Complejidad de Húngaro

Probar la complejidad $O(n^4)$ del algoritmo Húngaro y dar una idea de cómo se la puede reducir a $O(n^3)$

Demostración de $O(n^4)$

Primero, tenemos que entender cómo es el funcionamiento del algoritmo Húngaro:

- Dada la matriz $n \times n$ de costos M, realizamos lo siguiente para obtener el matching *perfecto* de menor costo:
  1. Calculamos el mínimo m de cada fila y se lo restamos a todos los elementos de la fila. Se hace lo mismo con las columnas
  2. Se obtiene un *matching inicial* cualquiera
  3. Dado un matching parcial, queremos extenderlo (agregar 1 nodo)
     1. Para ello, comenzamos etiquetando las filas que **no** tienen elementos en el matching actual. Las agregamos a la cola
     2. Para cada elemento de la cola, realizamos lo siguiente:
        1. Si es fila, entonces revisamos *todos* los elementos y nos interesan los nulos. Para ellos, si su columna no está etiquetada, la etiquetamos y agregamos a la cola
        2. Si es columna, entonces hay dos casos:
           1. Si tiene un elemento nulo en el matching, entonces, si su fila no está etiquetada, la etiquetamos y agregamos a la cola
           2. Si tiene elementos nulos pero ninguno está en el matching, entonces terminamos y pasamos a (4) donde se extiende el matching
     3. Si la cola se queda vacía y no podemos extender el matching, entonces debemos cambiar la matriz. Luego se continúa el proceso con la misma cola y etiquetas
        1. Sean S y $\Gamma(S)$ las filas y columnas etiquetadas, respectivamente, entonces lo que se hace es:
           1. Calcular el mínimo m de $S \times \Gamma(\bar{S})$
           2. Restar m de las filas S
           3. Sumar m de las columnas $\Gamma(S)$
  4. La forma de extender el matching es agregar el elemento nulo en el que terminamos al matching e ir sacando y metiendo en función de las etiquetas que

tengan los elementos hasta que lleguemos a una de las filas iniciales. Además, se eliminan las etiquetas de las filas y columnas, y se vacía la cola.
5. Seguimos con el paso 3

Teniendo esto en cuenta, podemos dividir el algoritmo en las siguientes partes:

- **Inicialización**: paso 1
  - Como se calcula mínimo de cada fila, tenemos que se itera por todos los elementos $O(n^2)$
  - Como se les restan estos números, se usa otro $O(n^2)$
  - Como se hace lo mismo para las columnas, tenemos complejidad total de $O(4 \times n^2) = O(n^2)$
- **Matching inicial**: paso 2
  - Como tratamos de poner los $0s$ para armar un matching inicial, debemos iterar por $O(n^2)$ elementos
- **Extensión del matching** (en un lado): paso $3 \to$ se aplica n veces ya que se va agregando de a 1 lado
  - Como es revisar filas y columnas, su complejidad (sin tener en cuenta el cambio de matriz) es de $O(n^2)$
    - Revisar las columnas es $O(1)$ porque queremos ver solo si está libre o no
    - Revisar las filas es $O(n)$ porque tenemos que ver todos los elementos nulos
    - Luego, la peor complejidad es la de revisar todas las filas hasta poder extender el matching en un lado ($O(n^2)$)
  - Dado esto, si consideramos que la complejidad de cambiar la matriz es CM y tenemos que hacer un máximo de T cambios de matrices para agregar un lado, entonces la complejidad de este paso va a ser $O(n^2) + CM \times T$
- **Cambio de la matriz** (para un caso): paso $3.3 \to$ hay T cambios posibles
  - Calcular el mínimo es, a lo sumo, $O(n^2)$ ya que hay que revisar los elementos
  - Restar m de las filas S es $O(n)$
  - Sumar m de las columnas $\Gamma(S)$ es $O(n)$
  - Luego, la complejidad de hacer un cambio es $O(n^2)$. Por ello, $CM = O(n^2)$
- **Cambio del matching**: paso 4
  - Dado que tenemos $m : m \text{ es } O(n)$ elementos en el matching parcial y queremos extenderlo para que tenga $m+1$, entonces debemos hacer a lo sumo $2m+1$ cambios (mover todos los del matching parcial y agregar el nuevo elemento).
  - Además, el "reseteo" de las etiquetas es, de a lo sumo, n filas y n columnas, por lo que es $O(2n) = O(n)$
  - En total, entonces, este paso es $O(n)$

Luego, entonces, tenemos que si juntamos todas las complejidades, el total del húngaro es:

$$O(n^2) + O(n^2) + O(n) \times [O(n^2) + CM \times T + O(n)] = O(n^3) + O(n) \times (CM \times T)$$

Sabemos que $CM = O(n^2)$. Luego, solo queda ver T.

- Primero, veamos la siguiente propiedad *clave*: "luego de un cambio de matriz, o crece el matching, o crece el S"
  - Más precisamente: "supongamos que la extensión del matching parcial de ceros se detiene al encontrar el S con $|S| > |\Gamma(S)|$, que se hace el cambio de matriz como se describió arriba y se continúa con la búsqueda con las etiquetas guardadas (se mantienen).
    Entonces, con la nueva matriz obtenida, o bien al correr el algoritmo se agrega un nuevo lado al matching, o bien se detiene con un S_2 con $|S_2| > |\Gamma(S_2)|$ y este cumple que $|S_2| > |S|$"
  - Demostración
    - Para esto, veamos que, sea $x \in S, v \in \bar{\Gamma(S)} : m = M_{x,v}$, entonces al restar y sumar m en la nueva matriz, esta tendrá un nuevo 0 en la entrada x, v
    - Como $x \in S$, al *continuar* el algoritmo de búsqueda, va a agregar a v a la cola (i.e., la fila x etiqueta a la columna v)
    - Ahora hay dos casos principales:
      - v **no está en el matching parcial**
        - Como v no está en el matching parcial, entonces la columna está libre y se puede extender
      - v **está el matching parcial**
        - Como v está en el matching parcial, entonces está matcheada con alguna fila z.
        - Como $v \notin \Gamma(S)$, entonces $z \notin S$. Luego, se etiqueta a z con el tag de v (i.e., se agrega a S_2)
        - Si luego de agregar a z y continuar la búsqueda se extiende el matching, todo OK.
          Caso contrario, tenemos que para en algún momento con $|S_2| > |\Gamma(S_2)|$ y, por lo visto antes, $|S_2| > |S|$
- Dicho esto, entonces, a lo sumo se van a hacer n cambios de matrices (T) ya que por la propiedad anterior S no puede crecer más que $O(n)$ veces porque solo hay n filas.

Entonces, visto esto y en base a lo anterior, tenemos que la complejidad del Húngaro es:

$$O(n^3) + O(n) \times (T \times CM) = O(n^3) + O(n) \times O(n) \times O(n^2) = O(n^3) + O(n^4) = O(n^4)$$

por lo que se demuestra.

Idea para $O(n^3)$

La idea para bajar la complejidad es respecto a CM para amortizar los cálculos del mínimo y las sumas y restas en las demás partes del código que ya son $O(n^2)$. Esto "esconde" las operaciones.

Para hacer esto, sin embargo, usamos la suposición de que los números que manejamos son, aproximadamente, menores a 10^{19} para que las operaciones entre ellos puedan ser $O(1)$ en la computadora (i.e., no afecte a la complejidad).

Dicho esto, la idea es **nunca** cambiar los valores de los elementos al realizar el *cambio* de matriz, sino que simplemente guardamos en un array que a la fila i se le resta m, mientras que a la columna j se le suma k.

- Digamos que estos arrays son $row[], col[]$ respectivamente, luego, el valor actual del elemento con posición x, v es $M[x][v] - row[x] + col[v]$ donde $M[x][v]$ es el valor inicial de esa posición

Además, respecto al cálculo del mínimo, este lo vamos a calcular cuando estemos en la parte de la búsqueda y etiquetado ya que para cada fila de S, debemos iterar sí o sí por todos sus elementos.

- Para ello, vamos a considerar el array $min[]$ inicializado en números enormes y en cada pasada por una fila de S, cuando revisemos sus elementos, vamos a actualizar los mínimos de las columnas
- Luego, de allí vamos a ver que si $min[i] = 0$ entonces $i \in \Gamma(S)$ ya que tiene alguna fila vecina con la cual comparte el 0
  - Por ello, lo que nos interesa para calcular m es el mínimo de los valores de min que no son 0

Dada esta idea, podemos notar que su complejidad se esconde en las diferentes partes del programa en operaciones $O(1)$ y en la sección CM solo queda para que sea $O(n)$ puesto que simplemente se deben agregar valores a row y col (cada uno con tamaño n).

Por ello, dada la complejidad de Húngaro, tenemos que:

$$O(n^3) + O(n) \times (T \times CM) = O(n^3) + O(n) \times O(n) \times O(n) = O(n^3) + O(n^3) = O(n^3)$$