1. Variants de 3SAT.
(i) Demostreu que la versió restringida de 3SAT que anomenem 3SAT-Twice at most i que tot seguit definim, és NP-complet: Donada una fórmula booleana F en forma normal conjuntiva, amb exactament 3 literals per clàusula i en la que cada literal apareix com a màxim 2 vegades, decidir si F és satisfactible

Para demostrar que 3SAT-Twice está en NP-Completo tenemos que demostrar que es un problema NP y NP-Hard.
Para demostrar que 3-SAT está en NP, solo tenemos que verificar que dada una instancia de 3SAT-Twice, es decir, un fórmula booleana F con sus propiedades y tenemos que verificar(si es F es satisfacible) en tiempo polinomial. Esto puede hacerse comprobando si la asignación dada de variables satisface la fórmula booleana.

Per demostrar que 3SAT-Twice está en NP-Hard.
Supongamos que en la instancia 3Sat, la variable aparece en k > 3 cláusulas. Entonces sustituye su primera aparición por x1, la segunda por x2, y así sucesivamente, sustituyendo cada una de sus k apariciones por una nueva variable diferente. Por último, añada la cláusula (-x1 v x2) (-x2 v x3) ... (-xk v x1).
Y repítelo para cada variable que aparezca más de tres veces. Es fácil ver que en la nueva fórmula ninguna variable aparece más de tres veces (y, de hecho, ningún literal aparece más de dos veces). Además, las cláusulas adicionales que incluyen x1,x2...xk obligan a estas variables a tener el mismo valor. Por lo tanto, la instancia original de 3SAT es satisfactoria si y sólo si la instancia 3SAT-Dos veces es satisfactoria.

(ii) En canvi, quan restringim les fórmules d'entrada de manera que cada literal aparegui com a màxim una vegada, el problema es pot resoldre en temps polinòmic. Formalment, definim 3SAT-once at most de la manera següent: Donada una fórmula booleana F en forma normal conjuntiva, amb exactament 3 literals per clàusula i en la que cada literal apareix com a màxim una vegada, decidir si F és satisfactible Demostreu que 3SAT-once at most $\in$ P

Sin pérdida de generalidad, podemos suponer que cada variable aparece exactamente una vez positivamente y exactamente una vez negativamente (si una variable sólo aparece una vez fija su valor para satisfacer la cláusula y elimina la cláusula). También podemos suponer que una variable no aparece en una cláusula más de una vez (si una variable aparece tanto positiva como negativamente en una cláusula, entonces la cláusula se satisface y puede eliminarse). Esto no alterará la satisfacción.

Utilicemos ahora la regla de resolución para eliminar las variables una a una (dado que cada variable aparece exactamente una vez positiva y una vez negativamente, se trata de un proceso determinista). Si en algún punto obtenemos la cláusula vacía el conjunto de cláusulas es insatisfactible, en caso contrario es satisfactible. Esto se debe a que

la resolución es un sistema de prueba proposicional completo (es decir, si una cláusula está implicada lógicamente por el conjunto de cláusulas, entonces es derivable del conjunto de cláusulas utilizando sólo la regla de resolución),

un conjunto de cláusulas es insatisfactible si implica lógicamente la cláusula vacía.

Este algoritmo tardará un tiempo polinómico, ya que cada variable se resuelve exactamente una vez. En particular, cada aplicación de la resolución reducirá el número total de cláusulas en uno, por lo que el número de cláusulas no aumenta. Por ejemplo, aplicando la resolución a $(x \vee B) \wedge (\overline{x} \vee C) \wedge ...)$
produce $(B \vee C) \wedge ...$

que tiene una cláusula menos que antes de la resolución. En cambio, si se aplicara esto a una fórmula 3SAT sin restricción en el número de apariciones de cada literal, aplicar la resolución podría hacer que el número de cláusulas aumentara exponencialmente.

2. Conjunt dominant. En un graf no dirigit G = (V, E), diem que D ⊆ V és un conjunt dominant en G si per cada vèrtex u ∈ V tenim que v ∈ D o és adjacent a un v ∈ D, (u, v) ∈ D. Definim el problema Conjunt Dominant de la manera següent: Donats un graf no dirigit G = (V, E) i un natural b decidir si existeix un un conjunt dominant D en G tal que |D| ≤ b. Demostreu que Conjunt Dominant és NP-complet.

El conjunto dominante es NP completo Si cualquier problema está en NP, entonces, dado un "certificado", que es una solución al problema y una instancia del problema , podremos verificar el certificado en tiempo polinómico

```
flag=true
for every vertex v in V:
  if v doesn't belong to Dominating Set:
     verify the set of edges
     corresponding to v
     if v is not adjacent
        to any of the vertices in DS,
        set flag=False and break
if (flag)
   solution is correct
else
   solution is incorrect
```

Realizaremos una reducción del problema del VC (vertex cover) a nuestro problema. Cada instancia del problema Vertex Cover consiste en un grafo G = (V, E) y un entero k consistente en el subconjunto de vértices como entrada puede convertirse en un problema Dominating Set consistente en el grafo G' = (V', E'). Construiremos el grafo G' de la siguiente manera:
- E' = Por cada arista E formada por vértices {u, v} en el grafo G, añadimos un nuevo vértice {uv} y lo unimos individualmente a los nuevos vértices u y v.
- V' = Añadir todos los vértices V del grafo original G.

-Supongamos que el grafo G tiene una cubierta de vértices VC de tamaño k. Cada arista de G tiene uno de los vértices pertenecientes a la cubierta de vértices. Por lo tanto, para cada arista e, formada por vértices {u, v} al menos u o v forma parte de la cubierta de vértices. Por lo tanto, si u está contenido en VC, entonces el vértice adyacente es v, también está cubierto por algunos de los elementos en VC. Ahora, para todos los nuevos vértices añadidos UV para cada borde, el vértice es adyacente tanto a u y v, uno de los cuales es al menos una parte de la CV, como se ha demostrado anteriormente. Por lo tanto, los vértices adicionales para todas las aristas también están cubiertos por este CV. El conjunto de vértices que forman la Cubierta de Vértices de tamaño k forman el Conjunto Dominante en el grafo G'. Por lo tanto, si G tiene una cubierta de vértices G' tiene un conjunto dominante del mismo tamaño.

Suponemos que el grafo G' tiene un Conjunto Dominante de tamaño k. Pueden darse dos posibilidades, que el vértice en el DS sea un vértice original o que pertenezca al nuevo vértice añadido UV para cada arista {u, v}. En el segundo caso, ya que cada nuevo vértice está conectado a los dos vértices de la arista, u y v, por lo tanto, puede ser

sustituido por cualquiera de u o v. Dado que, estos tres vértices forman un triángulo, por lo tanto, incluso mediante la sustitución de vista con cualquiera de u o v, podemos seguir para abarcar todos los vértices que se abarcó antes de reemplazar. Esto conducirá a la eliminación de todos los nuevos vértices añadidos, mientras que abarca todas las aristas del gráfico G '. Los nuevos vértices añadidos están dominados por el DS modificado y cubren todas las aristas de G con al menos u o v para cada arista UV. Por lo tanto, si G' tiene un conjunto dominante de tamaño k, G tendrá una cobertura de vértices de tamaño máximo k .

3. Una pila de pedretes. Considereu el joc següent. Tenim una pila de n pedretes i som dos jugadors. A cada moviment podem treure de la pila un cert nombre de pedretes. Cadascun de nosaltres tenim assignat un conjunt $S \subseteq \{1, ..., n\}$ i $T \subseteq \{1, ..., n\}$, respectivament, indicant el nombre de pedretes que podem treure en el nostre torn. Per exemple, si el meu conjunt és $S = \{1, 2, 3\}$ i el teu conjunt és $T = \{2, 4, 6\}$, en el meu torn només puc treure 1, 2 o 3 pedretes de la pila i en el teu torn tu pots treure 2 o 4 o 6 pedretes de la pila. No podem treure de la pila més pedretes de les que hi ha i qui primer buidi la pila guanya. Si la pila té menys pedretes de les que pot treure el jugador a qui li toca jugar, és a dir no té cap jugada possible, aleshores considerem empat. Per exemple si $n = 8$ i és el meu torn, puc guanyar treient 3 pedretes. Queden 5 pedretes: tu en pots treure 2 o 4, deixant 3 o 1 pedretes a la pila, respectivament. I en cada cas jo puc buidar la pila. Definim el el problema BuidarPila: Donats n, $S \subseteq \{1, ..., n\}$ i $T \subseteq \{1, ..., n\}$, decidir si el jugador amb S que inicia el joc té una estratègia guanyadora (pot guanyar el joc).

El problema BuidarPila pertenece a la clase de problemas computacionales que est á n en la clase de complejidad PSPACE-completo. Esto significa que el problema es al menos tan dif í cil como cualquier otro problema en PSPACE, y tambi é n es posible reducir cualquier problema en PSPACE a BuidarPila.

Para ver por qu é es as í , podemos utilizar el hecho de que BuidarPila puede modelarse como un juego de dos jugadores en un grafo dirigido, donde cada nodo representa un estado del juego (es decir, el n ú mero de piedras en el mont ó n), y cada arista representa un posible movimiento de uno de los jugadores. Los jugadores se turnan para hacer movimientos hasta que el juego termina (ya sea porque el mont ó n est á vac í o o porque ninguno de los jugadores puede hacer un movimiento).

Para determinar si el jugador inicial tiene una estrategia ganadora, podemos utilizar un algoritmo recursivo que compruebe todos los movimientos posibles a partir del estado actual y determine si alguno de ellos conduce a una posici ó n ganadora. Este algoritmo necesita llevar un registro del estado actual de la partida y del jugador actual, y necesita explorar todo el á rbol de juego para determinar el resultado de la partida.

Dado que el n ú mero de estados y movimientos posibles en el juego puede ser exponencial en el tama ñ o de la entrada (es decir, el n ú mero de piedras en el mont ó n), el problema est á en PSPACE. Adem á s, es dif í cil en PSPACE porque cualquier problema en PSPACE puede reducirse a BuidarPila codific á ndolo como un juego en un grafo dirigido.

Por lo tanto, BuidarPila es un problema computacionalmente dif í cil

4. Random 3SAT. Demostreu que hi ha un algorisme aleatori amb un temps esperat polinòmic tal que, donada una fórmula en Forma Normal Conjuntiva amb 3 literals per clàusula calcula una assignació que satisfà com a mínim 7/8 del nombre total de les clàusules.

Theorem 2. *For the MAX-3SAT problem, there exists a randomized algorithm whose expected number of satisfied clauses is at least* $7/8 \cdot OPT$.

Proof. The algorithm is simple: set each variable x_i to be true with probability $1/2$ independently. Now let Y_j be a random variable such that Y_j is 1 if the clause C_j is satisfied and 0 otherwise. Then we have

$$\Pr(Y_j = 0) = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

because all 3 variables involved in C_j must fail simultaneously for C_j to not be satisfied. Thus, we also have

$$\mathbb{E}[Y_j] = \Pr(Y_j = 1) = 1 - \frac{1}{8} = \frac{7}{8}.$$

Now let N denote the number of clauses satisfied by our algorithm, so

$$N = \sum_{j=1}^{m} Y_j.$$

Taking the expectation of both sides and applying linearity of expectation yields the following:

$$\mathbb{E}[N] = \mathbb{E}\left[\sum_{j=1}^{m} Y_j\right] = \sum_{j=1}^{m} \mathbb{E}[Y_j] = \sum_{j=1}^{m} \frac{7}{8} = \frac{7m}{8} \geq \frac{7}{8}\text{OPT},$$

as desired. Note that the Y_j are not necessarily independent, but as mentioned above, linearity of expectation still holds. □

It is remarkable that no better approximation algorithm exists for MAX-3SAT unless P = NP.

5. The Contraction Algorithm Un cut-set d'un graf no dirigit G = (V, E) és un subconjunt d'arestes C ⊆ E tals que si les esborrem d' E el graf resultant (V, E − C) conté 2 o més components connexes. Un global min-cut o min-cut (depèn de les fonts bibliogràfiques) és un cut-set de cardinalitat mínima. Fixeu-vos que la cardinalitat d'un min-cut d'un graf G és el mínim nombre d'arestes que cal esborrar per a desconnectar G. Presenteu the Contraction Algorithm conegut també per Karger's Algorithm i analitzeu-ne el temps de computació i la probabilitat d'error.

El algoritmo básico de Karger contrae iterativamente aristas elegidas al azar hasta que sólo quedan dos nodos; esos nodos representan un corte en el grafo original. Al iterar este algoritmo básico un número suficiente de veces, se puede encontrar un corte mínimo con alta probabilidad.

Análisis de complejidad :

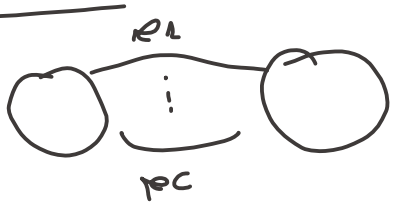
Complejidad temporal : O(EV^2), donde E es el número de aristas y V es el número de vértices del grafo. No se garantiza que el algoritmo encuentre siempre el corte mínimo, pero tiene una alta probabilidad de hacerlo con un gran número de iteraciones. La complejidad temporal está dominada por el bucle while, que se ejecuta durante V-2 iteraciones y realiza una cantidad constante de trabajo en cada iteración. El trabajo en cada iteración incluye encontrar el subconjunto de un elemento, la unión de subconjuntos y contraer la arista. Por lo tanto, la complejidad temporal es O(EV^2) donde V es el número de vértices y E es el número de aristas.

Espacio auxiliar : El espacio auxiliar del código anterior es O(E + V), donde E es el número de aristas del grafo y V es el número de vértices. Esto se debe a que el

programa utiliza una lista de adyacencia para almacenar el grafo, que requiere O(E) de espacio. También utiliza dos listas para almacenar los vértices visitados y la cola BFS, que requieren O(V) de espacio. En total, el programa utiliza O(E + V) de espacio.

```
1)  Initialize contracted graph CG as copy of original graph
2)  While there are more than 2 vertices.
      a) Pick a random edge (u, v) in the contracted graph.
      b) Merge (or contract) u and v into a single vertex (update
         the contracted graph).
      c) Remove self-loops
3) Return cut represented by two vertices.
```

Probabilidad $\geq 1/n^2$



S_1 = una de las aristas es escogida en la iteración 1

S_2 = ..

La probabilidad del min-cut es si ninguno de los eventos S_1 ... pasa.

Entonces la probabilidad es $P[S_1' \cap S_2' \cap S_3' \cap \dots]$

$\geq \frac{1}{n^2}$

5.Random 3SAT. Demostreu que hi ha un algorisme aleatori amb un temps esperat polinòmic tal que, donada una fórmula en Forma Normal Conjuntiva amb 3 literals per clàusula calcula una assignació que satisfà com a mínim 7/8 del nombre total de les clàusules

Solucio: Per resoldre aquest problema, podem utilitzar un algorisme de cerca local anomenat WalkSAT, que és un algorisme aleatori que funciona bé en molts casos pràctics.

L'idea darrere de l'algorisme WalkSAT és començar amb una assignació aleatòria i iterativament millorar-la per maximitzar el nombre de clàusules que satisfan. En cada iteració, triem una clàusula que no estigui satisfeta i canviem el valor d'un dels literals en aquella clàusula per intentar satisfer-la. Aquest canvi es fa aleatòriament amb una probabilitat ε, i en cas contrari es tria el literal que maximitzi el nombre de clàusules satisfetes. Això s'anomena una "heurística de la disjuntiva".

L'algorisme WalkSAT es para quan s'ha trobat una assignació que satisfà un nombre suficientment gran de clàusules (per exemple, 7/8 del total) o quan s'ha arribat a un nombre màxim d'iteracions.

L'esperança matemàtica del nombre d'iteracions que requereix l'algorisme per trobar una assignació que satisfaci un nombre suficientment gran de clàusules és polinòmica en el nombre de variables i clàusules de la fórmula en 3SAT.

Per tant, l'algorisme WalkSAT és un algorisme aleatori amb un temps esperat polinòmic que satisfà el requeriment establert en la pregunta.

6. El sistema criptogràfic RSA. Diem que el sistema RSA és fàcilment vulnerable quan donada la clau pública i un missatge codificat, aquest es pot decodificar en temps polinòmic.

(i) Demostreu que si P = NP, aleshores els sistema RSA seria fàcilment vulnerable.
(ii) I si tinguéssim manera de vulnerar fàcilment el sistema RSA, això implicaria que P = NP?

Solucio:
(i) Si P = NP, això implica que tot problema NP es pot resoldre en temps polinòmic. En particular, el problema de factorització, que és el problema clau en el sistema RSA. Si fos possible factoritzar ràpidament nombres enters grans en temps polinòmic, això significaria que podríem obtenir la clau privada del sistema RSA a partir de la clau pública. Per tant, si P = NP, els sistema RSA seria fàcilment vulnerable.

(ii) No, si tinguéssim una manera de vulnerar fàcilment el sistema RSA, això no implicaria directament que P = NP. El problema de factorització és només un problema en NP, no en NP complet. Per tant, el fet que puguem resoldre el problema de factorització ràpidament no significa necessàriament que puguem resoldre tots els problemes NP ràpidament. Així, encara que el sistema RSA fos fàcilment vulnerable, no podríem concloure directament que P = NP.

7. Espiant RSA. Suposeu que en el sistema RSA l'espia Eve aconsegueix (N, d), la clau privada d'Alice. La clau publica d'Alice és (N, e) amb e = 3. Demostreu que per aquesta clau pública, l'espia Eve pot calcular eficientment la factorització de N.

Solucio:
Calcula k = d*e - 1. Este valor es múltiplo de φ(N) porque d*e ≡ 1 mod φ(N) según el teorema de Euler.
Calcula x = 2^k mod N utilizando la función de exponenciación modular eficiente.
Calcula el máximo común divisor entre x-1 y N, es decir, p si x-1 es múltiplo de p, o q si x-1 es múltiplo de q.

La otra factorización será N/p o N/q, dependiendo del valor obtenido en el paso 5. Además, el valor k es múltiplo de φ(N) debido al teorema de Euler, lo que hace que la exponenciación modular de x sea igual a 2 mod N. Luego, el valor x-1 será múltiplo de p o q, y podemos encontrar la factorización de N calculando el mcd de x-1 y N.

Sea ⟨N, e⟩ una clave pública RSA. Dada la clave privada d, se puede factorizar eficientemente el módulo N = pq.

Calcular k = de - 1. Por definición de d y e sabemos que k es múltiplo de φ(N). Como φ(N) es par, k = 2^t*r con r impar y t ≥ 1. Tenemos g^k = 1 para todo g ∈ ℤN×, y por tanto g^k/2 es una raíz cuadrada de la unidad módulo N. Por el Teorema del Resto Chino, 1 tiene cuatro raíces cuadradas módulo N = pq. Dos de estas raíces cuadradas son ±1. Las otras dos son ±x donde x satisface x = 1 mod p y x = -1 mod q. Usando cualquiera de estas dos últimas raíces cuadradas, la factorización de N se revela calculando gcd(x - 1, N). Un argumento sencillo muestra que si g se elige al azar de ℤN× entonces con probabilidad al menos 1/2 (sobre la elección de g) uno de los elementos de la secuencia gk/2, gk/4, ..., gk/2t mod N es una raíz cuadrada de la unidad que revela la factorización de N. Todos los elementos de la secuencia se pueden calcular eficientemente en tiempo O(n3) donde n = log2(N).

1.Inicializar k = de -1.
2. Escoger un g random de {2,...,N-1} y t = k;
3. Si t es divisible por 2, t = t/2 y x = g^t mod N. else ir al paso 2)
4. Si x >1 & y = gcd(x-1,N) > 1 entonces p = y & q = N/y.

8. No incentiu de canvi. Recordem els jocs de creació de xarxes (NCG) introduïts per Fabrikant et al. Un joc Γ es defineix per un parell Γ = hV, αi on V = {1, . . . , n} és el conjunt de jugadors (o nodes de la xarxa) i α el cost d'establir un enllaç. Cada node u ∈ V pot establir enllaços a qualsevol dels altres nodes. Una estratègia del jugador u és un subconjunt su ⊆ V − {u} indicant els enllaços que u ha comprat. Un vector d'estratègies per Γ és una tupla s = (s1, . . . , sn) on per cada u ∈ V, su és l'estratègia del jugador u. A cada vector d'estratègia s li correspon un outcome graph, un graf no dirigit definit per G[s] = (V, E) amb E = {(u, v)|u ∈ sv ∨ v ∈ su}). El cost d'un jugador u depèn de les estratègies de tots els jugadors i es defineix de la manera següent: cu(s) = α|su| + P v∈V dG[s] (u, v). Considerem ara el problema EstaBé: Donats Γ = hV, αi, un vector d'estratègia s = (s1, . . . , sn), un jugador u i valor k decidir si el jugador u estaria content amb un cost k. Formalment, decidir que no hi ha cap estratègia s 0 u tal que cu(s−u, s0 u) < k.
Demostreu que el problema EstaBé és co-NP-complet.

Para demostrar que el problema EstaBé es co-NP-completo, necesitamos demostrar que est en co-NP y que es co-NP-duro.

En primer lugar, demostraremos que el problema de EstaBé es co-NP. Para ello, necesitam demostrar que dado un certificado de que el jugador u no está contento, podemos verificar eficientemente. Un certificado de que el jugador u no está contento es un vector de estrategias s' tal que cu(s-u, s') < cu(s). Podemos verificar este certificado calculando cu(s-u, y cu(s), y comprobando si cu(s-u, s') < cu(s). Esto puede hacerse en tiempo polinómico, ya q s ólo necesitamos calcular las distancias en los grafos G[s] y G[s-u, s'].

A continuación, demostramos que el problema EstaBé es co-NP-duro. Para ello, reducimo problema co-NP-completo Not-All-Equal 3SAT al problema EstaBé.

Dada una instancia del problema Not-All-Equal 3SAT con variables x1, ..., xn y cláusulas C1 Cm, construimos una instancia del problema EstaBé como sigue. Sea V = {x1, ..., xn, y1, ..., donde yi representa la cláusula i-ésima. Fijamos α = 1, y dejamos que cu(s) sea el número cláusulas que no son satisfechas por la asignación correspondiente al vector de estrategias

Para cada variable xi, añadimos los siguientes estratagemas a su conjunto de estrategias:

{xi, ¬ xi}
{xi, yi | Ci contiene xi}
{¬ xi, yi | Ci contiene ¬ xi}
Para cada cláusula yi, añadimos la siguiente estratagema a su conjunto de estrategias:

{xi | xi aparece en Ci} ∪ {¬ xi | ¬ xi aparece en Ci}
Puede demostrarse que el jugador xi está contento en el juego EstaBé si y sólo si existe un asignación satisfactoria para la instancia 3SAT No-Todo-Igual.

Por lo tanto, el problema EstaBé es co-NP-completo.