la Computación.
Facultad de Matemática y Computación, Universidad de La Habana, Cuba

23 de abril de 2023

I. Repositorio del proyecto

https://github.com/computer-science-crows/algorithms-design-and-analysis

II. Definición inicial del problema

Tito se dió cuenta de que la carrera de computación estaba acabando con él y un día decidió darle un cambio radical a su vida. Comenzó a estudiar Ingeniería Industrial. Luego de unos años de fiesta, logró finalmente conseguir su título de ingeniero. Luego de otros tantos años ejerciendo sus estudios (¿?), consiguió ponerse a la cabeza de un gran proyecto de construcción de carreteras.

La zona en la que debe trabajar tiene n ciudades con m posibles carreteras a construir entre ellas.

Cada ciudad que sea incluida en el proyecto aportará a_i dólares al proyecto, mientras que cada carretera tiene un costo de w_i dólares.

Si una carretera se incluye en el proyecto, las ciudades unidas por esta también deben incluirse.

El problema está en que Tito quiere utilizar una de las habilidades que aprendió en sus años de estudio, la de la malversación de fondos.

Todo el dinero necesario para el proyecto que no sea un aporte de alguna ciudad, lo proveerá el país y pasará por manos de Tito.

El dinero aportado por las ciudades no pasará por sus manos.

Tito quiere maximizar la cantidad de dinero que pasa por él, para poder hacer su magia. Ayude a Tito a seleccionar el conjunto de carreteras a incluir en el proyecto para lograr su objetivo.

III. Definición en términos matemático - computacionales

I. Preliminares

IV. Línea de pensamiento

V. Algoritmo

I. Explicación del algoritmo

II. Complejidad Temporal

III. Complejidad espacial

VI. Generador de casos de prueba

En *src/app/generator.py* fue implementado un generador, el cual recibe una cantidad *s* de muestras a producir, genera valores random con el formato de entrada de los algoritmos implementados, halla la solución óptima con *backtrack* y las guarda en *json/test_cases.json*. Se generaron 3000 casos de prueba, con *n* máximo igual a 11, dado que, como se mencionó anteriormente, es lo que puede ejecutar el *backtrack*.

VII. Tester

En *src/app/tester.py* fue implementado un tester, que recibe una función y prueba el desempeño de la misma en cuanto a si obtuvo la solución óptima o no, y el tiempo que demoró en hacerlo, comparando con los casos de prueba obtenidos con el generador. Dichos resultados se muestran en consola de la siguiente forma:

Además, estos resultados se guardan en un *.json* con el nombre de la función en la carpeta tests. Las soluciones implementadas fueron testeadas para todos los casos de prueba generados y pueden encontrarse en *json/tests/simplex_solution.json* y *json/tests/hungarian_solution.json*

VIII. Comparación de soluciones implementadas

Referencias

[1] Cormen, Thomas H. y otros. *Introduction to Algorithms*. The MIT Press. 4ta Edición. Cambridge, Massachusetts. 2022.