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 % = 10 % + 10 %)

   Dados los siguientes problemas:

   1) FLOYD: Dado un grafo G, devuelve la matriz de distancias mínimas entre cada par de vértices.
   2) CLIQUE: Dado un grafo G, devuelve el orden t del subgrafo completo (K_t) más grande que podemos encontrar en G.
   3) PARTICION: Dado un conjunto S de enteros, devuelve SÍ si es posible dividir S en dos subconjuntos $S_1, S_2 \subset S$ tales que $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$ y

      $$\sum_{s_1 \in S_1} s_1 = \sum_{s_2 \in S_2} s_2.$$

      En caso contrario, devuelve NO.
   4) SUMA: Dado un conjunto S de n números, devuelve su suma.
   5) CONJUNTO_DOMINANTE: Dado un grafo G, devuelve el menor subconjunto S de vértices tal que, todo vértice $v \notin S$ será adyacente a un vértice de S.

   Responded a las siguientes preguntas:

   a) Para cada problema, indica su tipo (decisión, cálculo, optimización). Indica también la menor clase de complejidad a la que pertenece (en el caso de problemas de cálculo o de optimización, indica la clase de complejidad de su versión decisional). Justifica brevemente la respuesta.

   b) ¿Cuáles de las siguientes reducciones son posibles? (en el caso de problemas de cálculo o de optimización, considera su versión decisional). Justifica la respuesta. En el caso de que la última reducción sea posible, propón una función de reducción.

      i) FLOYD $\leq_p$ CLIQUE.
      ii) CONJUNTO_DOMINANTE $\leq_p$ FLOYD.
      iii) CONJUNTO_DOMINANTE $\leq_p$ CLIQUE.
      iv) SUMA $\leq_p$ FLOYD.

Solución:

a) 1) FLOYD: Se trata de un problema de optimización con complejidad polinomial $O(n^3)$ y, por lo tanto, pertenece a P.

2) CLIQUE: Se trata de un problema de optimización. Es un problema NP-Completo. Es preciso evaluar todas las combinaciones de nodos hasta encontrar el mayor t para el cual K_t es un subgrafo de G, lo que nos obliga a probar, en el peor de los casos 2^n conjuntos, siendo n el orden de G. Además, cada conjunto puede verificarse en tiempo polinómico.

3) PARTICION: Se trata de un problema de decisión NP-Completo. Existen, como en el caso anterior 2^n formas de escoger el primer subconjunto S_1, quedando el segundo subconjunto determinado de forma automática. Además, cada una de estas opciones puede verificarse en tiempo polinómico.

4) SUMA: Se trata de un problema de cálculo y su complejidad es $O(n)$ ya que debemos recorrer todo el conjunto para sumar sus elementos. Por lo tanto, pertenece a P.

5) CONJUNTO_DOMINANTE: Se trata de un problema de optimización (queremos el menor de los subconjuntos dominantes) y es NP-Completo. Al igual que PARTICION, requiere probar todos los subconjuntos de vértices para saber si son dominantes y tomar el menor de ellos. Además, cada uno de estos subconjuntos puede verificarse en tiempo polinómico.

b) i) Es posible. Puesto que FLOYD es un problema P, existe una reducción a un problema NP.

ii) No es posible salvo que $P = NP$. No podemos reducir un problema NP a un problema P.

iii) Es posible. Por ser ambos problemas NP-Completos, pueden reducirse uno al otro.

iv) Es posible. Una reducción podría ser la creación de un grafo trayecto en el que los pesos de cada arista sean cada uno de los elementos del conjunto a sumar. La mayor de las distancias que devuelva Floyd será la suma de todos los elementos.

---

2. (Valoración de un $15\,\% = 5\,\% + 10\,\%$)

Considera las siguientes expresiones lógicas:

1) $(a \vee b) \wedge c$,
2) $(a \wedge b) \vee (c \wedge \bar{a})$,
3) $(\bar{a} \wedge (b \vee \bar{c})) \wedge (a \vee b)$.

a) Transforma a FNC aquellas que no estén en FNC.

b) Rellena la siguiente tabla de verdad. ¿Existe algún conjunto de valores que haga ciertas todas las expresiones? ¿Y alguno que las haga todas falsas?

a	b	c	1)	2)	3)
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Solución:

a) Transformación a FNC:

1) $(a \vee b) \wedge c$ ya está en FNC.
2) A continuación, se detalla la conversión paso a paso:
   * Aplicamos la propiedad distributiva: $(a \vee c) \wedge (a \vee \bar{a}) \wedge (b \vee c) \wedge (b \vee \bar{a})$.
   * Eliminamos $a \vee \bar{a}$ ya que siempre es cierto: $(a \vee c) \wedge (b \vee c) \wedge (b \vee \bar{a})$.
   * Eliminamos $b \vee c$, ya que no afecta al resultado. Si b y c son ciertos, la expresión es cierta y, si ambos son falsos, siempre será falsa independientemente de que este término pertenezca o no a la expresión. Finalmente, obtenemos

$$(\bar{a} \vee b) \wedge (a \vee c).$$

3) Esta expresión está en FNC aunque puede simplificarse:
   * Cambiamos el orden de los términos: $\bar{a} \wedge (a \vee b) \wedge (b \vee \bar{c})$.
   * Podemos simplificar $\bar{a} \wedge (a \vee b)$ por $\bar{a} \wedge b$, ya que a no puede ser cierto y falso al mismo tiempo: $\bar{a} \wedge b \wedge (b \vee \bar{c})$.
   * Como en el apartado anterior, si b es cierto, el tercer término será cierto independientemente del valor de $\bar{c}$, y si b es falso, la expresión será falsa también de forma independiente al valor de $\bar{c}$. Por lo tanto, podemos eliminar el tercer término y obtenemos

$$\bar{a} \wedge b.$$

b) Tabla de verdad:

a	b	c	1)	2)	3)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	1	0
1	1	1	1	1	0

De la tabla de verdad podemos comprobar que, por ejemplo, $a = 0$, $b = 0$, $c = 0$ hace falsas todas las expresiones, mientras que $a = 0$, $b = 1$, $c = 1$ las hace ciertas todas.

---

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

   Una entidad bancaria que ha sufrido diferentes procesos de restructuración e integración con otras entidades financieras, ha incluido nuevas plataformas de pago e interacción con sus usuarios y está en proceso de implantación de nuevas estrategias de clasificación de clientes mediante técnicas de aprendizaje automático.

   Cada día se llevan a cabo miles de procesos automatizados para actualizar los riesgos de los clientes, asegurar que todas las transacciones han terminado correctamente, decidir el efectivo a llevar a los cajeros, etc.

   En el banco detectan que cada vez más procesos no llegan a finalizarse nunca, siendo preciso que los responsables del sistema los reinicien o los ejecuten paso a paso, o que se realicen procesos manuales para que terminen correctamente.

   - Se encarga a la empresa Grafos_y_Complejidad (GyC) que realice un análisis de procesos. El conjunto de todos los procesos a analizar es $P = \{p_1, p_2, ..., p_n\}$ y se construye un mapa de dependencias donde el proceso p_j estará conectado con p_k si p_j depende de p_k.
   - Inicialmente el equipo de GyC comienza identificando los ciclos de orden 3. Se determina que uno de los objetivos principales del análisis consistirá en identificar ciclos de cualquier longitud dentro del mapa de procesos.
   - En alguna reunión se hace hincapié en que, si se dibuja el diagrama de un proceso $p \in P$ y los procesos de los que depende, en un caso ideal, se trataría de un árbol.
   - Como recomendación general a la entidad bancaria se recomienda introducir mecanismos de trazabilidad en los procesos (la información de trazabilidad nos dice cuándo empieza y termina un proceso, cuál fue el proceso anterior y cuál es el proceso siguiente que se ejecutará). Puesto que el número de procesos es extremadamente alto y el tiempo de

programación escaso, se opta por asegurar que cada proceso o bien incluye información de trazabilidad o la incluye alguno de sus vecinos.

Contesta razonadamente a las siguientes preguntas, teniendo en cuenta que un inventario inicial ha identificado 25,000 procesos y 20,000 dependencias:

a) Indica si se trata de un grafo dirigido o no dirigido, e identifica cuáles son los vértices y aristas. ¿Qué quiere decir en términos del grafo que existan más procesos que dependencias?

b) ¿Por qué se quieren identificar todos los ciclos? ¿Qué representa un ciclo de longitud n?

c) ¿Por qué en un caso ideal las dependencias de los procesos tienen forma de árbol?

d) Indica qué problema, de los descritos en los materiales, permite identificar los procesos en los que se deben implantar mecanismos de trazabilidad. Indica el número máximo de opciones que tendría que verificar un algoritmo para encontrar la solución (o soluciones) óptimas a nuestro problema. Si este algoritmo es capaz de verificar 1000 opciones por segundo, ¿cuánto tiempo tardaría en obtener todas las soluciones?

Solución:

a) Se trata de un grafo dirigido donde los vértices serán los procesos y las aristas las relaciones de dependencia. En términos del grafo, que existan más procesos que dependencias nos indica que no se trata de un grafo conexo y que existirán procesos que no dependen de ningún otro.

b) Un ciclo representa un problema crítico ya que forma una relación de dependencia cerrada entre varios procesos. En la práctica, un ciclo de longitud 3, significaría que el proceso p_1 depende de p_2 que depende de p_3 que a su vez vuelve a depender de p_1 por lo que nunca termina. Un ciclo de cualquier longitud supone el mismo bloqueo pero tardando más en producirse.

c) Puesto que un árbol es un grafo acíclico, que las dependencias tengan forma de árbol significa que los procesos no tienen dependencias cíclicas entre sí.

d) Un problema de los descritos en los materiales que soluciona el problema planteado es VERTEX_COVER. El problema VERTEX_COVER necesita comprobar todos los conjuntos posibles para encontrar el de menor cardinalidad. En el peor de los casos, necesita realizar 2^n validaciones (siendo n el número de vértices) y, teniendo en cuenta que puede realizar 1000 por segundo, esto supone $\frac{2^{25000}}{1000}$ segundos.

---

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

Indica (y justifica) si las siguientes afirmaciones son ciertas o falsas. Cada apartado es independiente.

a) Si $A \in NP$ y B es NP-Completo, entonces $A \leq_p B$ y $B \leq_p A$.

b) Sean A, B y C tres problemas. Si $A \in P$, y sabemos que $C \leq_p B$ y $B \leq_p A$, entonces $C \in P$.

c) Si A es un problema cuyas soluciones pueden validarse en tiempo polinomial, entonces $A \leq_p 3SAT$.

Solución:

a) Falso, salvo que A también sea NP-Completo (dato que no nos viene indicado en el enunciado). Podemos asegurar la primera afirmación ($A \leq_p B$) por la definición de problema NP-Completo, pero no la segunda. De hecho, si se cumpliera podríamos asegurar que A también es NP-Completo ya que es NP y un problema NP-Completo se puede reducir a él.

b) Cierto. Si $C \leq_p B$ y $B \leq_p A$, entonces por la propiedad transitiva de la reducción polinómica, podemos afirmar que $C \leq_p A$. Como $A \in P$, tenemos que $C \in P$.

c) Cierto. El problema A pertenece a NP y podrá reducirse a $3SAT$ por ser $3SAT$ NP-Completo.

---

5. (Valoración de un 30 % = 15 % + 15 %)

El problema KNAPSACK, o problema de la mochila, consiste en optimizar la solución de un problema teniendo en cuenta una restricción de coste.

Se llama problema de la mochila ya que su descripción original establecía la necesidad de maximizar el valor de un conjunto de objetos a transportar en una mochila.

En la descripción original se quiere maximizar el valor económico de los objetos transportados (el valor) sin exceder la capacidad de la mochila (el coste).

Existen otras definiciones del problema de la mochila en las que el valor puede referirse, por ejemplo, a la cantidad de información que va en un fichero y el coste a su espacio de almacenamiento en disco.

La forma más habitual de plantear el problema de la mochila es la conocida como $0-1$ **Knapsack** que limita la cantidad de copias de cada objeto que podemos usar a una o ninguna.

a) **Algoritmo Voraz**

Aunque el problema de optimización asociado a KNAPSACK es NP-Difícil, se puede escribir un algoritmo voraz que devuelve una solución aproximada.

Dado un conjunto de objetos, cada uno con un coste y un valor, podemos calcular su beneficio como

$$beneficio = \frac{valor}{coste}.$$

Sea C una secuencia de objetos ordenados de forma decreciente por su beneficio. Asociamos cada objeto i de C con el campo COSTE que contiene el coste del objeto. Computacionalmente, podemos acceder al coste del objeto utilizando la instrucción i.COSTE que se considera 1 operación computacional. Sea c la capacidad. El siguiente podría ser un algoritmo voraz para obtener una solución aproximada al problema:

```
(1)  función voraz(C, c)
(2)     inicio
(3)        ocupacion←0
(4)        objetos← {}
(5)        para i∈C
(6)              si ocupacion + i.COSTE ≤ c
(7)                 entonces
(8)                          ocupacion←ocupacion + i.COSTE
(9)                          objetos←objetos ∪ i
(10)             finsi
(11)       finpara
(12)       retorno objetos
(13)    fin
```

1) Indica la complejidad de este algoritmo voraz.
2) Encuentra una instancia del problema (capacidad de la mochila y lista de objetos con coste, valor y beneficio) para la que el algoritmo voraz proporcione un resultado no óptimo.
3) A raíz de tu ejemplo, ¿dirías que el algoritmo funciona mejor cuando la capacidad es mucho mayor que el coste medio de los objetos? ¿por qué?

b) **Instancias de Knapsack**

Indica, de los siguientes problemas, cuáles pueden ser una instancia de KNAPSACK. Si lo son indica también qué representaría el coste, el valor y la capacidad.

1) Cobertura de red móvil: Dado un presupuesto máximo para antenas de telefonía móvil y sabiendo el límite de usuarios al que puede dar servicio cada tipo de antena, maximizar la cobertura de red móvil. **Nota:** No tengáis en cuenta la problemática de la ubicación de las antenas, sólo su capacidad de servicio.
2) Emergencias: Dada una red de carreteras, tenemos un grafo asociado en el que cada ciudad es un nodo y las aristas representan las conexiones por carretera. Queremos asegurar que cada ciudad o una de sus vecinas, tiene un centro de atención de emergencias.

3) Selección: Un repartidor quiere maximizar el beneficio de su ruta de reparto minimizando su coste de transporte. Para preparar su ruta dispone de una lista de ciudades y el coste de viajar entre cada una de ellas así como el beneficio esperado en cada ciudad.

Solución:

a) **Algoritmo Voraz**

1) El algoritmo tiene una complejidad $O(n)$, siendo n el número de objetos. Vemos que recorre todos los objetos de la lista una única vez ya que, según el enunciado, éstos ya venían ordenados por beneficio.
2) Por ejemplo, consideramos que la capacidad de la mochila es $c = 4$ y la secuencia C está formada por los siguientes elementos:
   - a: coste 3, valor 3,1 y beneficio $1,03$,
   - b: coste 2, valor 2 y beneficio 1,
   - c: coste 2, valor 2 y beneficio 1.

   El algoritmo voraz incluiría únicamente el objeto a, con un valor total de 3,1 y un coste de 3, cuando puede verse que la solución óptima consistiría en los objetos b y c con un valor total de 4 y un coste de 4.
3) El algoritmo voraz proporciona mejores resultados cuando c es mucho mayor que el coste medio, ya que el error total que podemos cometer es mayor cuando dejamos huecos grandes sin rellenar como en el ejemplo anterior.

b) **Instancias de Knapsack**

1) Cobertura: puede ser una instancia de KNAPSACK donde la capacidad es el presupuesto total, el valor es el número de clientes a los que da servicio una antena y el coste es el precio de instalar la antena.
2) Emergencias: no es una instancia de KNAPSACK sino de VERTEX_COVER.
3) Selección: No es una instancia de KNAPSACK sino una variación del TSP llamada TSP con beneficios. En este caso, puede parecer que tiene las características de un KNAPSACK pero, la diferencia más importante es que, cada vez que vamos a una ciudad varían todos nuestros costes de viaje por lo que no podemos tratarlo como un problema de mochila.

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**.
- 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 20/12/2018**. **No se aceptarán entregas fuera de término.**