Diseño y Análisis de Algoritmos. Problema 2: Problema en la pizarra

Jorge Soler González y Ernesto Alfonso Hernández

Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba

1. Definición del Problema

Problema en la pizarra

Un día iba David por su facultad cuando ve un cuadrado formado por nxn cuadraditos de color blanco. A su lado, un mensaje ponía lo siguiente: "Las siguientes tuplas de la forma x_1, y_1, x_2, y_2 son coordenadas para pintar de negro algunos rectángulos. (coordenadas de las esquina inferior derecha y superior izquierda)"Luego se veían k tuplas de cuatro enteros. Finalmente decía: "Luego de tener el cuadrado coloreado de negro en las secciones pertinentes, su tarea es invertir el cuadrado a su estado original. En una operación puede seleccionar un rectángulo y pintar todas sus casillas de blanco. El costo de pintar de blanco un rectángulo de hxw es el mínimo entre h y w. Encuentre el costo mínimo para pintar de blanco todo el cuadrado."

En unos 10 minutos David fue capaz de resolver el problema. Desgraciadamente esto no es una película y el problema de David no era un problema del milenio que lo volviera millonario. Pero ¿Sería usted capaz de resolverlo también?

2. Modelación del Problema

El problema consiste en hallar el costo mínimo para pintar el cuadrado de blanco, a partir de que este cuadrado tiene en su interior varios rectángulos pintados de negro, el costo de pintar de blanco un rectángulo negro está dado por el mínimo entre la altura y el ancho del mismo, de esta parte inicial sacamos 3 conlusiones en cuanto al comportamiento del costo: Sea h la altura de un rectángulo pintado de negro y sea w su ancho:

- $min(h, w) = w \Rightarrow$ pintar el rectángulo columna por columna no cambia el costo, dado que pintar una columna del rectángulo es de costo 1, luego la sumatoria de todas las columnas hace el ancho del rectángulo que era igual costo de pintarlo de blanco, lo mismo pasa cuando el mínimo es h
- $min(h, w) = w \Rightarrow$ añadirle nuevas filas a ese rectángulo con ese mismo ancho, no afecta el costo de pintar el mismo ya que el costo seguiría siendo w
- el costo máximo que tiene pintar el cuadrado de nxn completo es n

A partir de esto, podemos decir que el problema se resume en encontrar la menor combinación de filas/columnas que cubran todos los rectángulos pintados de negro.

3. Primera Solución

Como primera solución se nos ocurre separar la matriz en islas, donde estas islas son celdas de la matriz que están pintadas de negro y son adyacentes entre sí. Luego con estas isla, ir viendo una a una la mejor combinación de filas y columnas y luego quedarnos con la suma de todas, pero esto no funciona, ya que si se pinta una fila de una isla, puede ser que también se pinten celdas de otra isla, cosa que no tuvimos en cuenta y que nos sirvió para sacar las conclusiones expuestas en la modelación del problema.

4. Mejor solución encontrada

Para esta solución vamos a utilizar las premisas planteadas en la modelación del problema.

En este punto nos damos cuenta que un subproblema de este problema es que para cada celda pintada de negro en la matriz, es necesario saber si esa celda se escoje pintar como fila o como columna, teniendo esto, ya tendríamos una distribución de filas/columnas que cubren todas las celdas pintadas de negro de la matriz. Ahora el problema es encontrar la menor combinación de filas/columnas que cubran todos los rectángulos pintados de negro. Si se lograse armar un grafo en el que los vértices representaran las filas y las columnas entonces el problema de buscar la menor combinación de filas/columnas que participen en los rectángulos pintados de negro se resume a buscar la cobertura mínima de vértices en dicho grafo.

Para esto vamos a armar un grafo con nxn vértices, donde cada vértice representa una fila ó una columna de la matriz original. Luego existe una arista de un vértice a otro si en su intercción hay una celda pintada de negro, o sea solo existen aristas entre vértices que representen filas, con vértices que representen columnas, nunca entre ellos. Nos queda un grafo bipartito por definición sería:

Sea M la matriz original, T el conjunto de tuplas que dan de entrada, t_i una de esas tuplas.

Se define el grafo bipartito G de la siguente manera $G = \{(V, E) : V = R \cup C\}$ donde R es el conjunto de los vértices que representan una fila y C el conjunto de los vértices que representan una columna en la matriz original, luego $E = \{(r, c) : r \in R \wedge c \in C \wedge M[r, c] \in t_i\}$

Notar que en este grafo, la cobertura mínima de vértices representa la menor combinación de filas/columnas que participan en los rectángulos pintados de negro, dado que las aristas del grafo representan todas las celdas pintadas de negro en la matriz, por tanto también representa el costo mínimo de pintar de blanco la matriz.

El problema de hallar la cobertura mínima de vértices de un grafo es NP-Completo, pero nuestro grafo es bipartito y por el Teorema de König que a grandes rasgos, establece una equivalencia entre la cobertura mínima de vértices de un grafo bipartito y la cardinalidad del emparejamiento máximo, por lo tanto, nos enfocamos en encontrar el emparejamiento máximo del grafo.

Para esto vamos a construir una red de flujo, vamos a modificar el grafo G de la siguiente manera:

Sea $G' = (V', E')$ donde $V' = V \cup \{s, t\}$ y $E' = \{(s, r) : r \in R\} \cup \{(r, c) : (r, c) \in E\} \cup \{(c, t) : c \in C\}$, para terminar con la construcción de la red de flujo se le asigna capacidad 1 a todas las aristas de E'. Hallar el flujo máximo en este grafo es equivalente a la cardinalidad del emparejamiento máximo, que a su vez es el costo mínimo como se dijo anteriormente. La demostración de esto último se hizo en clase práctica. Para calcular el flujo máximo utilizaremos el algoritmo de Ford-Fulkerson.

4.1. Complejidad temporal

Armar el grafo descrito es $O(n^2)$ siendo n x n la dimensión de la matriz. Luego el algoritmo de Ford-Fulkerson es $O(|E||f^*|)$ pero como estamos hallando emparejamiento máximo en un grafo bipartito, el flujo máximo es a lo sumo el $min(R, C) = n$, lo que implica que el flujo máximo es $O(n)$, luego $|E|$ es a lo sumo n^2, por tanto el algoritmo para resolver el problema tiene complejidad temporal: $O(n^2 + n * n^2) = O(n^3)$

4.2. Pseudocódigo

Para una mejor modelación del problema nos apoyamos en una clase llamada BiGraph que se encuentra en el archivo tools.py, mientras que el main, se encuentra en el archivo solve.py