Tarea 2

Saul Ivan Rivas Vega

Diseño y análisis de algoritmos

Equipo Completo:
Yadira Fleitas Toranzo
Diego de Jesús Isla Lopez
Saul Ivan Rivas Vega

5 de septiembre de 2019

1. Ejercicio 1. Extensión del problema del matrimonio.

1.1. Ofrece una instancia del problema con al menos 5 mujeres tal que el conjunto N satisfaga que $N \neq \emptyset$ y $N \neq H \times M$; Encuentra todos los apareamientos estables posibles.

	1st	**2nd**	**3rd**	**4th**	**5th**
Carla	Pedro	Tavo	Paco	Beto	Pablo
Luna	Pablo	Tavo	Beto	Pedro	Paco
Ana	Paco	Tavo	Beto	Pablo	Pedro
Diana	Tavo	Pedro	Beto	Paco	Pablo
Nia	Tavo	Pedro	Pablo	Beto	Paco

Cuadro 1: Matriz de preferencias de las mujeres.

	1st	**2nd**	**3rd**	**4th**	**5th**
Pedro	Carla	Diana	Nia	Ana	Luna
Pablo	Carla	Ana	Nia	Luna	Diana
Tavo	Carla	Luna	Diana	Nia	Ana
Beto	Luna	Carla	Ana	Diana	Nia
Paco	Ana	Carla	Diana	Nia	Luna

Cuadro 2: Matriz de preferencias de los hombres.

$$N = \{(Pablo - Ana), (Tavo - Nia), (Tavo - Carla)\} \tag{1}$$

Único Emparejamiento Estable:

$$S = \{(Pedro{-}Carla), (Paco{-}Ana), (Beto{-}Diana), (Tavo{-}Luna), (Pablo{-}Nia)\} \tag{2}$$

Demostración Un emparejamiento estable que da el algoritmo es óptimo para los hombres y pésimo para las mujeres si es que los hombres proponen, sin embargo si las mujeres proponen el emparejamiento es óptimo para las mujeres y pésimo para los hombres, entonces si el emparejamiento es el mismo quiere decir que tanto la mejor y peor pareja para cada quién es la misma y por lo tanto única.

1.2. Utiliza las definiciones de *pareja válida* y *mejor pareja válida* para demostrar que en cualquier ejecución del algoritmo si (h, m) está en el apareamiento resultante, entonces m es la mejor pareja válida de h

Demostración Por contradicción.
Supongamos que un hombre h al terminar la ejecución del algoritmo no fue emparejado con su *mejor pareja válida* en este caso m. Es decir que en el emparejamiento resultante S pasa que $(h, m) \notin S$.
Entonces h debió ser rechazado por m en favor de otro h' que fuera de mayor preferencia para m.
Tomemos la primera vez que esto pasa, que h fuera el primer hombre rechazado durante la ejecución del algoritmo.
Eso quiere decir que h' no ha sido rechazado ni por la mujer que mas prefiere, pues en el algoritmo los hombres se proponen en orden de preferencia.
Por lo anterior decimos que m es la mujer de mayor preferencia para h'.
Por la definición de *pareja válida* debe existir un emparejamiento estable S' donde $(h, m) \in S'$. En S' la pareja de h' sería otra mujer m'.
Ahora tenemos $(h', m') \in S'$ y $(h, m) \in S'$. Pero como m es la mujer de mayor preferencia para h' y m prefiere a h' antes que a h, hay una inestabilidad y por lo tanto dicho emparejamiento S' no es estable.
Así demostramos que para todo hombre no existe un emparejamiento estable que lo tenga emparejado con una mujer que sea de mayor preferencia que su pareja al terminar la ejecución del algoritmo.
Finalmente decimos que en cualquier ejecución del algoritmo si (h, m) está en el apareamiento resultante, entonces m es la mejor pareja válida de h.

2. Ejercicio 2.

En clase revisamos que es falso lo siguiente: para cualesquiera funciones f y g cuyo codominio es $\mathbb{R}$ se cumple que si existe $c > 0$ y una cantidad infinita de $x \in Dom(f)$ satisfacen la desigualdad $f(x) \geq cg(x)$, entonces $f \in \Omega(g)$.

Demuestra que existen (dando un ejemplo) funciones f y g cuyo codominio es $\mathbb{R}$ y una constante $c > 0$ tales que para una cantidad infinita de $x \in Dom(f)$ la desigualdad $f(x) \leq cg(x)$ es verdadera y además $f \in \Omega(x)$

Demostración Usando las siguientes funciones:

$$f(x) = x \tag{3}$$

$$g(x) = 1 \tag{4}$$

Tomando en cuenta que el Dominio de las Funciones es $\mathbb{R}$ tenemos una cantidad infinita de x que cumplen que $0 \leq x \leq 1$ y en esa cantidad infinita de x se cumple que existe una c, en este caso $c = 1$, donde $f(x) \leq cg(x)$, sin embargo si tomamos $x_0 = 1$ para la definición de Ω tendremos que a partir de ese x_0 para toda $x \geq x_0$ será verdadero que $f(x) \geq cg(x)$ y por lo tanto $f \in \Omega(g)$.
Finalmente se demuestra con el ejemplo que si existen f y g para las cuales existe una constante $c > 0$ tal que para una cantidad infinita de $x \in Dom(f)$ la desigualdad $f(x) \leq cg(x)$ es verdadera y además $f \in \Omega(x)$.

3. Ejercicio 3. Considera la siguiente definición:

f es $\Omega'(g)$ si existe $c > 0$ tal que $f(n) \geq cg(n) \geq 0$ para un número infinito de valores de n.

3.1. Demuestra que, para cualesquiera f y g tales que $f, g \geq 0$, ya sea que $f \in O(g)$ o $f \in \Omega'(g)$

Demostración. Por contradicción, supongamos que existen $f, g \geq 0$ tales que no cumplen $f \in O(g)$ ni $f \in \Omega'(g)$, es decir que $f \notin O(g)$ y $f \notin \Omega'(g)$.

Para que $f \notin \Omega'(g)$ quiere decir que no existe una constante $c \geq 0$ tal que $f(n) \geq cg(n) \geq 0$ para un número infinito de valores de n, entonces debe tener un número finito o ningún valor de n donde $f(n) \geq cg(n) \geq 0$ para toda constante $c \geq 0$.
Pero en f al terminar el numero finito de valores (o al inicio si no hay) se empieza a cumplir que $f(n) < cg(n)$ para toda $c \geq 0$ pero eso entra en la definición para que $f \in O(g)$ lo que implica contradicción a la suposición inicial de que $f \notin O(g)$ porque f no puede estar y no estar al mismo tiempo en $O(g)$.
Por lo tanto decimos que no existen $f, g > 0$ tal que no se cumpla que $f \in O(g)$ o $f \in \Omega'(g)$.
Entonces demostramos que para cualesquiera f y g tales que $f, g \geq 0$, ya sea que $f \in O(g)$ o $f \in \Omega'(g)$.

3.2. Prueba que de hecho existe f tal que $f \in O(g)$ y $f \in \Omega'(g)$.

Demostración. Encontrando f y g que cumplan con que $f \in O(g)$ y $f \in \Omega'(g)$.
Tomando en cuenta que $Dom(f)$ y $Dom(g)$ es $\mathbb{R}$ y también su codominio es $\mathbb{R}$ podemos proponer las siguientes funciones:

$$f(n) = 1 \tag{5}$$

$$g(n) = n \tag{6}$$

Tenemos una cantidad infinita de n que cumplen que $0 \leq n \leq 1$ y en esa cantidad infinita de n se cumple que existe una c, en este caso $c = 1$, donde $f(n) \geq cg(n)$, sin embargo si tomamos $n_0 = 1$ para la definición de O tendremos que a partir de ese n_0 para toda $n \geq n_0$ será verdadero que $f(n) \leq cg(n)$ y por lo tanto $f \in O(g)$.
Finalmente se demuestra con el ejemplo que si existen f y g para las cuales existe una constante $c > 0$ tal que para una cantidad infinita de n la desigualdad $f(n) \geq cg(n)$ es verdadera, por lo tanto $f \in \Omega'(g)$ y además $f \in O(g)$.

3.3. Demuestra que el inciso anterior no es verdadero si se cambia Ω por Ω'

3.3.1. Demuestra que es falso que, para cualesquiera f y g tales que $f, g \geq 0$, ya sea que $f \in O(g)$ o $f \in \Omega(g)$

Demostración. Encontrando f y g que cumplan con que $f \notin O(g)$ y $f \notin \Omega(g)$.
Podemos proponer las siguientes funciones:

$$f(n) = \begin{cases} 1, & \text{n es par} \\ n^2, & \text{n no es par} \end{cases} \tag{7}$$

$$g(n) = n \tag{8}$$

$f \notin O(g)$ por que no existen una $c > 0$ ni un n_0 tales que para toda $n \geq n_0$ se cumpla $f(n) \leq cg(n)$, puesto que en el caso donde n no es par el valor de n^2 será mayor al de n.

$f \notin \Omega(g)$ por que no existen una $c > 0$ ni un n_0 tales que para toda $n \geq n_0$ se cumpla $f(n) \geq cg(n)$, puesto que en el caso donde n es par el valor de 1 será menor al de n.
Como $f \notin O(g)$ y $f \notin \Omega(g)$ se demuestra que es falso que para cualesquiera f y g tales que $f, g \geq 0$, ya sea que $f \in O(g)$ o $f \in \Omega(g)$.

3.3.2. Demuestra que es falso que, existe f tal que $f \in O(g)$ y $f \in \Omega(g)$.

Observación: Aunque la descripción de la tarea menciona que hay que demostrar la falsedad del enunciado realmente demostraremos que es verdadero pues es la definición misma de Θ.

Demostración. Encontrando f y g que cumplan con que $f \in O(g)$ y $f \in \Omega(g)$.
Podemos proponer las siguientes funciones:

$$f(n) = n \tag{9}$$

$$g(n) = n \tag{10}$$

$f \in O(g)$ por que existen una $c > 0$, en este caso $c = 1$ y un n_0, en este caso $n_0 = 0$ tales que para toda $n \geq n_0$ se cumple $f(n) \leq cg(n)$.
$f \in \Omega(g)$ por que existen una $c > 0$, en este caso $c = 1$ y un n_0, en este caso $n_0 = 0$ tales que para toda $n \geq n_0$ se cumple $f(n) \geq cg(n)$.
Como $f \in O(g)$ y $f \in \Omega(g)$ se demuestra que es verdadero que existe f tal que $f \in O(g)$ y $f \in \Omega(g)$.

4. Ejercicio.4 Busca las definiciones de omega minúscula y o minúscula, explícalas en tus propias palabras y presenta ejemplos de funciones que te parezcan interesantes respecto a estas notaciones (explica por que te lo parecen), da propiedades y su relación con O y Ω

4.1. o (o minúscula)

La pequeña o es para denotar una cota superior que no puede ser «tocada» por f es decir que a diferencia de O si $f \in o(g)$ no puede pasar que $f(n) = cg(n)$, tiene que ser estrictamente menor. Formalmente $f \in o(g)$ si existen una constante $c > 0$ y una n_0 tal que para toda $n \geq n_0$ se cumple que $f(n) < cg(n)$.
Un ejemplo interesante es ver el efecto que tiene usar o en funciones iguales:

$$f(n) = n \tag{11}$$

$$g(n) = n \tag{12}$$

En este caso $f \notin o(g)$ porque la definición requiere que sea estrictamente menor.

$$f(n) = n \tag{13}$$

$$g(n) = n^2 \tag{14}$$

En este caso $f \in o(g)$.

Propiedades y relación con O Podemos ver que si $f \in o(g)$ entonces $f \in O(g)$ ya que para que $f \in O(g)$ basta con que sea menor o igual en la condición para encontrar una constante $c > 0$ y una n_0 tal que para toda $n \geq n_0$ se cumple que $f(n) \leq cg(n)$.
Otra propiedad es que si $f \in o(g)$ entonces $g \in \omega(f)$ lo cual veremos a continuación al ver la definición de ω.

4.2. ω (omega minúscula)

La pequeña ω es para denotar una cota inferior que es estricta al igual que la pequeña o solo que ahora tiene que ser estrictamente mayor su crecimiento. Formalmente $f \in \omega(g)$ si existen una constante $c > 0$ y una n_0 tal que para toda $n \geq n_0$ se cumple que $f(n) > cg(n)$.
Un ejemplo es ver el efecto que tiene usar ω en funciones iguales que de igual forma que o implica que $f \notin \omega(g)$.

$$f(n) = n^2 \tag{15}$$

$$g(n) = n^3 \tag{16}$$

En este caso $f \in \omega(g)$, si nos damos cuenta prácticamente requiere ser incluso de un orden superior el polinomio para que se cumpla la condición.

Propiedades y relación con Ω Podemos ver que si $f \in \omega(g)$ entonces $f \in \Omega(g)$ ya que para que $f \in O(g)$ basta con que sea mayor o igual en la condición para encontrar una constante $c > 0$ y una n_0 tal que para toda $n \geq n_0$ se cumple que $f(n) \geq cg(n)$.

5. Ejercicio 5.

Presenta un algoritmo de complejidad $O(n^2)$ para el problema 3-SUM: Dado un arreglo de n enteros diferentes, encuentra tres de ellos cuya suma sea 0.

5.1. Algoritmo

Data: Un arreglo A de numeros diferentes ordenados de tamaño n.
Result: 3 indices i, j y k que determinan los 3 números de A que sumados dan 0
Marcamos una variable bandera de encontrado con valor 0;
encontrado := 0;
for $i := 0$ *to* $n - 1$ *step* 1 **do**
    Para cada elemento i reiniciamos los indices de búsqueda;
    $j := 0$;
    $k := n - 1$;
    Asignamos el valor a buscar como el inverso aditivo del elemento i en A;
    valor_buscado := $-1 \times A[i]$;
    while $j < k$ **do**
        Si j es igual a i incrementamos j;
        if $i = j$ **then**
            $j := j + 1$;
        end
        Si k es igual a i decrementamos k;
        if $i = k$ **then**
            $k := k - 1$;
        end
        Validamos que k se menor a n y que j sea menor a k;
        if $j >= k$ *or* $k >= n$ **then**
            break;
        end
        suma_actual := $A[j] + A[k]$;
        Validamos que la suma sea el valor buscado;
        if *suma_actual* = *valor_buscado* **then**
            Marcamos la variable bandera y terminamos el ciclo While;
            encontrado := 1;
            break;
        else
            Si la suma es menor aumentamos j o disminuimos k si es mayor;
            if *suma_actual* < *valor_buscado* **then**
                $j := j + 1$
            else
                $k := k - 1$
            end
        end
    end
    Si la variable bandera esta marcada terminamos el ciclo For;
    if *encontrado* = 1 **then**
        break;
    end
end
Reportar los indices i, j y k si se encontró;

Algorithm 1: Algoritmo para el problema de 3-SUM

5.2. Complejidad

Prueba en detalle que es correcto y que su complejidad es en realidad $O(n^2)$.

Demostración

Primero demostremos que el algoritmo termina:
El algoritmo termina porque en el ciclo *while* interno en cada iteración se incrementa el valor de j o decrementa el valor de k, si es que *suma_actual* no es igual a *valor_buscado*, y cada valor de A es referenciado a lo más una vez, ya sea por j o por k. Entonces el *while* interno a lo más puede tener n iteraciones, que es haber hecho alguna combinación de incrementos a j y decrementos a k tal que se referenciaron todos los valores en A, y como el ciclo *while* se manda a llamar una vez por cada elemento en A serían n veces que se corre el ciclo *while*. Finalmente como se realiza n veces el ciclo que a lo mas hace n iteraciones se llega a que el algoritmo termina después de n^2 operaciones por lo tanto es de complejidad $O(n^2)$.

Ahora demostremos que el algoritmo da la respuesta:
El ciclo *for* recorrerá cada indice i, de los n posibles, buscando si hay una pareja j y k distintos entre sí y distintos a i tales que cumplan que sumados sean igual al inverso aditivo del elemento en el índice i (*valor_buscado*), la búsqueda de esta pareja j y k se hace en el ciclo *while* en el cual se mantendrá la siguiente invariante en cada iteración no-final, *suma_actual* es mayor o menor a *valor_buscado*.
Si en la iteración *suma_actual* es igual a *valor_buscado* entonces es una etapa final por que termina el algoritmo y reporta los valores de los indices i, j y k.
Ahora seguimos en una iteración no-final donde *suma_actual* es menor a *valor_buscado*, debemos incrementar j lo que hará crecer *suma_actual* puesto que los elementos en A estan ordenados. Observemos que, si incrementamos k también aumentaría *suma_actual*, sin embargo todos los indices mayores a k ya fueron analizados y de haber pertenecido a la respuesta lo habríamos tomado junto con otro índice menor o igual a j, y como no fue así, no tiene sentido que si *suma_actual* es menor a *valor_buscado*, debemos incrementar k.

En otro caso en una iteración no-final donde *suma_actual* es mayor a *valor_buscado*, debemos decrementar k lo que hará disminuir *suma_actual*. Por una razón similar al punto anterior no tiene sentido decrementar j por que esos indices ya fueron considerados y saltados.
Si en el *while* se puede buscar una pareja de indices tales que la suma de los elementos a los que hacen referencia den un valor determinado, podemos decir que como esa búsqueda se hace por el inverso aditivo de cada elemento en A aseguramos que de existir dicha pareja de indices el algoritmo termina con una pareja de indices j y k tales que la suma de los elementos a los que hacen referencia es igual al inverso aditivo del elemento referenciado por i y por lo tanto si se suman esos 3 elementos tendríamos como resultado 0 que es lo que se busca en la definición del problema.

Finalmente como el algoritmo da la respuesta y termina demostramos entonces que es correcto y tiene una complejidad de $O(n^2)$.

6. Ejercicio 6.

En la implementación que revisamos del algoritmo de Gale-Shapley se muestra un algoritmo para invertir la lista de preferencia de un estudiante. Demuestra que el algoritmo propuesto tiene complejidad $O(n)$ donde n es el número de hospitales.

Demostración El algoritmo al que nos referimos es:

```
for i := 1 to n do
    rank[pref[i]] := i;
end
```

Algorithm 2: Algoritmo para invertir la lista de preferencia de un estudiante.

El algoritmo solo hace uso de un ciclo *for* (por cada estudiante), si decimos que el acceso a los arreglos y la asignación son una operación elemental que este en O(1) podemos decir entonces que el número de operaciones estará dado por el número de iteraciones en el ciclo *for*. En el ciclo se recorrerá por todos los valores entre 1 y n así pues decimos que hará n operaciones elementales y por lo tanto es de orden $O(n)$.