Segon Parcial: Ampliació d'Algorísmia **QP 2023-2024**

Una solución **3 de juny de 2024**

---

1. **Ejercicio 1 [4 punts] (European debates.)**

   La asociación para la promoción de la Identidad Europea está planificando una jornada formada por una serie de debates sobre diferentes temas europeos. Disponen de una lista de *participantes* formada por las personas que se han comprometido a participar en la jornada, asisitiendo a todos los debates, siempre que la asociación les envíe una invitación formal.

   El comité organizador en vista de los temas previstos y de las experiencias previas ha seleccionado para cada debate dos listas disjuntas de personas: la lista *éxito* y la lista *fracaso*. El comité examina estas listas con optimismo. La presencia de al menos una de las personas en la lista de éxito o la ausencia de al menos una de las personas en la lista de fracaso de un debate en particular garantiza que este debate será exitoso.

   El comité organizador enfrenta el problema de seleccionar el subconjunto de personas a quienes la asociación enviará una invitación formal. La asociación desea invitar a un conjunto de personas que garantice que se maximice el número de debates exitosos (según la regla anterior).

   Diseñad una 2-aproximación para el problema. Justificar su corrección y eficiencia.

   Una solución: El problema se puede reducir al problema de MaxSAT utilizando la siguiente transformación. Tenemos una variable x_i por cada posible participante i, por cada debate generamos una clausula conteniendo como variables afirmadas las correspondientes a los participantes en su lista de éxito y negadas las correspondientes a los participantes en su lista de fracaso.

   Si una selección de participantes S la hacemos corresponder con una asignación en la que $x_i = 1$ si y solo si $i \in S$, el número de debates exitosos con esta selección de invitados coincide con el número de clausulas satisfechas por la asignación correspondiente y viceversa.

   Como tanto el coste de obtener la fórmula en CNF y com el de recuperar el conjunto desde una asignaciónm es polinómico, podemos utilizar cualquiera de los algoritmos de 2-aproximación polinómicos vistos en clase para MaxSAT. Por ejemplo, invitar a todos o a nadie dependiendo de cual de las dos opciones hace que haya más debates exitosos.

2. **Ejercicio 2 [4 punts] (Tightly linked group.)**

Los estudiantes del curso MEI PTDMA deben diseñar un algoritmo para revelar un grupo estrechamente vinculado en una red social. Para aprobar el curso basta con diseñar una App que identifique un conjunto de *distorsión* lo más pequeño posible en la red social. Un *conjunto de distorsión* es un subconjunto de participantes cuya eliminación deja un subconjunto de vértices *estrechamente vinculados*. Para este ejercicio, podéis asumir que la red social es un grafo conexo y no dirigido, y que un subconjunto de vértices X está *estrechamente vinculado* si y sólo si, para cada par de vértices distintos en X, o están conectados directamente por una arista o ambos están conectados mediante una arista a un tercer vértice en X.

Demostrad que la versión decisional del problema parametrizado por el tamaño del conjunto de distorsión pertenece a FPT.

Una solución: En la versión decisional del problema la entrada es un grafo y un entero no negativo k, el parámetro es k y la propiedad a decidir es si exixte un conjunto de distorsión X con $|X| = k$.

La propiedad que define un conjunto de distorsión implica que entre cada par de vértices en X tiene que haber un camino de longitud 1 o dos. Por lo tanto si tenemos dos vértices que están a distancia mayor que 2 en el grafo, uno de los dos vértices tiene que pertenecer al conjunto de distorsión. Utilizando esta propiedad podemos diseñar un algoritmo de búsqueda acotada para el problema de la forma habitual.

```
Distorsion(G, k)
if diam(G) = 2 then
    return True
if k = 0 then
    return False
Sean u, v ∈ G a distancia mayor que 2 en G
return Distorsion(G − u, k − 1) or distorsion(G − v, k − 1)
```

Distorsion ejecuta como mucho dos llamadas recursivas y la altura del érbol de recursión es k. Así el número total de llamadas es 2^k. Por cada llamada recursiva la operación más costosa es la de determinar si el diamétro es dos y en caso de no serlo obtener dos vértices a distancia mayor que 2. Esto se puede resolver realizando un BFS de profundidad 2, por cada vértice, con coste en caso peor $O(n^2)$. El cost etotal del algoritmo es $O(2^k n^2)$ por lo que es un algoritmo FPT.

Como hemos encontrado un algoritmo FPT el problema considerado pertenece a la clase FPT.

3. **Ejercicio 3 [2 punts] (2-colorable?)**

Considerad el siguiente algoritmo de streaming:

```
1: procedure PartTree(int n, stream s)
2:     F = ∅, x = 1
3:     while not s.end() do
4:         (u, v) = s.read()
5:         if x then
6:             if F ∪ {(u, v)} does not contain a cycle then
7:                 F = F ∪ {(u, v)}
8:             else
9:                 if F ∪ {(u, v)} contains an odd cycle then
10:                    x = 0
11:    On query, report x
```

(a) Analizad el coste de PartTree.

(b) Demostrad que la respuesta a una consulta es 1 y sólo si el grafo procesado hasta ese momento es 2-colorable.

Una solución:

(a) Para analzar el coste de PartTree necesitamos fijar una implementación que nos permita detectar ciclos y ciclos de longitud impar enficientemente.

Sabemos que un grafo es 2-colorable si y solo si es no contiene ningún ciclo de longitud impar. En la implementación que propongo el algoritmo mantendrá por una parte las componnetes conexas, utilizando un ED de union-find, junto con su tamaño y un 2-coloreado de los vértices del grafo. Inicialmente tendremos un set por cada vértice, el tamaño es 1 para cada set y todos los vértices tienen color 1.

Al incorporar una arista podemos detectar si hay un ciclo viendo que los dos extremos están en la misma componente conexa, y en este caso, el ciclo será de longitud impar si los dos extremos tiene el mismo color.

Si no añadimos la arista al grafo, la ED interna no se modifica. Si añadimos la arista al grafo, antes de hacer el Union de los dos sets actualizaremos el 2-coloreado. Si u y v están en dos componente conexas diferentes y tienen colors diferentes la coloración se mantiene. En caso contrario cambiaremos el color (al color complementario) a todos los vértices que estén en la componente conexa con menos vértices.

El algoritmo hace solo una pasada por los datos y el coste en caso peor por elemento es $O(\alpha(n) + n) = O(n)$. El coste $O(n)$ se debe al posible cambio de coores, esta operación se realizará como mucho $n - 1$ veces.

(b) Para demostrar qie ña la respuesta a una consulta es 1 y sólo si el grafo procesado hasta ese momento es 2-colorable. Por un lado tenemos que si en algún momento $x = 0$ el algoritmo detecta un ciclo negativo y por lo tanto el grafo considerado hasta este momento, y cualquier ampliación con más aristas, no es 2-colorable. Por lo que la respuesta será correcta en relación al grafo procesado.

Por otra parte, mientras que $x = 1$, el algoritmo mantiene una 2-coloración del grafo procesado hasta el momento. Para ver que este coloreado es siempre válido lo podemos

hacer por inducción. Inicialmente es correcto ya que el grafo procesado no tiene aristas. Supongamos que antes de procesar una arista (u, v) tenemos una 2-coloraicón y que después de procesarla $x = 1$. Teiendo en cuenta que en una 2-coloración si intercambiamos en una componnete conexa el color de cada vértice por el color complenetario obtenemos otra 2-coloración. La coloración despues de porcesar la arista, tanto si incorporamos (u, v) a F como si no, es una 2-coloración del grafo ampliado. Por lo que la respuesta será correcta en relación al grafo procesado.