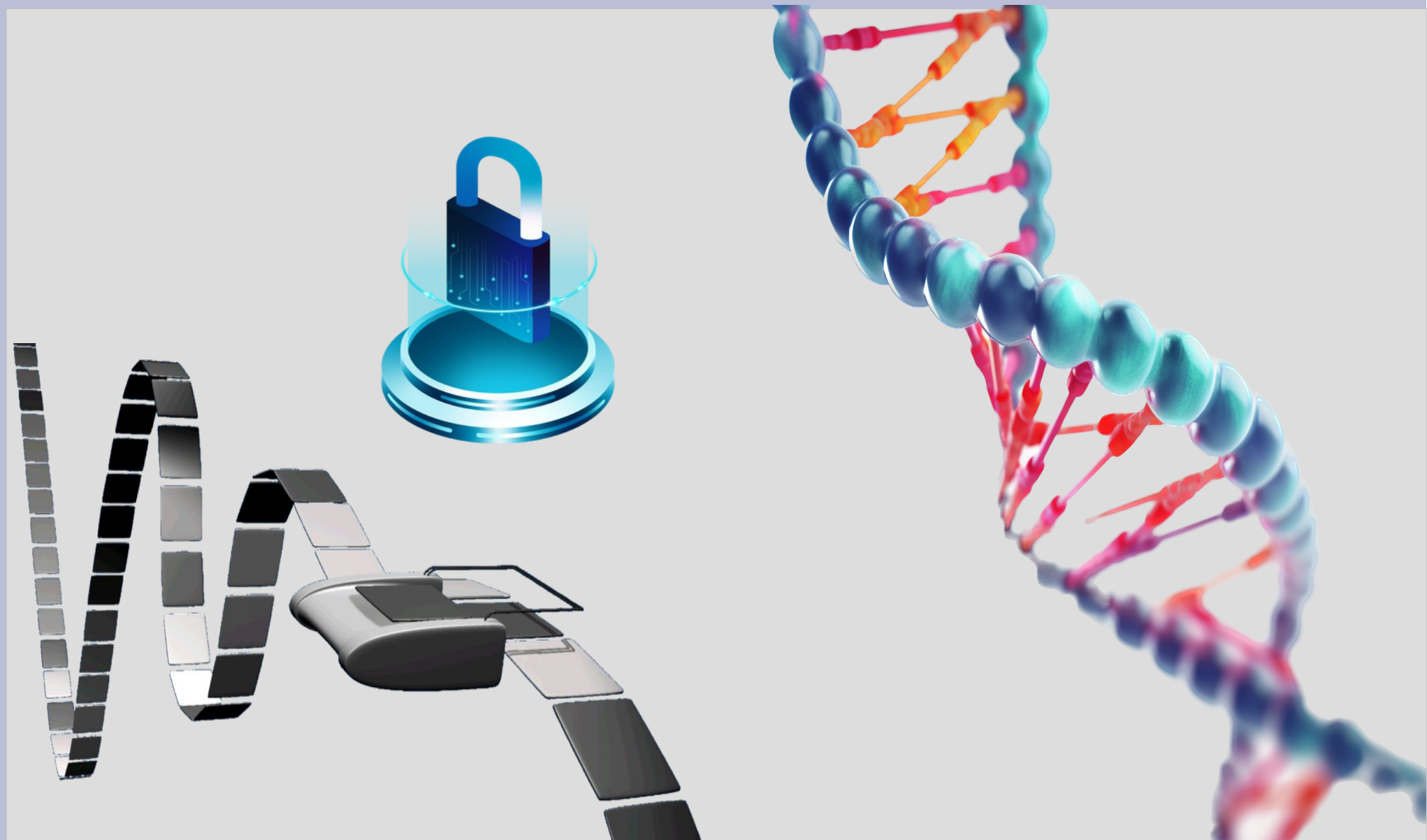
El problema P=NP.
Aplicaciones a la criptografía



Definiciones básicas

Definición alfabeto: Un alfabeto A es un conjunto finito de elementos que denominaremos símbolos o letras.
Ejemplos: A={0,1} es el alfabeto binario y B = {a, b, c, d, e, f, g, h, i, j, k, l, m, n, ñ, o, p, q, r, s, t, u, v, w, x, y, z} es el alfabeto castellano.

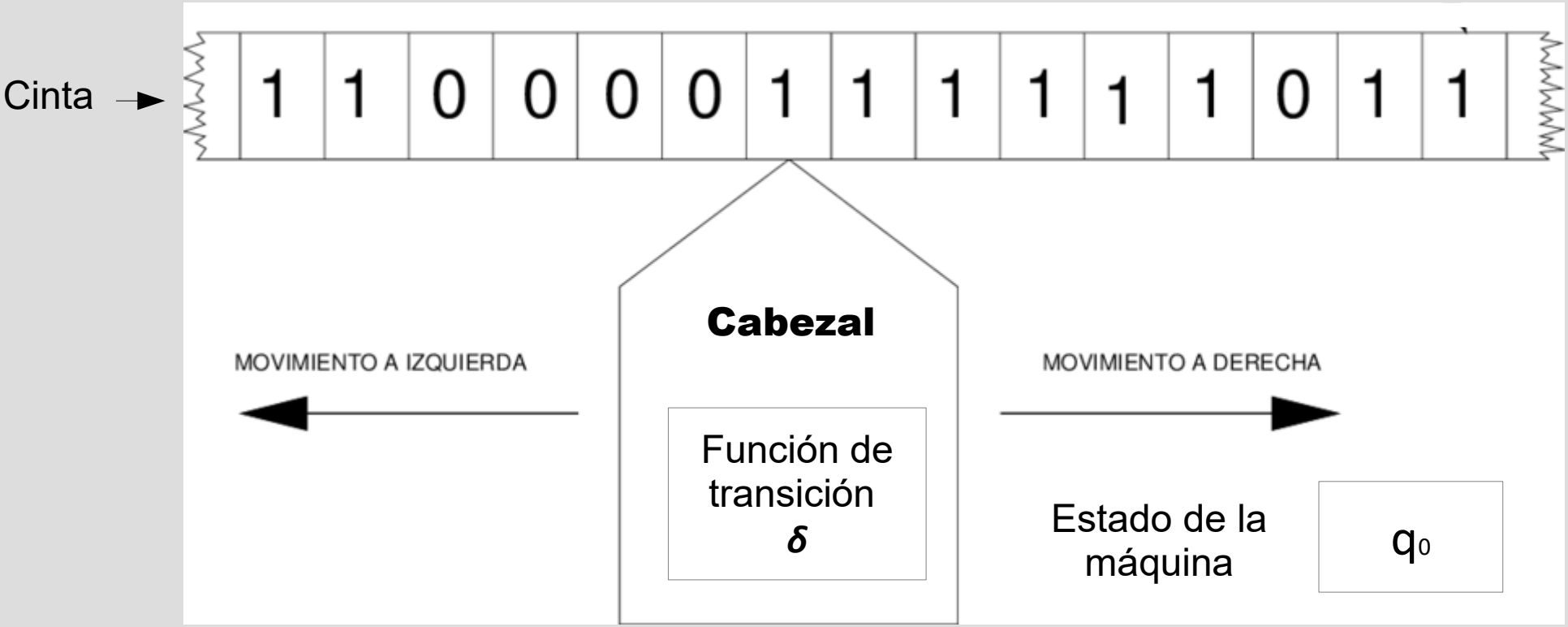
Definición palabra: Una palabra sobre un alfabeto A es una sucesión finita de elementos de dicho alfabeto. Una máquina de Turing recibirá como entrada una palabra. Una palabra será aceptada por una máquina de Turing si alcanza un estado terminal al recibirla como entrada.

Ejemplo: 01110 es una palabra en el alfabeto A={0,1}.

Definición lenguaje: Un lenguaje es un subconjunto del conjunto de todas palabras pertenecientes a un alfabeto. Un lenguaje se considera aceptado por una máquina de Turing si el alfabeto del lenguaje esta contenido en el alfabeto de entrada de la máquina de Turing y la máquina de Turing acepta todas las palabras de dicho lenguaje y ninguna otra.

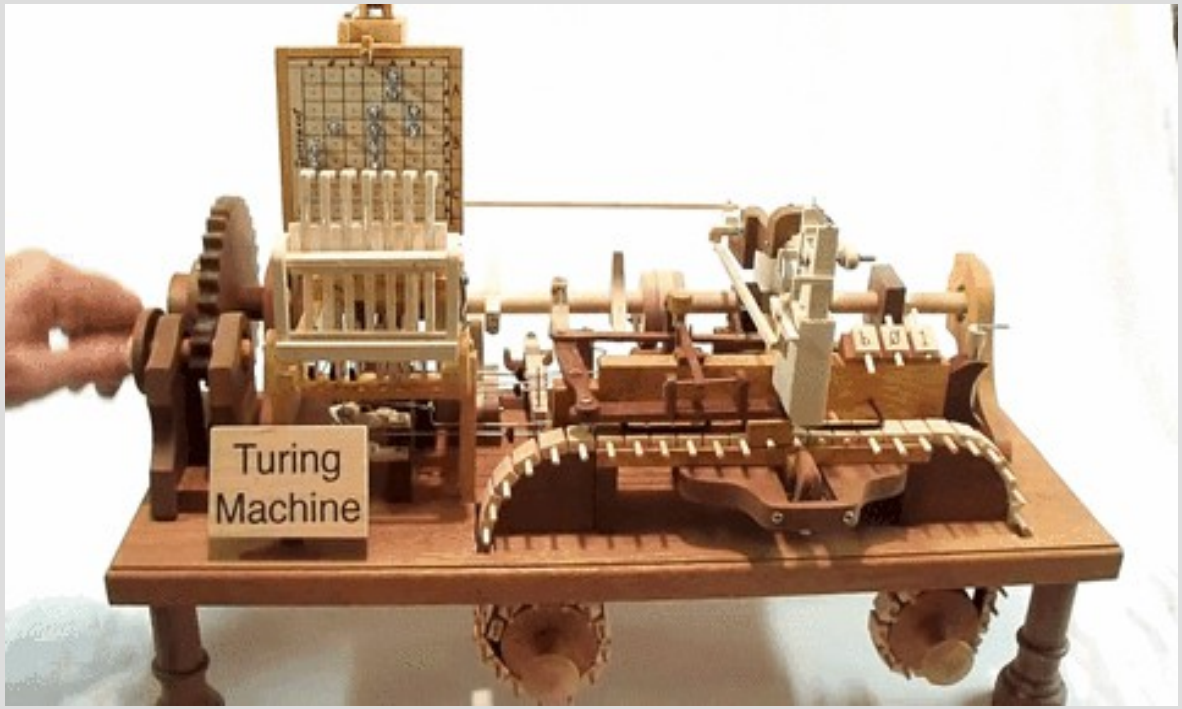
Máquina de Turing

Una máquina de Turing es una máquina compuesta por tres partes: una cinta infinita en que cada posición contiene un símbolo, un cabezal que puede leer un símbolo de una posición y reemplazarlo por otro y una lista de instrucciones, o función de transición, que debe seguir la maquina, que le indican a la maquina dado un determinado símbolo y el estado de la maquina como debe proceder.



Funcionamiento de una máquina de Turing

Una máquina de Turing recibirá como entrada una palabra que pertenezca a su alfabeto de entrada. Comenzará siempre con el cabezal en la primera posición de dicha entrada y en su estado q_0. A partir de ese punto seguirá las reglas marcadas por su función de transición cambiando símbolos y moviendo su cabezal hasta alcanzar un estado terminal.



Variaciones de la máquina de Turing

Definición máquina de Turing determinista: Una máquina de Turing determinista tiene una función de transición en la que a cada pareja de estado y símbolo le corresponde de forma unívoca otro estado, símbolo y movimiento del cabezal.

Definición máquina de Turing no determinista: Una máquina de Turing no determinista es una variación en que para cada pareja (q, b) $\in$ QxB, tendremos que la función de transición puede tener una lista finita de posibles transiciones que ejecutar. A la hora de ejecutar pasos de calculo se escogerá cualquiera de las posibles transiciones de δ y se procederá de igual manera que la máquina de Turing determinista. Se denomina que una máquina de Turing no determinista acepta una palabra cuando existe alguna sucesión de elecciones de transiciones tal que la maquina llega a un estado de aceptación al comenzar con dicha palabra.

Complejidad Algorítmica

Definición complejidad de una máquina de Turing: Una máquina de Turing se dice que es de complejidad f(n) si para toda entrada x tal que |x|=n (longitud de la entrada) la máquina se para en menos unidades que f(n). Estas unidades podrán ser pasos de calculo y entonces diremos que estamos midiendo la complejidad en el tiempo o podrán ser el numero de casillas en la cinta que la máquina ha usado y entonces diremos que estamos midiendo la complejidad en el espacio.

Definición complejidad de un lenguaje: La complejidad de un lenguaje se dirá f(n) si existe alguna máquina de Turing tal que acepta el lenguaje y es de complejidad f(n). De igual manera, se dirá que la complejidad de un problema es igual a la complejidad del lenguaje que contiene todos sus casos positivos.

Ejemplos P y NP

Clases de complejidad a estudiar:

- **Clase P:** lenguajes aceptados por máquinas deterministas en tiempo $O(n^j)$.
- **Clase NP:** lenguajes aceptados por máquinas no deterministas en tiempo $O(n^j)$.

Ejemplo problema P: Multiplicar dos números

Dado como entrada dos números a y b, podríamos multiplicarlos recorriendo el primero por cada dígito del segundo, teniendo una complejidad del orden de TIEMPO(n*n).

Ejemplo problema NP: Saber si un número es compuesto

Dado un número p, podríamos escoger un número aleatorio menor que p y probar a dividir p por él. Si para algún número se puede, entonces p es compuesto. Como dividir tiene una complejidad de O(n*n), este problema tendrá una complejidad del orden de NTIEMPO(n*n).

Problema NP $\Leftrightarrow$ Solución comprobable en tiempo polinomial

NP-completitud

Definición reducción de un lenguaje: Un lenguaje L1 se dice reducible a un lenguaje L2, denotado $L1 \propto L2$. Si existe una maquina de Turing que calcule en tiempo polinomial una función R del alfabeto de L1 al alfabeto de L2 y tal que $x \in L1 \Leftrightarrow R(x) \in L2$.

Definición lenguaje NP-completo: Un lenguaje NP es NP-completo si cualquier otro lenguaje NP se reduce a él. De igual manera un problema sera NP-completo si todo otro problema NP es reducible a él.

Problema P vs NP

¿Es la clase P igual a la clase NP?

$\Leftrightarrow$

¿Los problemas cuya solución se puede encontrar en tiempo polinomial y cuyas soluciones se pueden comprobar si son correctas en tiempo polinomial son los mismos?

Conjetura de Cook: La clase de complejidad P y la clase NP son distintas.

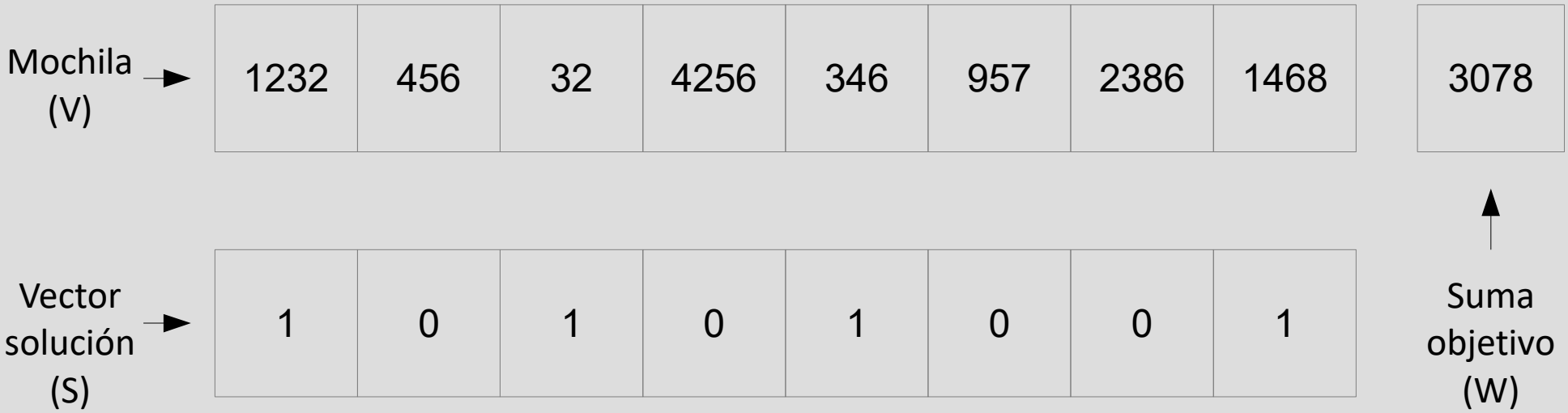
Criptografía

Criptografía: es la ciencia que estudia como proteger la información mediante transformaciones de los datos para que sean difíciles de entender por persona no autorizadas, pero que sean comprensibles por las personas autorizadas.

Criptografía asimétrica: consiste en el uso de dos claves, una clave pública que sirve para encriptar mensajes y que todo el mundo conoce y una clave privada para desencriptar que sólo conoce el receptor de los mensajes. Ejemplo: RSA (Rivest-Shamir-Adleman).

Criptosistemas basados en problemas NP-completos: se basan en construir casos particulares de problemas NP-completos que sean fácilmente resolubles y enmascararlos para que parezcan casos genéricos del problema. Son sistemas de criptografía asimétrica, donde la clave privada permitirá devolver el problema a su versión fácil de resolver.

Problema de la suma de subconjuntos



Problema de la suma de subconjuntos: El problema de la suma de subconjuntos es un problema de decisión en que dado un conjunto V de enteros positivos se debe determinar si existe alguna suma de elementos de V que den un valor W determinado.

Criptosistema de Merkle-Hellman

Ejemplo 3.2. Sea V = {64, 32, 16, 8, 4, 2, 1} y W = 77, tendremos que W = 1+4+8+64, que es equivalente a encontrar la representación en binario de 77, es decir, 1001101.

Problema de subconjuntos fácil: Sea V = {v_1, . . ., v_n}, tal que $v_i > \sum_{j=0}^{i-1} v_j$, ∀ i ≤ n, entonces dado W = x, tendremos que cualquier posible solución deberá incluir a v_n, equivalentemente $s_n = 1$, si y solo si $x \geq v_n$, puesto que sumando todos los demás valores de V nunca se llega a sumar hasta v_n. Siguiendo la misma lógica tendremos que $s_{n-1} = 1$ si y solo si $x - v_n s_n \geq v_{n-1}$ y, en general, $s_i = 1$ si y solo si $x - \sum_{j=i+1}^{n} v_j s_j \geq v_i$.

Criptosistema de Merkle-Hellman

Enmascarar el problema como difícil: Presupongamos que poseemos V y W una versión fácil del problema como descrita anteriormente, de la cual podemos hallar la solución S fácilmente, y elijamos dos números m y p tal que sean primos relativos y $m > \sum_{i=0}^{n} v_i$.
Podemos crear una nueva versión V' y W' del problema mediante la siguiente relación:

$$v_i' = p\, v_i \bmod m$$
$$W' = p\, W \bmod m$$

Tal que conocidos V', W', p y m se podrá deshacer el cambio de la siguiente manera:

$$v_i = p^{-1}\, v_i' \bmod m$$
$$W = p^{-1}\, W' \bmod m$$

Donde utilizamos que mcd(p,m) = 1 y que $m > \sum_{i=0}^{n} v_i$, tenemos que:

$$W = p^{-1} \sum v_i'\, s_i \bmod m = p^{-1} \sum p\, v_i s_i \bmod m = \sum v_i s_i \bmod m = v_i s_i$$

Criptosistema de Merkle-Hellman

Funcionamiento del criptosistema de Merkle-Hellman:

Supongamos tenemos V' obtenido a partir de m y p, entonces tendríamos que V' sería nuestra clave pública y m y p serían nuestra clave privada.

Supongamos alguien desea enviarnos un mensaje M con este criptosistema. Sea n la longitud de V', tal que M escrito en binario sea un vector de tamaño menor o igual a n. Entonces sea W' el producto escalar de M y V', será únicamente W el mensaje que se enviará.

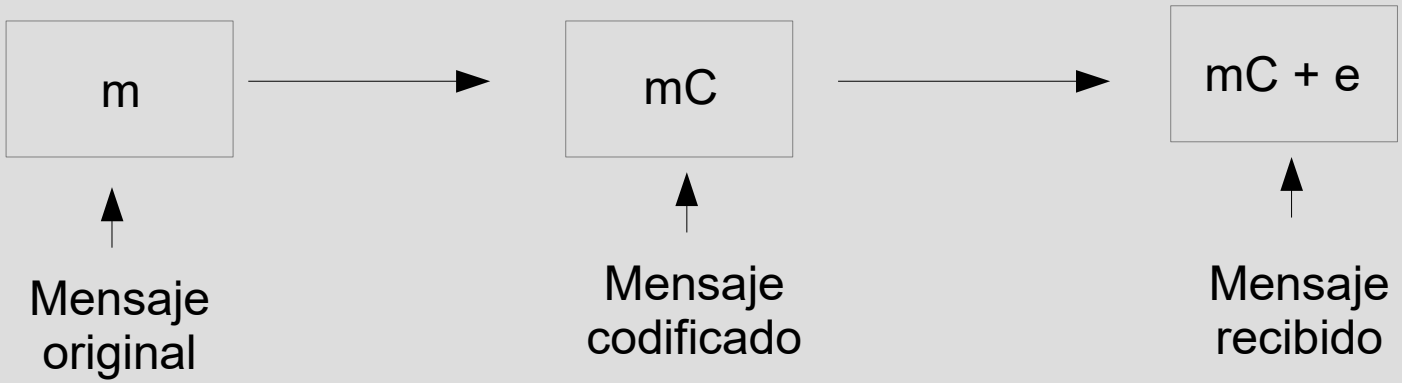
Si nosotros recibimos W', usaremos m y p para obtener la versión fácil del problema V y W. Resolveremos este problema y obtendremos la solución S que será igual a M.

Si cualquier otro recibe W', tendrá que resolver el problema de la suma de subconjuntos con V' y W' que es un problema NP-completo.

Problema de decodificación de un código lineal general

Definición distancia de Hamming: Dadas dos palabras $c_1...c_n$ y $d_1...d_n$ que tengan la misma longitud se denomina distancia de Hamming al numero de posiciones en que los símbolos c_i y d_i son distintos. El peso de Hamming de una palabra es la distancia de Hamming a la palabra de igual longitud que solo contiene el símbolo en blanco del alfabeto. Para números esto equivale a contar los dígitos distintos de 0.

Problema de decodificación de un código lineal general: Dada una matriz C binaria, un vector m' y un valor entero positivo t. Se denomina problema de decodificación de un código lineal general a determinar si existe algún vector e de peso de Hamming menor o igual a t, tal que m' = mC + e para algún vector m.



Teoría de códigos

Códigos correctores de errores: Consiste en introducir información redundante en los datos que se desean transmitir tal que el receptor tenga la capacidad de corregir un número determinado de errores.

Matrices Goppa binarias: son un tipo de códigos correctores de errores creados a partir de un polinomio con raíces simples sobre un cuerpo finito que poseen un algoritmo para corregir errores de forma eficiente.

Se utilizan las matrices Goppa principalmente porque fueran las usadas en la publicación original y porque no se ha roto la seguridad de este criptosistema con estas matrices.

Ejemplo matriz Goppa binaria

1	0	0	1	0	1	1
0	1	0	1	0	1	0
0	0	1	1	0	0	1
0	0	0	0	1	1	1

Criptosistema de McEliece

Enmascarar el problema como difícil: Dada una matriz Goppa C de dimensión n x k tal que sepamos corregir hasta t errores. Escogeremos P y S tal que P es una matriz de permutación k x k y S es una matriz aleatoria no singular n x n. Crearemos C' = SCP.

Veamos que podemos corregir errores con C', supongamos recibimos c = mC' + e, donde m es el mensaje original y e tiene peso de Hamming menor que t.

- Calculamos $cP^{-1} = mSC + eP^{-1}$
- Usamos que C es matriz Goppa para corregir eP^{-1} y obtener mS
- Invertimos S y obtenemos m

De esta forma hemos recuperado m, pero la matriz C' no aparenta ser un código corrector de errores.

Criptosistema de McEliece

Funcionamiento del criptosistema de McEliece:

Supongamos tenemos C' obtenido a partir de C matriz Goppa capaz de corregir hasta t errores, S y P como antes, entonces tendríamos que C' y t sería nuestra clave pública y S y P serían nuestra clave privada.

Supongamos alguien desea enviarnos un mensaje m con este criptosistema. Entonces calcularían mC' y le sumarían un vector e aleatorio de peso de Hamming t. Enviarían únicamente mC' + e.

Si nosotros recibimos mC' + e, usaremos S y P para obtener el mensaje original como ya hemos visto.

Si cualquier otro recibe mC' + e, tendrá que resolver el problema de decodificación de un código lineal general que es un problema NP-completo.

Algoritmos genéticos

Los algoritmos genéticos se centran en crear poblaciones en que cada individuo tenga una serie de genes y utilizar una función de adaptabilidad para determinar los individuos con los genes más aptos. De esta manera a mayor valor de dicha función mayor probabilidad de que dichos genes pasen a la siguiente generación. Podemos distinguir 5 procesos principales en un algoritmo genético:

- **Inicialización:** Crear la población inicial
- **Selección:** Elegir los progenitores
- **Reproducción:** Crear los nuevos individuos
- **Mutación:** Modificar genes aleatoriamente
- **Reemplazamiento:** Reemplazar individuos de la generación anterior por nuevos

Algoritmos meméticos

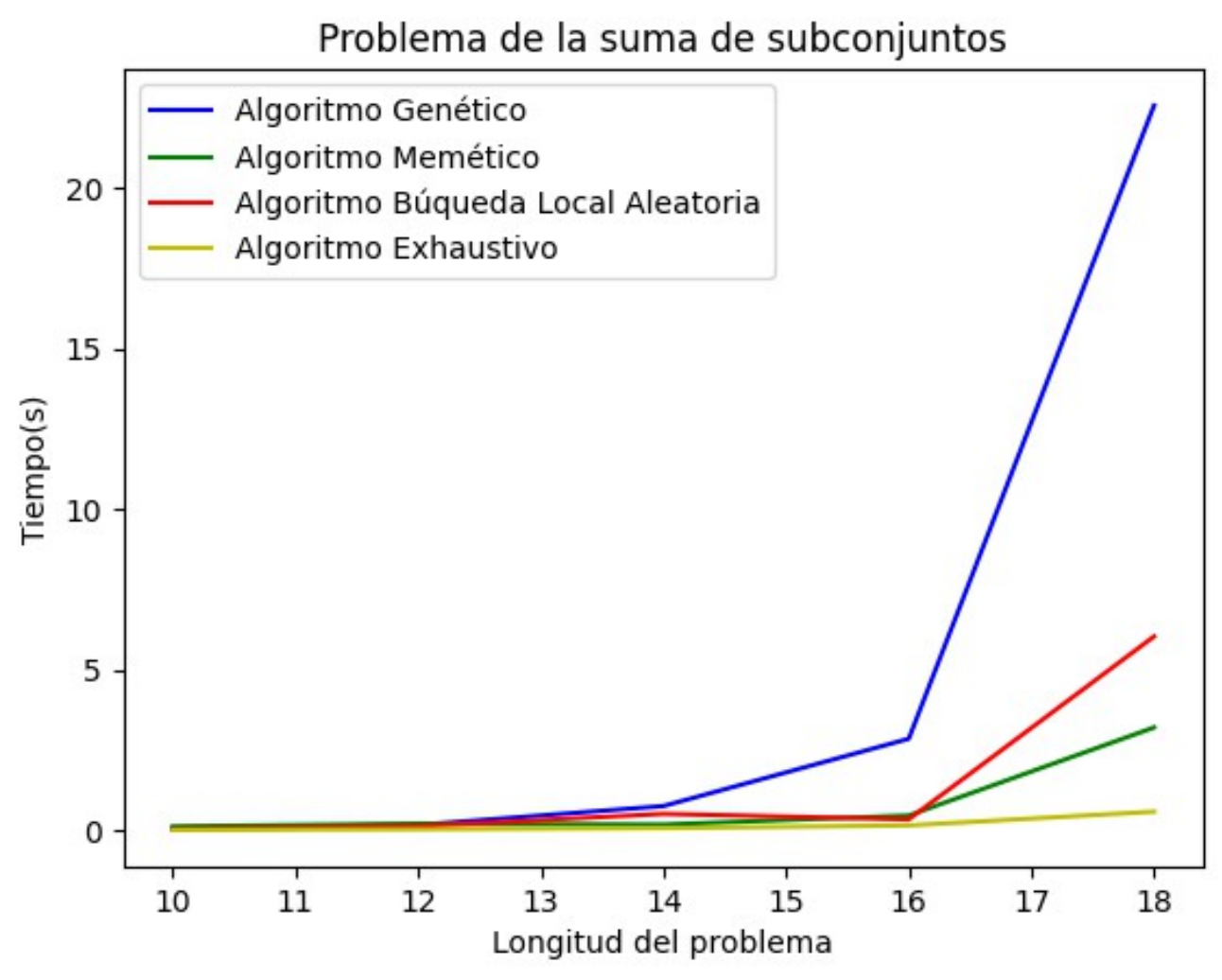
La diferencia de los algoritmos meméticos respecto a los genéticos es que a la población se le aplica un algoritmo de búsqueda local y el individuo con mejor adaptabilidad que encuentre dicho algoritmo es el que se añade a la población.

El algoritmo de búsqueda consistirá en cambiar un número L de genes y comprobar si el individuo obtenido es mejor que el original.

Algorithm 4: **Búsqueda local simple**

```
1: MejorIndividuo <- IndividuoActual
2: Combinaciones <- Todas las combinaciones de L elementos de un total de N
3: for combinación ∈ Combinaciones do
4:    NuevoIndividuo <- IndividuoActual
5:    for i ∈ combinación do
6:       GenesNuevoIndividuo[i] <- Cambiar(GenesNuevoIndividuo[i])
7:    end for
8:    if Adaptabilidad(NuevoIndividuo) >Adaptabilidad(MejorIndividuo) then
9:       MejorIndividuo <- NuevoIndividuo
10:   end if
11: end for
```

Experimentación:
Problema de la suma de subconjuntos



Aplicaciones al problema de decodificación de un código lineal

Tengamos una matriz binaria C de dimensión n x k, un número t de errores un un mensaje c=mC + e, del que desconocemos e.

Método directo: cada individuo de la población será un vector de tamaño n, cada posición suya será un gen y la función de perdida será la distancia de cada individuo por C al mensaje c.

Método con la inversa: calculamos D una inversa por la derecha de C, tal que CD = Identidad. Cada individuo de la población será un vector de tamaño k y peso de Hamming t. Tal que ahora cada individuo intenta aproximar el error e y no el mensaje original m. La función de perdida será multiplicar cada individuo por DC y calcular su distancia con c.

Experimentación:
Problema de decodificación de un código lineal