Presentación

Esta PEC profundiza en el concepto de complejidad computacional que cubre los contenidos estudiados en los módulos 6 y 7 de la asignatura. Los ejercicios trabajan los conceptos de medida de complejidad, la reducción y completitud, la clase NP-completo y algunos de los problemas intratables más importantes que se conocen.

Competencias

En esta PEC se trabajan las siguientes competencias del Grado de Ingeniería Informática:

- Capacidad para utilizar los fundamentos matemáticos, estadísticos y físicos para comprender los sistemas TIC.
- Capacidad para analizar un problema en el nivel de abstracción adecuado en cada situación y aplicar las habilidades y conocimientos adquiridos para resolverlo.

Objetivos

Los objetivos concretos de esta PEC son:

- Entender los conceptos de intratabilidad y no-determinismo.
- Conocer las diferentes clases de complejidad y saber clasificar los problemas en cada una de estas.
- Entender el concepto de reducción entre problemas y saber demostrar cuando un problema es NP-completo.
- Reconocer problemas intratables que aparecen de forma habitual en informática y en ingeniería.
- Entender y saber aplicar las técnicas básicas de reducción polinómica de los problemas NP-completos.

Descripción de la PEC a realizar

1. (Valoración de un 20 % = 5 % + 5 % + 5 % + 5 %)

   Indicad la complejidad temporal de cada uno de los siguientes problemas. Todas las respuestas deben estar justificadas, las respuestas que consistan únicamente en soluciones tipo '$O(n)$' sin indicar cómo se ha llegado a ellas no recibirán valoración.

   a) BUSCA_SUBMATRICES: Dada una matriz M de $n \times n$ posiciones y un número k, devuelve CIERTO si existe una submatriz de M, con filas y columnas contiguas, tal que la suma de sus elementos sea igual a k, FALSO en caso contrario.

   b) COMPLETO: Dado un grafo G (sin aristas múltiples ni lazos) para el que nos dan su lista de adyacencias, devuelve CIERTO si G es un grafo completo, FALSO en caso contrario.

   c) SUMA_MATRIZ: Dada una matriz de $n \times n$ posiciones, devuelve CIERTO si la suma de los elementos de alguna fila o columna es igual a la suma del resto de elementos de toda la matriz (excepto esa fila o columna), FALSO en caso contrario.

   d) PARES_IMPARES: Dado un vector de n posiciones, devuelve CIERTO si la suma de los elementos de las posiciones pares es igual a la suma de los elementos de las posiciones impares, FALSO en caso contrario.

Solución:

a) BUSCA_SUBMATRICES: $O(n^6)$. Se trata de encontrar de cuántas formas diferentes podemos seleccionar filas y columnas consecutivas (para que sean submatriz). Una vez encontradas todas las submatrices debemos sumar sus elementos para comprobar si son iguales a k.

Para las filas, podremos escoger n combinaciones de filas comenzando por la primera, $n-1$ combinaciones de filas comenzando por la segunda, ..., 1 para la última fila.

Procediendo de igual forma para las columnas y sabiendo que ambas selecciones son independientes, nos quedan $\frac{n(n+1)}{2}$ formas de escoger las filas por $\frac{n(n+1)}{2}$ formas de escoger las columnas y, por lo tanto, la selección de la submatriz es de complejidad temporal $O(n^4)$.

Cada una de estas submatrices tendrá, a lo sumo, $n \times n$ elementos, por lo tanto la suma de los elementos tendrá una complejidad temporal de $O(n^2)$.

Combinando ambas complejidades llegamos a $O(n^6)$.

b) COMPLETO: Teniendo la lista de adyacencias solamente tenemos que contar el número de elementos a los que es adyacente cada vértice. Esta operación puede realizarse en tiempo $O(n)$ si podemos acceder directamente al número de elementos en la lista de adyacencias de cada vértice, o en $O(n^2)$ en el peor caso si tenemos que recorrer cada lista de adyacencias de forma individual.

c) SUMA_MATRIZ: $O(n^2)$. Podemos realizar una primera pasada sumando todos los elementos de la matriz, con una complejidad temporal de $O(n^2)$. Llamamos a esa suma *total*. A continuación, podemos realizar la suma de cada fila o columna por separado comprobando si esa suma es igual a $total/2$. Realizar estas sumas de filas y columnas se puede realizar también con una complejidad temporal de $O(n^2)$.

Opcionalmente podríamos realizar una única pasada almacenando la suma de las filas, columnas y total sin que cambie la complejidad temporal del problema.

d) PARES_IMPARES: $O(n)$. Solamente tenemos que recorrer el vector una vez totalizando, en dos variables diferentes, la suma de las posiciones pares e impares. Al terminar solo debemos comprobar si ambas sumas son iguales.

---

2. (Valoración de un 15 % = 5 % + 5 % + 5 %)

a) Dados los siguientes problemas:

- CLIQUE: Dado un grafo G y un entero k, devuelve CIERTO si existe un subgrafo completo de G de orden k, FALSO en caso contrario.
- SUMA_PARES: Dado un vector de enteros v y un entero k, devuelve CIERTO si la suma de todos los elementos en las posiciones pares de v es igual a k, FALSO en caso contrario.
- SUMA_SUBCONJUNTO: Dado un vector de enteros v y un número entero k, devuelve CIERTO si existe un subconjunto de elementos de v tal que su suma sea igual a k, FALSO en caso contrario.
- SAT: Dada una fórmula booleana V en FNC, devuelve CIERTO si existe una asignación de valores de verdad a las variables de V que se evalúe a CIERTO, FALSO en caso contrario.
- AJEDREZ: Dada una configuración de un tablero de ajedrez, devuelve CIERTO si un jugador tiene una secuencia de jugadas ganadora a partir de esta configuración, FALSO en caso contrario.

Indicad si las siguientes reducciones polinómicas son posibles basándote en la jerarquía de complejidad y las propiedades de las funciones de reducción:

a) SUMA_PARES $\leq_p$ CLIQUE
b) SUMA_SUBCONJUNTO $\leq_p$ AJEDREZ
c) AJEDREZ $\leq_p$ SAT

Solución:

a) Es posible. SUMA_PARES tal como está definido es un problema P y siempre podemos transformar un problema P a CLIQUE que es NP-Completo, siendo posible esa reducción en tiempo polinómico.

b) Es posible. SUMA_SUBCONJUNTO puede ser verificable en tiempo polinómico y, por lo tanto, pertenecerá a NP. Por su parte, AJEDREZ, tal como se indica en los apuntes, es un problema EXP y sabemos que la reducción de un problema NP a uno EXP es posible.

c) No es posible. Independientemente de si P=NP o no, sabemos que no es posible reducir en tiempo polinómico un problema EXP a un problema NP, sea completo o no.

---

3. (Valoración de un 15 % = 7,5 % + 7,5 %)

Dados los siguientes problemas:

- PARES_IMPARES: Dado un vector de números enteros, devuelve CIERTO si la suma de los elementos de las posiciones pares es igual a la suma de los elementos de las posiciones impares, FALSO en caso contrario.
- SUMA_SUBCONJUNTO: Dado un vector de enteros v y un número entero k, devuelve CIERTO si existe un subconjunto de elementos de v tal que su suma sea igual a k, FALSO en caso contrario.
- SUMA_PARES: Dado un vector de enteros v y un entero k, devuelve CIERTO si la suma de todos los elementos en las posiciones pares de v es igual a k, FALSO en caso contrario.

Indicad si las siguientes son funciones de reducción polinomiales válidas. Recuerda que todas las respuestas deben ser justificadas, respuestas del tipo 'No es válida' o 'Es válida' sin justificar por qué es válida o no lo es, no recibirán valoración.

A continuación definimos algunas funciones utilizadas en las funciones de reducción dadas.

- La función **append** añade un elemento a un array.
- La función **MOD** devuelve el resto de la división de dos números.
- La función **DIV** devuelve el resultado de la división entera de dos números.

a) PARES_IMPARES $\leq_p$ SUMA_PARES:

```
(1)  función Reducción_PI_a_SP(v)
(2)    inicio
(3)      iguales ← FALSO
(4)      // longitud nos dice el número de posiciones del array
(5)      n ← longitud(v)
(6)      // Creamos un array que tiene 0 en las posiciones impares
(7)      // y lo rellenamos con las posiciones pares de v
(8)      array_pares ← array() // Inicializamos un array para pares
(9)      // Calcularemos la suma de las posiciones impares
(10)     suma_impares ← 0 // Inicializamos la suma a 0
(11)     para i ← 1 hasta n
(12)           si MOD(i, 2) ← 0 entonces // Si la posición del vector original es par
(13)             array_pares.append(v[i])
(14)           sino // Si la posición es impar
(15)             array_pares.append(0)
(16)             suma_impares ← suma_impares + v[i]
(17)           finsi
(18)     finpara
(19)     si SUMA_PARES(array_pares, suma_impares) entonces
(20)       iguales ← CIERTO
(21)     finsi
(22)     retorno iguales
(23)   fin
```

b) SUMA_SUBCONJUNTO $\leq_p$ SUMA_PARES:

```
(1)  función Reducción_SS_a_SP(v,k)
(2)    inicio
(3)      encontrado ← FALSO
(4)       // longitud nos dice el número de posiciones del array
(5)      n ← longitud(v)
(6)      // Creamos todos los subconjuntos posibles de v
(7)      para i ← 1 hasta 2^n
(8)            vector ← array() // Inicializamos un array
(9)            check ← i
(10)           para j ← 1 hasta n
(11)                 vector.append(0) // Ponemos un 0 en cada posición impar del array
(12)                 si MOD(check, 2) = 0 entonces
(13)                   vector.append(v[j])
(14)                 sino
(15)                   vector.append(0)
(16)                 finsi
(17)                 check ← DIV(check, 2)
(18)           finpara
(19)           si SUMA_PARES(vector,k) entonces
(20)             encontrado ← CIERTO
(21)           finsi
(22)     finpara
(23)     retorno encontrado
(24)   fin
```

Solución:

a) Es una reducción válida. La reducción se realiza en tiempo polinomial. Utiliza el resultado de la función SUMA_PARES para decidir el resultado de la función PARES_IMPARES, aunque no se utilice de forma directa el resultado de SUMA_PARES sino que es necesario evaluarlo antes del retorno.

b) No es una función de reducción en tiempo polinomial válida ya que, debido a que utiliza todos los subconjuntos posibles del vector original, no se ejecutará en tiempo polinómico.

---

4. (Valoración de un 10 % = 5 % + 5 %)

   Representad, en FNC, las siguientes funciones:

a) $a \wedge (b \vee (c \wedge a))$.

b) $b \wedge (a \wedge c) \vee (c \vee d)$.

Solución:

a) $a \wedge (b \vee c)$.

1. Distributiva: $a \wedge (b \vee c) \wedge (b \vee a)$.
2. Absorción: $a \wedge (b \vee c)$.

b) $c \vee d$.

1. Asociatividad: $(a \wedge b \wedge c) \vee (c \vee d)$.
2. Absorción: $c \vee d$ (el resultado a la izquierda del primer $\vee$ solo es cierto si c es cierto, por lo tanto no aporta nada en la solución final donde seguirá siendo cierto solo si c es cierto o si d es cierto.

---

5. (Valoración de un 20 % = 10 % + 10 %)

El gobierno del régimen autoritario de la república (distópica e imaginaria) de Kol¢hhovia está preocupado por el aumento del libre pensamiento entre sus ciudadanos.

Con el fin de evitar la proliferación de conversaciones que puedan derivar en un fin de la convivencia pacífica o que puedan poner en duda la legitimidad del gobierno, ha decidido tomar cartas en el asunto y solucionar el problema desde un punto de vista racional y objetivo.

Para ello, nos solicitan que les ayudemos a plantear, en forma de problemas complejos conocidos (y estudiados en el módulo 7 de la asignatura), las siguientes situaciones:

a) En un ejercicio de simplificación de la gestión de la ciudadanía, Kol¢hhovia quiere clasificar a sus ciudadanos entre *Disidentes* y *Agentes de seguridad*.

Gracias a su programa 'El gobierno **te escucha**', Kol¢hhovia tiene datos de todas las conversaciones que se han llevando a cabo entre sus ciudadanos, con todos los participantes correctamente identificados.

Con el objetivo de que los intereses comunes estén correctamente representados, nos piden encontrar, basándonos en los datos históricos de conversaciones, la mejor forma de reclutar al mínimo número de ciudadanos como *Agentes de seguridad*, garantizando que en cada conversación participa, como mínimo, uno de ellos.

b) Debido al buen clima de entendimiento entre todas las regiones de Kol¢hhovia, durante los últimos años, se han eliminado gran parte de las restricciones al viaje dentro del país. A pesar de este levantamiento de restricciones, cada ciudadano sigue asignado, en buena lógica, a la región en la que ha nacido.

Para asegurarse de que el intercambio de ideas en la sociedad se mantiene dentro de los estándares de calidad y excelencia kol¢hhovitas, se nos pide que determinemos el mayor número de regiones con las que ha estado en contacto una persona en cada viaje.

Para ello tenemos a nuestra disposición, para cada viaje que ha realizado cada ciudadano, los datos de todos los kol¢hhovianos con los que ha estado en contacto.

Para cada viaje nos proporcionan un grafo en el que los vértices son los ciudadanos, dos vértices estarán unidos por una arista si pertenecen a regiones diferentes.

Para cada una de estas situaciones, nos piden que identifiquemos el problema conocido que podríamos aplicar y un paralelismo entre los elementos del problema y los elementos del problema encontrado por este estado que les ayuden a obtener el resultado esperado.

Solución:

a) En el primer caso nos piden encontrar el recubrimiento de vértices mínimo.

Si tenemos la información de todas las conversaciones que se han llevado a cabo podemos plantear el grafo de la siguiente forma:

- Los vértices serán los ciudadanos de Kol¢hhovia.
- Las aristas unirán unos ciudadanos con otros según las conversaciones en las que hayan participado.

Por lo tanto, en este grafo, el recubrimiento mínimo nos diría qué ciudadanos debemos convertir en *Agentes de seguridad* para garantizar que en todas las conversaciones participa, al menos, uno.

b) En este caso queremos encontrar el número cromático de G.

El grafo estará formado por un conjunto de subgrafos disjuntos, cada uno de ellos representando el viaje de un ciudadano. Los vértices serán los ciudadanos con los que ha tenido contacto y, dibujaremos una arista entre los vértices cuando estos pertenezcan a regiones diferentes. El número cromático determinará el número de regiones diferentes con las que ha estado en contacto cada Kol¢hhovita.

---

6. (Valoración de un 20 %) Cuestionario de evaluación Moodle

Dentro del aula de la asignatura, en el Campus Virtual, encontraréis la nueva herramienta (Moodle) en la parte derecha. En este Moodle hay un cuestionario con diversas preguntas que debéis resolver como último ejercicio de esta PEC.

Leed atentamente las siguientes instrucciones **antes de abrir el cuestionario**:

- Los contenidos que se evalúan en este cuestionario corresponden a los módulos 6 y 7 de la asignatura. Es importante que hayáis asimilado estos conocimientos **antes de abrir el cuestionario**.
- El cuestionario estará abierto durante el plazo de la PEC y lo podéis resolver cuando queráis. De todas formas, una vez lo abráis tendréis un **tiempo limitado** para resolverlo (40 minutos).

  Importante: El cuestionario quedará cerrado a las 23:59 de la fecha límite de entrega. Si empezáis a hacerlo después de las 23:19 del último día, ¡tendréis menos de cuarenta minutos para hacerlo!
- Las respuestas a las preguntas se tienen que introducir directamente en el cuestionario Moodle. No es necesario que las entreguéis junto con el resto de respuestas de la PEC.
- Las preguntas del cuestionario son aleatorias: cada estudiante recibirá un enunciado diferente.
- En algunas preguntas tendréis que introducir la respuesta en un formato específico (p. ej. con los valores ordenados de una determinada forma y sin espacios). Es muy importante **seguir fielmente el formato indicado** a la hora de introducir vuestra respuesta.
- Disponéis de **2 intentos** para resolver el cuestionario. El objetivo de tener dos intentos es poder solventar posibles problemas que hayáis tenido en la realización del cuestionario, ya sean problemas técnicos o bien que hayáis abierto el cuestionario por error. Por tanto, debéis tener en cuenta que:
  - La nota que obtendréis en el cuestionario será la de vuestro último intento.
  - Después del 1r intento, no recibiréis la calificación obtenida ni recibiréis feedback sobre vuestra propuesta de solución. Por lo tanto, no recomendamos usar el 2o intento para intentar mejorar nota, ya que puede ser que obtengáis una nota inferior.
  - Si usáis el 2o intento, el enunciado que encontraréis será diferente del del 1r intento.
  - Podéis realizar los dos intentos en días diferentes, siempre que sea dentro del plazo de la PEC. Dispondréis de 40 minutos para cada intento.
  - Cada vez que iniciéis el cuestionario contará como un intento, aunque no enviéis la respuesta. Por ejemplo, **si habéis hecho el 1r intento y volvéis a abrir el cuestionario, invalidaréis vuestro 1r intento y os quedaréis con la nota del 2o**.

Recursos

Recursos Básicos

- Módulo didáctico 6. Complejidad computacional.
- Módulo didáctico 7. Problemas intratables.
- Colección de problemas

Recursos Complementarios

- PECs y exámenes de semestres anteriores.
- Programario para el estudio de algoritmos sobre grafos.
- Enlaces: Applets interactivos sobre algoritmos de grafos.

Criterios de valoración

- La PEC se tiene que resolver **de forma individual**. En caso que hayáis consultado recursos externos, es necesario referenciarlos.
- Es necesario justificar la respuesta de cada apartado. Se valorará tanto el resultado final como la justificación dada.
- En los apartados donde sea necesario aplicar algún algoritmo, se valorará la elección del algoritmo apropiado, los pasos intermedios, el resultado final y las conclusiones que se deriven.

Formato y fecha de entrega

Hay que entregar **un único documento** PDF con las respuestas de todos los ejercicios. El nombre del fichero tiene que ser: **PEC3_Apellido1Apellido2Nombre.pdf**.

Este documento se tiene que entregar en el espacio **Entrega y Registro de EC** del aula **antes de las 23:59 del día 16/12/2021**. **No se aceptarán entregas fuera de plazo.**