

- Clasificación de los problemas:

Tiempo polinomial: aquellos tiempos que están representados por: $O(1)$, $O(n)$, $O(n^2)$, $O(n^3)$, ..., $O(n^k)$. K es un número, n es el tamaño del problema.

Unidad II. Clasificación de los problemas

tiempo polinomial $\rightarrow O(1), O(n), O(n^2), O(n^3), \dots, O(n^k)$
donde $k \geq 0$
 $a_0 n^0 + a_1 n^1 + a_2 n^2 + a_3 n^3 + \dots + a_k n^k$
 k es un número
 n es el tamaño del problema.

Problema del viajante de comercio (agente viajero).

5 ciudades
viajante vive en la ciudad 3

$C = \{1, 2, 3, 4, 5\}$
 $(3, 1, 2, 4, 5)$

Matriz de distancias entre ciudades.

	1	2	3	4	5
1	0	1	5	0	3
2	1	0	1	3	4
3	5	1	0	2	4
4	0	3	2	0	3
5	3	4	4	3	0

$P \neq NP$

Es el conjunto de problemas que se resuelven en tiempo polinomial

No polinomial
Ej: Factorial en permutaciones.

➔ No se conoce un algoritmo en tiempo polinomial que lo resuelva. ➔ Obtener la solución óptima.

Ejemplos de problemas P: Ordenamiento, búsqueda, operaciones de matrices

Clasificación de Problemas

Universo de todos los computables

Se resuelven con ambas máquinas de Turing.

Se resuelven con máquina de Turing no determinística

P: Conjunto de los problemas que se resuelven con una máquina de Turing determinística con un algoritmo cuya complejidad de tiempo es polinomial.

Modelo de Turing

modelo de máquina abstracto

Máquina de Turing Determinística

• Viajante de comercio

Se necesitan

Problemas:

- Ordenamiento
- búsqueda de elemento en arreglo

no determinística

Se necesitan/

NP: Es el conjunto de problemas que se deben implementar en una máquina de Turing no determinística. y el algoritmo que comprueba una solución, tiene una complejidad de tiempo polinomial.

-
- Diagram illustrating a Turing Machine (T.M.) component:
- Módulo de programas para escribir** (Program module for writing)
 - cabezal lectora/escritora** (Reader/writer head)
 - cinta infinita por ambos extremos** (Infinite tape in both directions)

$P_k = a_0 + a_1n + a_2n^2 + \dots + a_kn^k$ (polinomio), para adquirir $k \geq 0$.

Mag. Turing NP : e
No Determinística

Modelo de ~~computação~~ ^P ~~computação~~ ^(shor)
Modelo de ~~computação~~ ^P ~~computação~~ ^(shor)
Calcular cada coisa separadamente

cinta infinita

Viajante n cidades
espacio de posibles
soluciones es $n!$

Mag Turing NP: es el conjunto de Problemas que se necesita implementar en una Máquina de Turing No determinística y el algoritmo que comprueba si un resultado obtenido es una solución tiene una complejidad Polinomial es decir $O(p_n)$

Ejemplo de Problema P: Sea U un conjunto de números naturales cuya cardinalidad es par. Encuentre dos subconjuntos A y B tal que $A \cap B = \emptyset$ y $A \cup B = U$, de tal forma que: $|A| = |B|$ y

1. Ordenados $A = \{2, 4, 11\}$ $B = \{8, 15, 22\}$
 $\sum a_i = 17$ $\max \left(\sum_{i=1}^{|A|} a_i - \sum_{i=1}^{|B|} b_i \right)$ $A = \{2, 4, 8\}$ $B = \{11, 15, 22\}$
 $\sum b_i = 45$

(1) 1er Paso Ordenar de menor a mayor
 (2) 2do Paso Sumar los primeros n números
 (3) 3er Paso " " segundas n números y hacer la resta cardinalidad del 1er y 2do subconjunto

Viajante n euclides
espacio de posibles
soluções es $n!$

1 es para A
0 es para B

Ordenamos

→ 10 00 000
1234567
(2)4,5,3,10,16,1

→ 1 0 1 1 0 1 0
2 3 4 5 6 7

$$A = \{2\}$$
$$B = \{4, 5, 3, 10, 15\}$$
$$A = \{2, 4\}$$
$$B = \{5, 3, 10, 15\}$$

18

$$- 34 \mid = 24$$
$$2^n = \sum_{i=0}^n \binom{n}{i}$$

$O(2^n)$ → genera todos la columnas
de 0 y 1.

Ejemplo de Problema NP: Sea T un conjunto de números naturales. Encuentre dos subconjuntos A y B tal que $A \cap B = \emptyset$ y $A \cup B = T$ de forma tal que:

na taxa que:

$$\min \left| \sum_{i=1}^{|A|} a_i - \sum_{i=1}^{|B|} b_i \right|$$

La heurística nos da la idea

$$A = \{1, 15\} \cup \{3, 5\} \quad B = \{2, 10\} \cup$$

Le un balance entre las sumas de los elementos de los conjuntos A y B.

Algoritmo para Generar las cadenas
Binarias en orden Lexicográfico

↓
000
001
010
011
100
101
110
111

funcion-Imprimir-Menor() {
print("Menor", Menor);

3 2 1
1 000
2 001
3 010
4 011
5 100
6 101
7 110
8 111

$n=3$

$2^3 = 8$

$V \begin{bmatrix} 1 & 2 & 3 \\ 5 & 2 & 6 \end{bmatrix}$

$A = \{ \}$

$B = \{ 5, 2, 6 \}$

funcion-Sumador-A-y-B() {
sumaA = 0; sumaB = 0;
for i = n to 1 {

if $g[i] == 1$ sumaA = sumaA + $V[i]$;

else sumaB = sumaB + $V[i]$;

}
if Menor > abs(sumaA - sumaB) { Menor ← abs(sumaA - sumaB);
funcion-Guarda-Menor(); }
}

Algoritmo Generación de código
Gray reflectivo binario

menor ← ∞
for j = 0 to n+1 do { $g_i \leftarrow 0$
 $r_i \leftarrow j+1$
i ← 0
output(g_0, g_1, \dots, g_i)
i ← 0
funcion-Sumador-A-y-B

$g_i \leftarrow \overline{r_i}$

$r_{i-1} \leftarrow r_i$

$r_i \leftarrow i+1$

if $i \neq 1$ then $r_0 \leftarrow 1$

Legenda

Si 1 pongo el elemento a
sumador del conjunto A
Si 0 pongo el elemento
al sumador del conjunto B

funcion-Imprimir-Menor();

funcion-Guarda-Menor(); }

Pseudocódigo para min ($\sum a_i - \sum b_i$) {

// Generando cadenas binarias con el código Gray

menor ← ∞

funcion-Leer(U) {

for j = 0 to n+1 do {

$g_j \leftarrow 0$

$r_j \leftarrow j+1$

i ← 0

while (i < n+1) {

output(g_0, g_1, \dots, g_i)

funcion-Sumador-A-y-B()

i ← r_0

$g_i \leftarrow \overline{r_i}$

$r_{i-1} \leftarrow r_i$

$r_i \leftarrow i+1$

if $i \neq 1$ then $r_0 \leftarrow 1$

}

funcion-Imprimir-menor()

}

1 2 3 4
 $U = \{ 3, 4, 10, 8, 2 \}$

0 1 2 3 4 5 6
 $g = 0 1 1 0 0 0 1$

$A = \{ 4, 10 \}$

$B = \{ 3, 8, 2 \}$

Legenda si $g[i] = 1$ $U[i] \in A$

si $g[i] = 0$ $U[i] \in B$

funcion-Sumador-A-y-B() {

sumaA = 0;

sumaB = 0;

for i = n to 1 {

if ($g[i] == 1$) sumaA = sumaA + $U[i]$;

else sumaB = sumaB + $U[i]$;

}

if (menor > abs(sumaA - sumaB)) {

menor ← abs(sumaA - sumaB)

for i = n to 1 {

$g_{menor}[i] \leftarrow g[i]$

}

} // fin if

} // fin funcion-Sumador-A-y-B()

0 1 2 3 4
 $g = 0 1 1 0 0 0 1$

$n=3$

// guarda el menor hasta el momento

Pseudocódigo para Max ($\sum a_i - \sum b_i$) {

// Tenemos un problema de Optimización

Leer-Arreglo-U();

Ordenar-Quick sort(); // menor a mayor

sumaA = 0; sumaB = 0;

for i = 1 to $|U|/2$ do {

sumaA = sumaA + $U[i]$;

sumaB = sumaB + $U[i + |U|/2]$;

}

imprimir("El maximo es", sumaA - sumaB);

Imprimir("se alcanza cuando A contiene a la mitad de los elementos más pequeños de U y B a la mitad de los más grandes, siendo $|A| = |B|$ ");

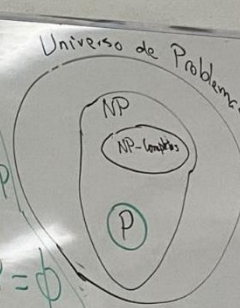
Problemas NP-Completo

Definición: Un Problema NP-Completo es aquel problema considerado como los más difíciles de resolver en la clase de Problemas NP.

$$P \subseteq NP$$

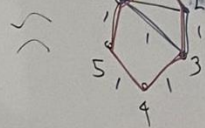
$$NP-completo \subseteq NP$$

$$NP-completo \cap P = \emptyset$$



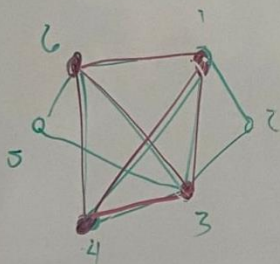
Agente Viajero

Un grafo es Hamiltoniano



Lista de Problemas NP-Completo.

Problema del Clique Máximo



- PLANTEAMIENTOS DE PROBLEMAS NP - COMPLETOS

Problema de, viajante comercio (PVC) o agente viajero: El problema PVC consiste (TSP Traveling Saleman Problem) en encontrar un recorrido cerrado en un grafo pesado de n vértices (ciudades) tal que la suma de las distancias entre las ciudades recorridas sea la mínima. Donde un recorrido cerrado de n ciudades se puede considerar como una permutación cíclica.

- Otro planteamiento del problema PVC

El PVC consiste en:

Planteamientos de Problemas NP-Completo.

4 es tamaño del problema

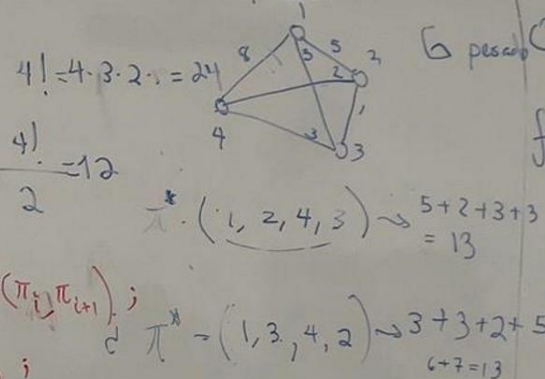
1	0	5	3	18
2	5	0	1	2
3	3	1	0	3
4	8	2	3	0

$O(n)$ { suma = 0
for $i=1$ to n
 suma = suma + $d(\pi_i, \pi_{i+1})$;
suma = suma + $d(\pi_n, \pi_1)$;
if suma \leq Umbral

Matriz de distancias

0	$d_{1,2}$	$d_{1,3}$	$d_{1,4}$
$d_{2,1}$	0	$d_{2,3}$	$d_{2,4}$
$d_{3,1}$	$d_{3,2}$	0	$d_{3,4}$
$d_{4,1}$	$d_{4,2}$	$d_{4,3}$	0

(costo)
tiempo
combustible



Otro Planteamiento del Problema PVC

EL PVC consiste en
$$f(\pi^*) = \min_{\pi \in \Pi_n} \left[\sum_{i=1}^{n-1} d_{\pi_i, \pi_{i+1}} + d_{\pi_n, \pi_1} \right] \quad (1)$$

donde $\pi = (\pi_1, \dots, \pi_n)$ es una permutación, n es número de ciudades, $d_{\pi_i, \pi_{i+1}}$

π^* será el mínimo $d_i(1)$

$n=4$
 $\min_{\pi} [d_{\pi_1, \pi_2} + d_{\pi_2, \pi_3} + d_{\pi_3, \pi_4} + d_{\pi_4, \pi_1}]$

$\pi = (2, 4, 3, 1) \rightarrow 13$
 $\pi = (3, 2, 1, 4) \rightarrow 17$

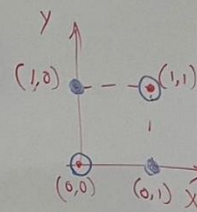
Problema de Satisfabilidad de una función Booleana (NP-completo) Teorema de Cook.

$f(x, y, z)$: función Booleana x, y, z

$x, y, z \in \{0, 1\}$

$f(x, y, z) \in \{0, 1\}$

x	y	z	f(x,y,z)
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0



$f(x,y) = 0$

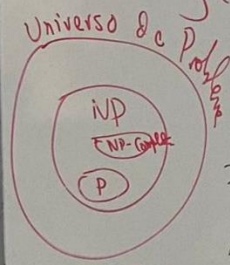
$f(x,y) = 1$

Gráfica

Tabla

Análisis

$f(x,y) = \bar{x}$



Problema de Satisfabilidad:

Para que valores de las variables la función es verdadera

$$2^2 = 4$$

$$x_1, x_2, \dots, x_n \rightarrow 2^n$$

$$O(2^n)$$

$P \neq NP$

$P = NP$

$P \neq NP$

$P \subseteq NP$ (NP) (P) $P, NP \rightarrow P = NP$

UNIDAD 3

TÉCNICAS PARA EL DISEÑO DE ALGORITMOS PARA RESOLVER UN PROBLEMA DADO

Diseño de un algoritmo glotón (ávido, primero mejor, "greedy").

- Diseñar el algoritmo glotón para el problema del Viajante comercio:
 - "En la iteración donde se encuentre el algoritmo, decidir tomar la mejor pieza para construir la solución"
 - Su complejidad sería P, va recorriendo una matriz

valor óptimo

"En la iteración donde se encuentre el algoritmo, decidir tomar la mejor pieza para construir la solución"

al valor del óptimo

$\pi = (\pi_1, \pi_2, \dots, \pi_n)$

π^* es el recorrido cerrado óptimo

$$f(\pi) = \min_{\pi} \left(\sum_{i=1}^{n-1} d_{\pi_i, \pi_{i+1}} + d_{\pi_n, \pi_1} \right)$$

$n=6$

Matriz de Distancias

posibles permutaciones

$\pi = (\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6)$

$\pi_{\text{dato}} = (1, 4, 2, 5, 3, 6)$

$d_{1,4} + d_{4,2} + d_{2,5} + d_{5,3} + d_{3,6} + d_{6,1}$

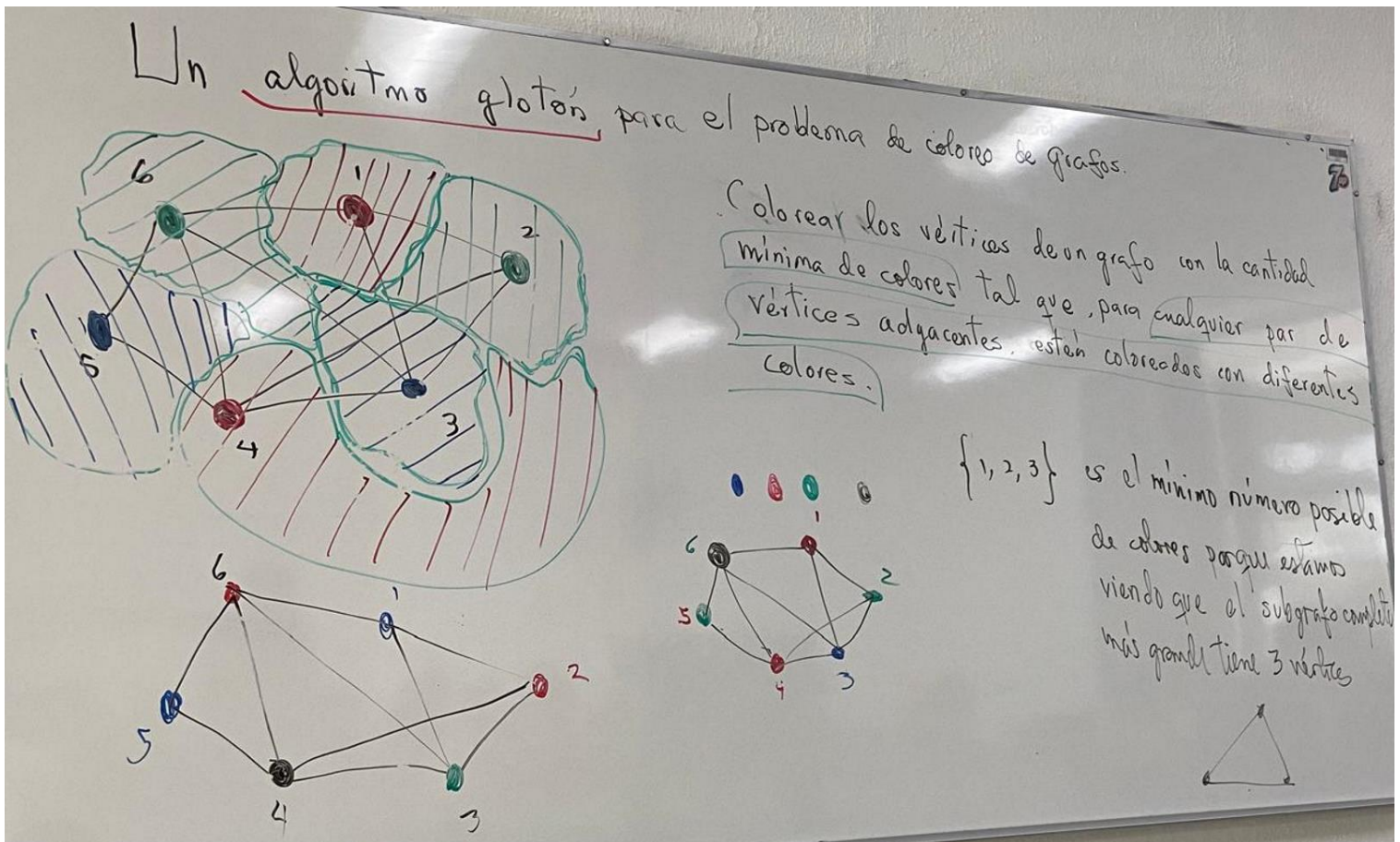
$3 + 2 = 9$

ALGORITMO GLOTÓN:

- Es determinístico
- No genera aleatorios.
- En el coloreo de grafos tratará de minimizar, de utilizar la menor cantidad de colores. En el problema del Viajante de comercio busca ir sumando ciudades de mínimo peso.
- Se usan para tener una sola solución.

Un algoritmo glotón para el problema de coloreo de grafos, tiene aplicaciones como en pintar mapas, trata de que los vértices estén coloreados y sus adyacentes tengan otro color diferente.

- Consiste en colorear los vértices de un grafo con la cantidad mínima de colores tal que para cualquier par de vértices adyacentes estén coloreados con diferentes colores.
- Para este problema no siempre sucede que obtiene la solución óptima.
- Comenzar con el que más vecinos tiene y colorear los que no están relacionados con el con otro color. Pero puede haber muchos criterios por el cual escoger. También se puede seleccionar al principio el que menor relaciones tiene y así colorear los demás no vecinos del mismo color. No sabes cual criterio es mejor para ese grafo.



PRACTICA

1. Diga si las siguientes sentencias son Verdaderas o Falsas:

(F) Un problema NP-Completo es el problema de Ordenamiento de elementos en una lista.

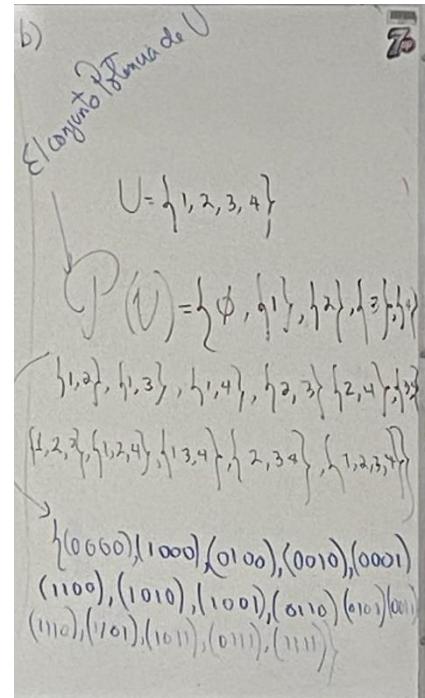
Justificación: la complejidad de un algoritmo para el problema de ordenamiento se conoce como $O(n^2)$ o $O(n \log n)$, esto quiere decir que el problema de Ordenamiento está en la clase P de Problemas. En donde NPC es Non deterministic Polinomial complete Problems. Los problemas P no están contenidos en NPC

(V) La máquina de Turing en la que puede resolver un problema NP-Completo es la no determinística.

Justificación: Las NP-Completo requieren un adivinador para encontrar la solución del problema.

(V) Un problema tal que una posible solución se puede comprobar con un algoritmo de complejidad $O(n^k)$ donde $k \geq 1$, es NP aunque las posibles soluciones a generar podrán llevar un tiempo exponencial.

Justificación: Por la definición ya que la comprobación de una solución se hace en tiempo polinomial en una máquina de Turing no determinística.



Práctica.

1 - Diga si las siguientes sentencias son Verdaderas o Falsas. Justificar

F

Un problema NP-completo es el problema de Ordenamiento de elementos en una lista.

V

La máquina de Turing en la que se puede resolver un problema NP-completo es la no determinística.

V

Un problema tal que una posible solución se puede comprobar con un algoritmo de complejidad $O(n^k)$ donde $k \geq 1$, es NP, aunque las posibles soluciones a generar puedan llevar un tiempo exponencial.

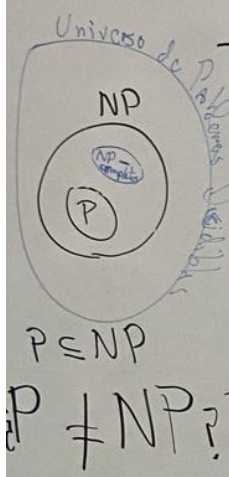
La complejidad de un algoritmo para el problema de Ordenamiento se conoce como $O(n^2)$ ó

$O(n \log n)$ y por lo tanto el problema de Ordenamiento está en la clase P de Problemas. $P \cap NPC = \emptyset$.

donde NPC es Non deterministic Polinomial complete Problems.

V Los NP-completo requieren un adivinador para encontrar la solución del problema.

V Por la definición ya que la comprobación de una solución se hace en tiempo polinomial en una máquina de Turing no determinística.

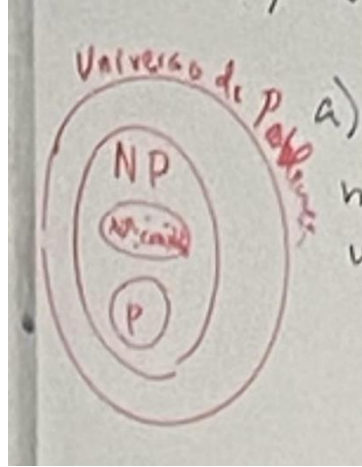


EJERCICIOS

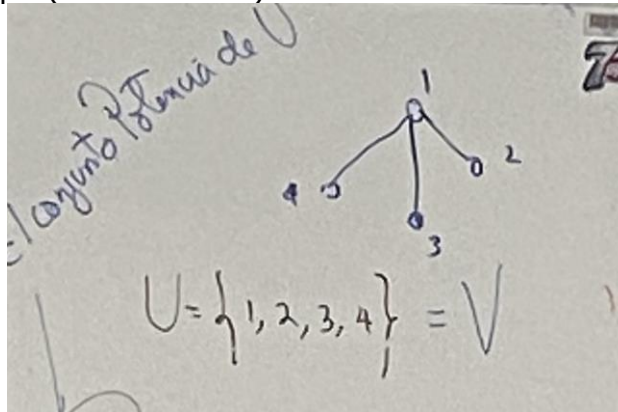
EJERCICIOS. Sobre problemas P, NP y NP-Completo.

Dado los siguientes problemas clasifíquelos según su complejidad y justifique su respuesta.

Tenemos el siguiente universo de problemas:



- a. Problema de cubrimiento de aristas por vértices. El problema de encontrar el subconjunto de vértices de un grafo $G=(V,E)$ más pequeño tal que toda arista del conjunto E de G , tenga un extremo en el subconjunto de vértices más pequeño. Generar todos los subconjuntos de un conjunto dado, en este caso el conjunto potencia de U .
- Si te dicen que el subconjunto más pequeño del grafo es (1) o (1000) entonces todas las aristas conectan con 1 [si no te dicen cuál es el más pequeño siempre será el vacío] y que es un grafo simple (no tiene bucles).



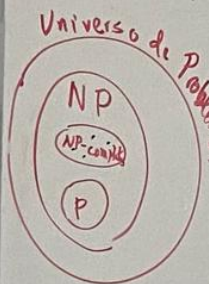
- Identificar que estamos en el caso del conjunto Potencia de V .
- Todas las cadenas posibles diferentes binarias de tamaño n .
- Si tenemos 2^n posibles soluciones entonces tendríamos que generar cada una y quedarme con el conjunto de vértices más pequeño que cumple que "toda arista del grafo tenga un extremo en el conjunto".
- Por lo tanto, la complejidad del algoritmo sería 2^n y no tengo otra forma de menor complejidad para conocer el conjunto más pequeño, o sea, no tengo un algoritmo de menor complejidad conocido.
- Respuesta: Por lo anterior, el problema sería NP y es NP-Completo porque está en la lista de Problemas NP-Completo.

Para que sea NP-Completo debería estar en la lista de problemas de NP-Completo, encontrar el juego de valores para los cuales es verdadera, se debería de pasar al problema de satisfabilidad.

Siempre es pensar en que busca el problema, cuál es su enfoque e identificar en primera instancia en que caso estas, por ejemplo, conjunto Potencia, generando permutaciones a manera de subconjuntos y este sería básicamente NP.

Ejercicios. Sobre problemas P, NP y NP-completos

Dada los siguientes problemas clasifíquelos según su complejidad y justifique su respuesta.

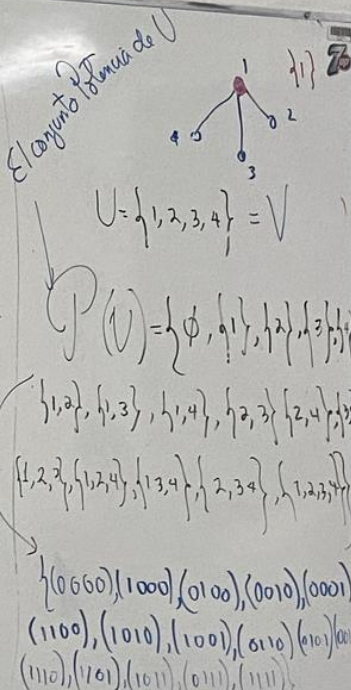


a) El problema de encontrar el subconjunto de vértices de un grafo $G=(V,E)$ más pequeño tal que toda arista del conjunto E de G , tenga un extremo en el subconjunto de vértices más pequeño.

R/ Por lo anterior el problema es NP y es NP-Completo porque está en la lista de Problemas NP-completos

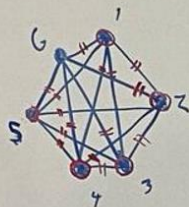
Identificar que estamos en el caso del conjunto Potencia de V

Si tenemos 2^n posibles soluciones, entonces tendríamos que generar cada una y quedarme con el conjunto de vértices más pequeño que cubra que toda arista del grafo tenga un extremo en el conjunto. La complejidad del algoritmo es 2^n y no tengo un algoritmo de menor complejidad.



d) Para un grafo completo el conjunto más pequeño de vértices que cubre todas las aristas del grafo, qué tamaño o cardinalidad tiene?

- 5 cubre
- 5, 1
 - 5, 2
 - 5, 3
 - 5, 4
 - 5, 6



R/ Si $G=(V,E)$ y $|V|=n$ entonces

Grafo Completo

con $n-1$ vértices se cubren todas las aristas de un grafo completo

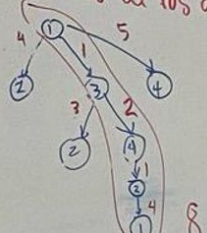
$$| \{5, 1, 3, 4, 2\} | = 6 - 1 = 5$$

Definición - Un vértice cubre una arista de un grafo si es uno de los dos vértices de la arista

1, 3, 4, 2, 1

	1	2	3	4
1	0	4	1	5
2	4	0	3	1
3	1	3	0	2
4	5	1	2	0

Árbol de Búsqueda de un algoritmo Glotón

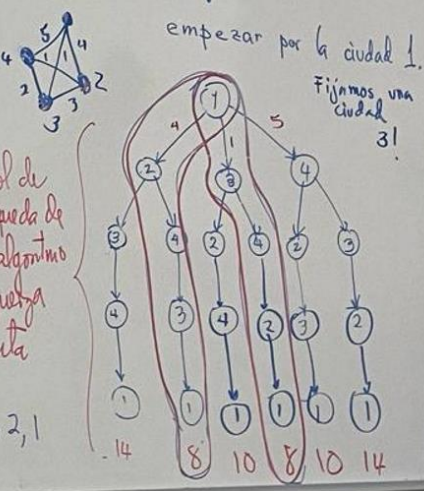


1, 3, 4, 2, 1

Árbol de búsqueda implícito. (No lo construyo en el código fuente)

Problema del viajante de comercio

empezar por la ciudad 1. Fijamos una ciudad 3!

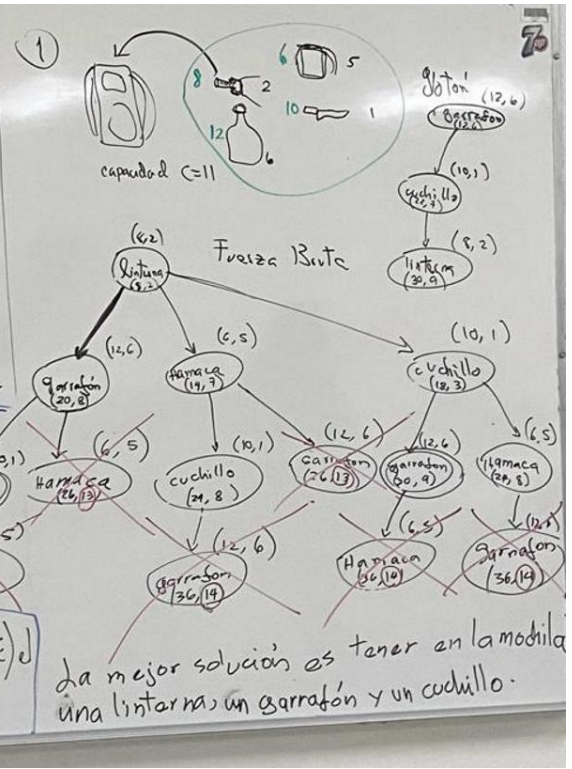
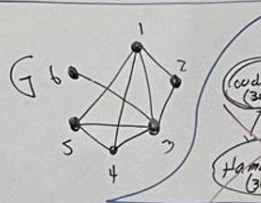


Práctica. Hacer árboles de búsqueda para los algoritmos de fuerza bruta y glotón de los siguientes problemas:

- ① Problema de la Mochila: En una mochila con capacidad $C=11$, se desea guardar los objetos más valiosos para sobrevivir en una selva. Se tiene la siguiente información sobre el valor de cada objeto y la capacidad que este ocupa.

(1,2,3,4)
(0,0,0,0)
(0,0,0,1)
(0,0,1,0)

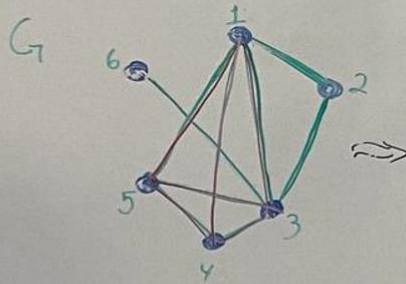
Objetos	valor	capacidad
1 linterna	8	2
2 garrafón	12	6
3 hamaca	6	5
4 cuchillo	10	1



- ② Problema del clique Máximo: Encuentra en el siguiente grafo $G=(V,E)$ el subgrafo completo más grande que este contenga.

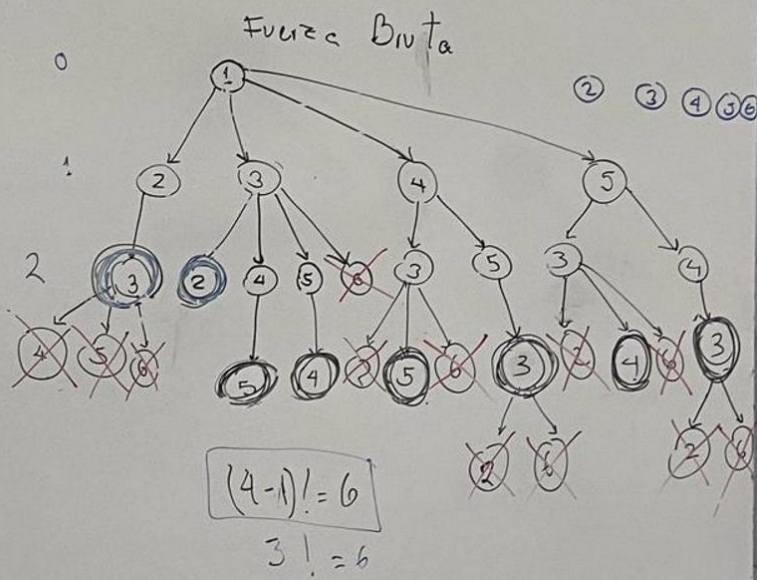
Clique: conjunto de vértices que están conectados con todos los demás. Su grado la cantidad de vértices menos uno

- Hacer el árbol de búsqueda para el algoritmo de fuerza bruta y el glotón.
② Problema del clique máximo:
Encuentre en el siguiente grafo $G=(V,E)$ el subgrafo completo más grande que este contenga.



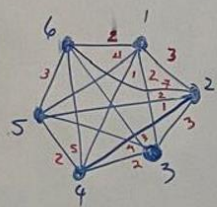
Matriz de Adyacencia

	1	2	3	4	5	6
1	0	1	1	1	1	0
2	1	0	1	0	0	0
3	1	1	0	1	1	1
4	1	0	1	0	1	0
5	1	0	1	1	0	0
6	1	0	1	0	0	0



Tarea sobre programación de algoritmos de fuerza bruta y algoritmo gloton para los siguientes problemas:

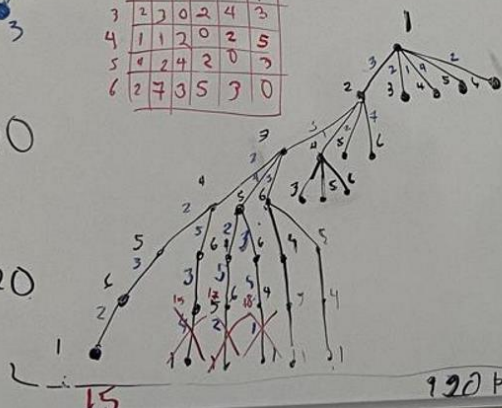
a) Problema del Viajante de comercio (Fijando la primera ciudad)
Fuerza Bruta (Algoritmo Primero en Profundidad ó en anchura)



	1	2	3	4	5	6
1	0	3	2	1	4	2
2	3	0	3	1	2	7
3	2	3	0	2	4	3
4	1	1	2	0	2	5
5	4	2	4	2	0	3
6	2	7	3	5	3	0

$$6! = 720$$

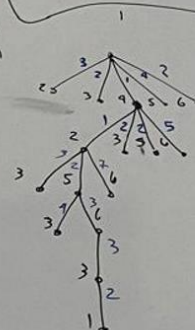
$$5! = 120$$



120 Permutaciones

b)

Gloton

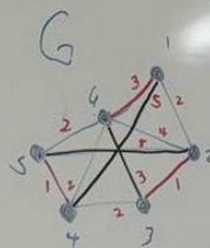


$$(1, 4, 2, 5, 6, 3, 1)$$

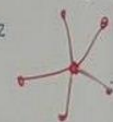
$$distancia: 1+1+2+3+3+2=12$$

Descripción del Problema del Pareo Perfecto Máximo.

Sea $G=(V, E)$ tal que para toda arista de E se asigna un peso que es un número natural sin el cero ($\mathbb{N} \setminus \{0\}$), donde $|V|$ es un número par. Encontrar el pareo perfecto de peso máximo, entendiéndose por pareo perfecto aquel conjunto de aristas $P \subseteq E$ tal que para cualquier par de aristas e_i, e_j no tengan vértices comunes. Un pareo perfecto de peso máximo es aquel que la suma de los pesos de sus aristas es la más grande posible. Además el conjunto de arista P (Pareo) debe cubrir todos los vértices del grafo.



$$E = \{ \{1, 2\}, \{1, 4\}, \{1, 6\}, \{2, 3\}, \{2, 5\}, \{3, 4\}, \{3, 6\}, \{4, 5\}, \{4, 6\}, \{5, 6\} \}$$



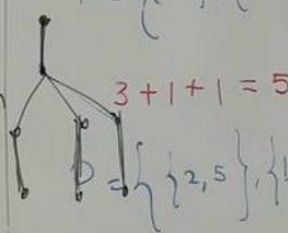
Alg. Gloton

$$P = \{ \{1, 6\}, \{2, 3\}, \{4, 5\} \}$$

Paso 1. Elige una arista con el peso mayor.

Paso 2. Elige la siguiente arista tal que no tengan vértices en común con las anteriores y que sea la siguiente peso mayor posible.

Paso 3. Es el final hasta que tengamos



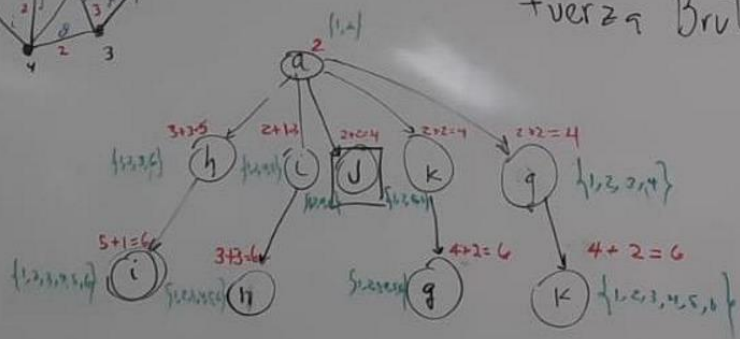
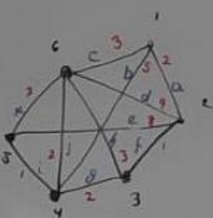
$$3 + 1 + 1 = 5$$

$$8 + 5 + 3 = 16$$

$$\frac{|V|}{2} \text{ aristas en el Pareo}$$

Pareo Perfecto Máximo

Fuerza Bruta.



b

c

d

e

j

g

h

i

j

k