Codificación en C del algoritmo hungaro.

Ya probamos que la complejidad es $O(n^3)$. Veamos ahora una codificación concreta. En clase la di en pseudocodigo, aqui en C.

1: Primera Aproximación

Para saber cuando parar, deberiamos tener un contador que nos diga cuantas filas no matcheadas hay. Cuando el contador llegue a cero, paramos.

Al tratar de extender el matching , escaneamos filas y columnas. Si al escanear una columna la encontramos vacia, entonces sabemos que podemos extender el matching, si no, debemos seguir escaneando filas. Si al terminar de escanear filas no hay columnas nuevas, debemos cambiar la matriz. Para ello, pondremos una bandera auxiliar que dirá si hemos hallado una columna libre o no. Como hallar una columna libre produce un orgasmo, a esta bandera la llamaremos "frigidez", i.e. no hay orgasmo. Esta parte sería asi:

Primera Aproximación Al Algoritmo

```
while (filas_no_matcheadas >0)
{
  frigidez=1;
  NuevasFilas=1;
  while (frigidez && NuevasFilas)
  {
    EscanearFilas;
    EscanearColumnas;
  }//end while(frigidez && NuevasFilas)
 if (frigidez) CambiarMatriz;
    else {
         ExtenderMatching;
         filas_no_matcheadas=filas_no_matcheadas-1;
         }
 if (filas_no_matcheadas >0)  ReinicializarEtiquetas;
}//endwhile(filas_no_matcheadas>0)
```

frigidez y NuevasFilas seran cambiadas ya sea en EscanearFilas o en EscanearColumnas. Si salimos del while pero todavia hay frigidez, debemos cambiar la matriz. si no, es porque salimos porque se puede extender el matching. Lo extendemos, lo cual reduce el numero de filas no matcheadas en uno. Si todavia quedan filas no matcheadas, reinicializamos las etiquetas para poder seguir.

Observar que un problema de esta forma es que si tenemos frigidez, cambiamos la matriz, y luego reinicializamos las etiquetas, lo cual nos obligaria a escanear desde el principio.

2: Eliminación de EscanearColumnas

Observemos una cosa: EscanearFilas hay que hacerlo, para ir etiqueteando las columnas correspondientes con el nombre de la fila. Pero cuando hacemos EscanearColumnas, en realidad solo hacemos una de dos cosas: si la columna esta matcheada, agregamos la fila con la cual esta matcheada a S, y luego etiqueteamos esa fila con el nombre de la columna. Pero como ese es en realidad el matching, esta etiqueta ya la tendremos y por lo tanto no hace falta hacerlo. Es decir, deberiamos tener de todos modos registros que nos guarden el matching, es decir, para cada fila, deberiamos tener un

registro que nos diga con que columna esta matcheada esa fila, y para cada columna, un registro que nos diga con que fila esta matcheada. Asi, la etiqueta de una fila en realidad sera la columna con la cual esta matcheada. Por lo que en realidad (casi) no nos hace falta escanear la columna: si ya esta matcheada, la etiqueta de la fila ya la tenemos, y si no esta matcheada, es cuando esta libre y tenemos orgasmo. Asi que en realidad al "escanear la columna", solo debemos revisar el registro que diga con quién esta matcheada: si ese registro indica una fila, bien, y si no, es que esta libre. Cuando lo implementemos, podemos numerar las filas y columnas desde 0 en adelante, y originalmente darles a los registros "ColQueMatcheaLaFila" y "FilaQueMatcheaLaCol" todos valores negativos. Asi, si "FilaQueMatcheaLaCol[y]" es negativo, significa que esta libre.

Y para no tener que hacer el ciclo de EscanearFilas-EscanearColumnas, lo que podemos hacer es, al EscanearFilas, y encontrar una columna adecuada, EN ESE MISMO MOMENTO podemos hacer EscanearColumnas solo con esa columna. Y esto se reducira a chequear que "FilaQueMatcheaLaCol[y]" sea no negativo. Si lo es, agregamos "FilaQueMatcheaLaCol[y]" a S y seguimos. Si no lo es, tenemos orgasmo. Es decir, haremos:
"**escanear fila, etiqueteando columnas, y chequeando en ese mismo momento si esa columna tiene matching o no**".

3: Observacion: la busqueda es BFS

Otro detalle: era visualmente facil ver las filas etiqueteadas y obtener el S. Esto lo podemos programar de dos formas: podriamos darle un registro a cada fila, que nos diga si esta etiqueteada o no, y entonces luego recuperar el S. En clase tambien es fácil al comenzar a escanear las filas, recorrer visualmente las filas para ver cuales son las filas que estan recientemente etiqueteadas. Para ello, deberiamos colocar otra etiqueta que nos diga si la fila ya ha sido escaneada o no. Pero entonces deberiamos recorrer todas las filas cada vez que querramos escanear, y esto complica la complejidad.

En vez de ello, es mas facil guardar directamente el S a medida que lo vamos construyendo, como una cola de la cual no borramos nada. Las "filas recientemente etiqueteadas" seran entonces simplemente las nuevas, y recorriendo S como una cola las barremos todas.

En particular esto significa que la busqueda la haremos en modo realmente EK, es decir, BFS. Con esto quiero decir que las filas etiqueteadas las revisaremos realmente en el orden en el que se ponen en la cola, y no en orden alfabetico como en clase.

4: Como salir de EscanearFilas

Queda un problema: ¿como sabemos cuando salir de EscanearFilas porque no hay mas filas nuevas para escanear? Simplemente, recorremos S con el metodo estandard de cola que estudiaron en Algoritmos II creo. Cuando se haya recorrido todo, se acabó: es una especie de FOR dinamico: debemos tener un registro s que guarde el tamaño de S, y un registro i que permita ver el i-esimo elemento de S. El s y el i iran creciendo, pero siempre tendremos $i \leq s$; cuando sean iguales paramos. (los elementos de S los numeramos entre 0 y $s-1$)

Asi, el algoritmo quedaria en realidad:

```
while (filas_no_matcheadas >0)
{
  i=0;
  frigidez=1;
  while (frigidez)
  {
    EscanearFilas();
  if (frigidez) CambiarMatriz();
  }//end while(frigidez).
 ExtenderMatching();
 filas_no_matcheadas=filas_no_matcheadas-1;
  if (filas_no_matcheadas >0)  ReinicializarEtiquetas();
}//endwhile(filas_no_matcheadas>0)
```

5: EscanearFilas en detalle

El EscanearFilas en principio seria simple: recorremos la fila, y donde encuentramos ceros, etiqueteamos la columna correspondiente, y chequeamos como dijimos a ver si "FilaQueMatcheaLaCol[y]" es o no negativo. Todo esto lo pondremos dentro de una subrutina, por lo que tendriamos:

EscanearFilas

```
while (i<s && frigidez )
   {
      x=S[i]; //escanear fila i del S.
      y=0;
      while (y<n && frigidez) ChequearLugarXY();//aca crecera el  "y'' o habra orgasmo
      // End escanear fila i del S
      i++;
   }//endwhile  principal
```

(el doble chequeo de frigidez es para que, al encontrar una columna libre, no sigamos buscando mas columnas ni escaneando mas filas)

La subrutina sería (pero en realidad es mas complicada, ver mas abajo)

ChequearLugarXY

```
if (Matriz[x,y]==0&&(etiqueta_de_columna[y]<0))//si ya esta etiqueteado no lo hacemos
  {
  if (FilaQueMatcheaLaCol[y]<0) frigidez=0; //la columna esta libre.
     else EtiquetearColumna;// El s crece aca.
  }
if (frigidez) y++;//solo cambio el "y" si debo seguir buscando.
```

donde "EtiquetearColumna" es:

```
etiqueta_de_columna[y]=x;
S[s]=FilaQueMatcheaLaCol[y];
s++;
```

es decir, etiqueteo, incremento el S y el s.

6: Extender Matching

Esta es simple:

ExtenderMatching:

```
/* Begin extender el matching solo si la columna "y",
 etiqueteada con "x" esta libre.*/
extension=0;//para saber cuando terminar
while (extension==0)
{
  k=col_que_matchea_la_fila[x];//<0 si x no esta matcheada
  col_que_matchea_la_fila[x]=y;
  fila_que_matchea_la_col[y]=x;
  /*        cambiamos  matching en fila "x"
                       de la columna "k" a la "y".
     con lo cual  "y" queda matcheada con "x".
                         La k queda libre.*/

  if (k<0) extension=1;//si "x" era una fila no matcheada
  else
    {
      x=etiqueta_de_columna[k];/*si "k" era columna real,
                               estaba etiqueteada:
                            cambio el "x" a la etiqueta
                          y sigo con  el loop*/
      y=k;
    }
}//endwhile del extension
```

7: CambiarMatriz

Queda la subrutina de CambiarMatriz. Para reducir la complejidad a $O(n)$ se requieren algunos trucos, que tambien nos obligarán a modificar un poco las subrutinas anteriores. (pero queremos hacerlo sin modificar sus complejidades)

Tenemos dos problemas: por un lado, como encontrar el minimo sin tener que recorrer los $O(n^2)$ elementos de $S \times \overline{\Gamma(S)}$, y el otro, como luego actualizar todos los valores de la matriz sin perder tiempo $O(n^2)$ en ello.

Veamos primero el segundo problema y luego el primero:

8: Cambio de la matriz una vez que tenemos el minimo

Salvo en las partes 1) y 2) donde restamos realmente los minimos; luego, en la parte 4), como explicamos cuando probamos que la complejidad del algorimto era $O9n^3$), en vez de cambiar realmente la matriz,haremos un cambio "virtual":

(en realidad tambien podriamos hacer un cambio virtual en 1) y en 2), pero no vale la pena pues solo los usamos una vez).

Guardamos en unos registros cuanto es lo que deberiamos restarle a cada fila de S y sumarle a las columnas de $\Gamma(S)$, pero no lo sumamos ni restamos en ese momento para no tener complejidad $O(n^2)$ sino solo $O(n)$ en el cambio de la matriz. (pues solo cambiamos $O(2n)$ registros contra $O(n^2)$ entradas de la matriz). Entonces, cuando buscamos "ceros", recien ahi calculamos cual es el valor real de la matriz en el lugar (x, y). Esto modifica la subrutina de ChequearLugarXY, pero la complejidad es la misma, asumiendo operaciones aritmeticas rapidas.

Tendriamos entonces una subrutina para restar el minimo de $S \times \overline{\Gamma(S)}$ (que llamaremos "Minimo") a las filas de S:

RestarMinimo

```
for (i=0;i<s;i++)
     restar_de_fila[S[i]]+=Minimo;
```

es decir, como dijimos, tendremos unos contadores que nos dirán, para cada fila, cuanto le tenemos que restar a esa fila. En esta subrutina, incrementamos el valor de ese contador en el

valor de "Minimo" que se supone que lo tenemos. Al incrementar el valor de lo que tenemos que restar, es como si virtualmente le hubieramos restado a toda la fila el valor "Minimo". Solo se lo restamos a las filas de S, y esto es facil de hacerlo sin tener que recorrer todas las filas, simplemente recorriendo la "cola" $S[]$.

Luego necesitariamos una subrutina que hiciera lo mismo pero con las columnas de $\Gamma(S)$, Pero, hay una diferencia con el anterior: en el anterior, es facil sumarlo solo a las filas de S, pues las tenemos. Acá, habria que recorrer todas las columnas y ver si cada columna es o no una columna en $\Gamma(S)$. Pero si vamos a recorrer todas las columnas, observemos que, en las columnas en $\overline{\Gamma(S)}$ es el único lugar donde pueden aparecer nuevos ceros. Por lo tanto, ya que vamos a revisar todas las columnas de todas formas, simplemente podemos hacer dos cosas:

1) Si la columna esta en $\Gamma(S)$, le sumamos el minimo (i.e., le restamos al contador)

2) Si la columna esta en $\overline{\Gamma(S)}$, chequeamos si hay nuevos ceros en ella. Esto nos evitara tener que re-escanear las filas ya escaneadas buscando nuevos ceros.

Tendriamos entonces:

```
CambiarMatriz
CalcularMinimo;//esto todavia nos falta ver como lo hacemos
RestarMinimo;
SumarMinimoYHallarNuevosCeros;
```

(luego explico bien la subrutina "`SumarMinimoYHallarNuevosCeros`".

9: Cómo hallar el minimo

Para calcular el minimo sin tener que chequear todas las entradas, se hace el siguiente truco, inspirado por el algoritmo de Dijkstra de caminos mas cortos:

Introducimos un registro que contenga, para cada columna, el minimo de esa columna cuando solo se miran las filas en S. (En el codigo lo llamaremos "min_de_col_int_S": minimo de la columna en su intersección con S).

Esto lo hacemos MIENTRAS armamos el S. Es decir, cuando estamos escaneando la fila x, al mirar el lugar x, y, updateamos el minimo de la columna "y" si es necesario. Esto solo nos lleva $O(1)$ operacion mas en ese lugar.

Una ventaja de esto es que directamente podemos saber que $\Gamma(S)$ son aquellas columnas que tienen min_de_col_int_S igual a cero. Es decir, para encontrar $\Gamma(S)$ no hace falta escanear S (lo que tomaria complejidad $O(n)$ por cada fila, con un total de $O(n^2)$ en total) sino solo chequear una cierta etiqueta en cada columna, con complejidad por columna $O(1)$ y complejidad total $O(n)$.

Entonces, cuando el algoritmo pare momentaneamente porque el matching no se puede extender, simplemente recorremos todas las columnas, calculando el minimo no cero de estos minimos. (digo el minimo no cero, porque solo debo escanear sobre $\overline{\Gamma(S)}$: como dijimos arriba, aquellas columnas cuyo minimo en S de cero son justamente las de $\Gamma(S)$). Este minimo no cero de minimos sera justamente el minimo de $S \times \overline{\Gamma(S)}$, con lo cual, al solo recorrer n columnas, lo encontramos en tiempo $O(n)$.

Tambien guardaremos en un registro **en qué fila** se produce este minimo. (si hay mas de uno, nos quedamos con el primero que lo encontró.)

Esto para que nos sirve? Recordemos que cuando hacemos la busqueda, si no podemos extender el match, i.e, nos da el S, cambiamos la matriz, dejando el match como estabamos, y entonces debiamos re-escanear todas las filas ya etiqueteadas, lo cual puede complicar la complejidad.

Pero dijimos que en vez de hacer esto, haremos el re-escaneo mientras cambiamos la matriz. Es decir, al encontrar los nuevos ceros al cambiar la matriz, etiqueteamos la columna con el nombre de la fila donde esta ese cero nuevo. Y como las columnas reciben una sola etiqueta, en realidad no hay que chequear todos los ceros nuevos, sino basta con el primer cero nuevo en una columna, porque los demas de esa columna no se usaran, pues ya estará etiqueteada.

Por eso, guardamos el lugar donde se produce el minimo, porque luego de restar habra un 0, y guardamos en esa columna, cúal es la fila que por primera vez halla el minimo. Asi podremos hacer el re-escaneo en forma eficiente. Entonces tendriamos:

`Calcular Minimo:` subrutina obvia de calcular minimos:

```
Minimo=2147483647;//``infinito"
   for (y=0;y<n;y++)
     if (min_de_col_int_S[y] && min_de_col_int_S[y]<Minimo)
       Minimo=min_de_col_int_S[y];
```

el if es decir que min_de_col_int_S este en $(0, Minimo)$.

Esto es claramente $O(n)$.

ChequearLugarXY queda ahora:

```
if (min_de_col_int_S[y]) //si el min es 0 alguien mas ya la etiqueteo: no se revisa.
 {
     xy_real=Matriz[x][y]-restar_de_fila[x]+sumar_a_col[y];
       /*el costo - lo que hay que restar a la fila  mas el incremento por columnas
         i.e., este es el A[x][y] ``real'' que deberiamos haber calculado*/
      if (xy_real<min_de_col_int_S[y])
           /* si no es menor, como estoy en el caso de min_de_col_int_S mayor a cero,
             tendremos en particular que no es cero, asi que no hace falta revisarlo.
              Ademas, si no es menor es mayor o igual, por lo que min_de_col_int_S,
              que guarda el minimo, no cambiara*/
        {
           if (xy_real==0)
             {
                 if (FilaQueMatcheaLaCol[y]<0)
                         frigidez=0; //i.e., orgasmo: la columna esta libre.
                  else EtiquetearColumna();//`s" crece.
               }
             else ActualizarMinimoColS(); //endif xy_real==0
           }//endif xy_real<mincol
    }//endif mincol>0
if (frigidez) y++;//solo cambio el "y" si debo seguir buscando.
```

A pesar de que bastante mas complicado que el original que teniamos, la complejidad sigue siendo $O(1)$, salvo quizas por la subrutina ActualizarMinimoColS. Pero esta es simplemente:

`ActualizarMinimoColS`:

```
min_de_col_int_S[y]=xy_real;
fila_donde_esta_min[y]=x;
```

Nos quedaba la subrutina:

```
SumarMinimoYHallarNuevosCeros:
    for (k=0;k<n;k++)
      if (min_de_col_int_S[k])//i.e., en  complemento(Gamma(S))
         NuevosCeros();
      else
        sumar_a_col[k]+=Minimo;
```

i.e., como dijimos, si la columna esta en $\overline{\Gamma(S)}$, buscamos nuevos ceros, y si esta en $\Gamma(S)$, actualizamos el registro de lo que hay que sumarle.

La NuevosCeros es:

```
  if (frigidez) //para no seguir si ya hay una columna libre
  {
     min_de_col_int_S[k]-=Minimo;/*esto hara que los que tenian el minimo sean
                                   los nuevos ceros, i.e.., se agreguen a Gamma(S)*/
      if (min_de_col_int_S[k]==0)//solo miro los que son cero
       {
         x=fila_donde_esta_min[k];
         y=k; /*necesito el "y" ahora: lo que pasa es que si hay orgasmo quedara fijo aca,
                    y el k lo necesito para updatear Gamma(S).*/
          if (FilaQueMatcheaLaCol[k]<0)// es una * por lo que voy a poder extender el match.
             frigidez=0;//i.e., orgasmo
           else //si no estoy en el caso anterior, etiqueteo para seguir la busqueda
              EtiquetearColumna();//aca tambien se usa el "y"
       }//endif mincol==0
   }//endif frigidez
```


También hay un pequeño cambio en EtiquetearColumna: hay que agregar la línea:

```
min_de_col_int_S[y]=0;
```

pues si estoy etiqueteando una columna es porque hay un cero en ella.