NOTAS DEL CURSO SOBRE COMPLEJIDAD ALGORITMICA

Por J.Diaz, Prof. de la Facultad de Informática. U.P.B.

Las páginas que transcribimos a continuación corresponden a las transparencias utilizadas por el Prof. J. Diaz en el curso sobre Complejidad Algoritmica, impartido en el Centro de Cálculo de Universidad Complutense, en Noviembre de 1980.

Es nuestra intención al publicarlas, que sirvan de guia al lector interesado en el tema, pudiendo completarlas con las referencias que aparecen en la Bibliografía.

INTRODUCCION

Desde que en 1936, A. Turing formalizó el concepto de "Procedimiento Mecánico" para resolver problemas, es decir Algoritmos, ha ido en aumento el interés por construir algoritmos más eficaces (rápidos y efectivos) y clasificar a los problemas de acuerdo a su complejidad computacional. En 1965 J. Edmonds (5) sugirió una clasificación de problemas decidibles en BUENOS, aquellos para los que existia un algoritmo determinista de complejidad polinomica, (n^k, para k=cte) y MALOS, aquellos problemas para los que los mejores algoritmos deterministas conocidos hasta el momento que resuelven dicho problema, tienen complejidad exponencial 2^n. Al principio de los 70, dichas clases cristalizarían en las clases P y NP, que darían pié a la actual teoría de la complejidad. La necesidad de esta clasificación de los problemas, viene expresada en la tabla siguiente, en la que como se ve la diferencia de tiempo de computación entre un algoritmo de complejidad polinomica y uno de complejidad exponencial, es tal como para hacer practicamente imposible el calcular el algoritmo de complejidad exponencial para valores grandes de n.

Funcion complejidad / tiempo	Tamaño n: 10	20	30	40	50	60
n	.00001 segundo	.00002 segundo	.00003 segundo	.00004 segundo	.00005 segundo	.00006 segundo
n^2	.0001 segundo	.0004 segundo	.0009 segundo	.0016 segundo	.0025 segundo	.0036 segundo
n^3	.001 segundo	.008 segundo	.027 segundo	.064 segundo	.125 segundo	.216 segundo
n^5	.1 segundo	3.2 segundos	24.3 segundos	1.7 minutos	5.2 minutos	13.0 minutos
2^n	.0001 segundo	1.0 segundo	17.9 minutos	12.7 dias	35.7 años	366 siglos
3^n	.059 segundo	58 minutos	6.5 años	3855 siglos	$2x10^8$ siglos	$1.3x10^{13}$ siglos

En la primera parte del curso se exponen los conceptos básicos de la complejidad, clases P,NP,NPC, Reducibilidad, etc... Esta sacada casi integramente del excelente libro de Garey y Johnson (7), y de los papeles originales de Cook (3) y Karp (10). Otros artículos que se pueden consultar con diferentes enfoques son Diaz (4), Hopcroft-Ullman (8) cap. 12-13, Karp (11) y Penttonen (14). A continuación va un desarrollo sobre las estructuras internas de la clase NPI y el teorema de Ladner (cronologicamente esto fue explicado el último día del curso). El material está sacado del artículo de Ladner (13).

Seguidamente se presenta un tipo de reducción más fuerte que el de Karp-Cook, pero que preserva la "estructura combinatoria" de los problemas.

Una vez un problema se le clasifica como perteneciente a la clase NPC, se puede perder toda esperenza de encontrar un algoritmo rápido que solucione dicho problema. Entonces existen cuatro posibles cursos de acción a seguir:

* Utilizar métodos heurísticos (aquí no se mencionan)

* Resolver subproblemas de dicho problema.

  En el curso después de demostrar que el problema del Recubrimiento de Vertices Minimo es NPC, se demostró que en el caso de que el grafo venga generado por un Orden Parcial, el problema tiene complejidad líneal (Berenguer, Diaz, (2)).

* Utilizar Algoritmos de Aproximación.

  Solo se mencionó esta posibilidad y se dieron basicamente dos referencias una el capítulo 6 del Garey, Johnson (9), otra el artículo de Johnson (9).

* Utilizar técnicas Probabilisticas a partir de las técnicas probabilisticas aplicadas a la teoria de grafos (6), Karp (12) y otros por ejemplo Anghluin-Valiant (1), utilizaron algoritmos probabilisticos para resolver casos particulares de problemas NP-Completos (*). Un nuevo tipo de analisis probabilistico, debida al matemático vietnamita Phan Dihn Dieu (15) entreve la posibilidad de utilizar el concepto de redución de Ausiello, explicado anteriormente, para generalizarlo a parte de la clase de problemas NPC.

El Curso es una introducción a la complejidad, y dejo grandes lagunas de la teoría de la complejidad sin tocar, concentrandome unicamente en las dos clases más "prácticas" P y NP.

(*) Sin embargo estos tipos de Algoritmos probabilisticos son de dificil generalización.

CONSIDEREMOS EL PROBLEMA Π: (VIAJANTE COMERCIO)

ENTRADA: DADOS UN CONJUNTO DE m CIUDADES

$$\{C_1, C_2, C_3 \ldots C_m\}$$

Y UN COSTO ASOCIADO A CADA PAR DE CIUDADES

$$d(C_i, C_j)$$

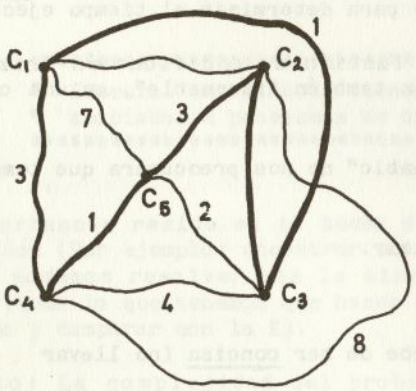
PROPIEDAD A SOLUCIONAR: ENCONTRAR LA ORDENACION DE LAS m CIUDADES $\{C_{\Pi(1)}, C_{\Pi(2)}, \ldots C_{\Pi(m)}\}$, QUE MINIMICE EL COSTO TOTAL

$$\left\{ \sum_{i=1}^{m-1} d(C_{\Pi(i)}, C_{\Pi(i+1)}) \right\} + d(C_{\Pi(m)}, C_{\Pi(m+1)})$$

EJEMPLO: Una entrada E de Π :

E: $C = \{C_1, C_2, C_3, C_4, C_5\}$

$d(C_1,C_2) = 3$; $d(C_1,C_3)=1$; $d(C_1,C_4)=3$; $d(C_1,C_3)=7$
$d(C_2,C_3) = 2$; $d(C_2,C_4)=8$; $d(C_2,C_5)=3$
$d(C_3,C_4) = 4$; $d(C_4,C_5)=1$; $d(C_3,C_5)=2$



(En grueso un circuito para el viajante de comercio)

Una solución: $\{C_1, C_3, C_2, C_5, C_4\}$

¿¿ UN ALGORITMO ??

SOLUCION: PRODUCIR TODOS LOS COSTOS T. POSIBLES

¿ ES EL ALGORITMO MAS EFICIENTE?

¡¡¡¿¿¿ EFICIENTE ??????

EFICIENTE: MAS RAPIDO (MENOS ESPACIO), CON RESPECTO A OTROS ALGORITMOS QUE RESUELVAN EL MISMO PROBLEMA, PARA TODAS (LA MAYORIA) DE ENTRADAS DEL PROBLEMA.

ESTA MEDIDA DE EFICIENCIA SE EXPRESA EN FUNCION DE UNA VARIABLE QUE ES EL TAMAÑO DE LA ENTRADA.

Por ejemplo, en el caso del Viajante de Comercio (V.C.) tenemos el número de ciudades, m, sus m etiquetas y las $\frac{m(m-1)}{2}$ distancias entre las ciudades.

Para definir de manera precisa el tamaño de la entrada, una (entre otras) forma es el codificar esos datos de la entrada utilizando un alfabeto fínito, mediante una función de codificación $\propto$, que a cada entrada E del problema le hace corresponder una cadena del alfabeto, y definir el tamaño de la entrada I como la longitud de su codificación.

Considerando el alfabeto $\sum = \{0, 1, ,\}$ y el ejemplo del viajante comercio visto anteriormente podemos expresar E por:

1,10,11 100,101,,,11,1,11,111,,,10,1000,11,,100,10,,1

¿CUAL ES LA $\propto$?

Bajo esta $\propto$, el tamaño de E es 52.

LA FUNCION DE COMPLEJIDAD TEMPORAL DE UN ALGORITMO EXPRESA EL TIEMPO DE EJECUCION (PEOR TIEMPO) PARA RESOLVER UN PROBLEMA CON ENTRADA DE UN TAMANO DETERMINADO.

Esta función no está bien definida, hasta que se especifique la función de codificación y el modelo de máquina usado para determinar el tiempo ejecución.

Pero, en realidad si utilizamos una función de codificación "razonable" diferirá en la complejidad de otra función también "razonable", en una cantidad polinómica.

Por lo tanto siempre que $\propto$ sea "razonable" no nos preocupara que tomemos una codificacion u otra.

El tiempo razonable es dificil de formalizar.

INTUITIVAMENTE SE PUEDE EXPRESAR POR:

(1) La codificacion de una entrada E debe de ser concisa (no llevar informacion necesaria)

(2) Los numeros en E deben ser expresados en binario o en cualquier otra base diferente de la monaria.

De hecho, generalmente se mide el tamano de la entrada, simplemente por el número de parametros.

En el algoritmo de busqueda exhaustiva para el VC, la complejidad sera $\geq O(2^n)$ ¡NO EXCESIVAMENTE EFICIENTE!

Con los modelos de maquina pasa algo similar a lo de la funcion de codificacion, si utilizamos un modelo "razonable" la funcion de complejidad temporal diferira de la funcion compleja con otro modelo razonable en una cantidad polinomica

Un modelo no razonable seria una maquina con capacidad de realizar muchas operaciones simultaneas en paralelo.

Vamos a considerar los problemas en los que queramos estudiar la complejidad, en forma decisional.

Un problema decisional Π consiste en un conjunto de entradas E_Π , y un subconjunto $S_\Pi \leq E_\Pi$ de entradas para las que el problema tiene respuesta si.

El VC en forma decisional seria:

ENTRADA: Un conjunto fínito de ciudades

$$C = \{C_1, C_2, \ldots C_m\}$$

Un conjunto de distancias entre cada par de ciudades $d(C_i, C_j)$

Una cota superior de $K \in \mathbb{Z}^+$

PROPIEDAD: Existe una ordenación $\{C_\Pi(1), C_\Pi(2), \ldots C_\Pi(m)\}$

tal que

$$\left\{\sum_{i=1}^{m-1} d(C_\Pi(i), C_\Pi(i+1))\right\} + d(C_\Pi(m), C_\Pi(1)) \leq K$$

```
**********************************************************************
*  Especial importancia tienen los problemas decisionales            *
*  asociados a problemas de optimizacion (maximo-minimo)             *
**********************************************************************
```

Esta importancia reside en el hecho de que si podemos resolver el problema de optimizacion (Por ejemplo: encontrar un circuito de costo mínimo para el VC), tambien podemos resolver con la misma complejidad el problema decisional asociado (todo lo que tenemos que hacer en el caso del VC, es sumar el circuito encontrado y comparar con la K).

Por tanto: La complejidad del problema de optimizacion nos determina la complejidad del problema decisional asociado.

(Si el problema optimizacion es sencillo de resolver $\Rightarrow$ el problema decisional tambien es sencillo) $\Longrightarrow$

$\Longrightarrow$

```
**********************************************************************
* (SI EL PROBLEMA DECISIONAL PERTENECE A LA CLASE PROBLEMAS          *
* "DIFICILES" ⇒ EL PROBLEMA OPTIMIZACION TENDRA QUE PERTENE-         *
* CER COMO MINIMO A DICHA CLASE)                                     *
**********************************************************************
```

El reciproco no tiene que ser necesariamente cierto siempre.

¿Por que introducir problemas en forma decisional?

Σ : CONJUNTO FINITO DE SIMBOLOS (ALFABETO)
Σ^*: CONJUNTO DE TODAS LAS CADENAS FINITAS DE ELEMENTOS DE
$L \leq \Sigma^*$ DIREMOS QUE L ES UN LENGUAJE SOBRE Σ

EJEMPLO: $\Sigma = \{0,1\}$;
$\Sigma^* = \{0,0,1,00,01,10,11,000,001,010,100,011 \ldots\ldots\}$
$L = \{01,001,110,10100\}$

RELACION LENGUAJES-PROBLEMAS

Si tenemos un problema decisional Π y tenemos un alfabeto específico Σ y una función de codificación $\propto : \{E\} \rightarrow \Sigma^*$ (es decir una función que asocia a cada dato específico de entrada (una codificación) una cadena de Σ^*), entonces Π y $\propto$ parten Σ^* en tres clases disjuntas:

1) Cadenas que no representan a ninguna entrada de Π, codificada.
2) Cadenas que representan las entradas codificadas de Π para las cuales la solución es NO.
3) Cadenas que representan las entradas codificadas de Π para las cuales la solución de Π es SI. Esta última clase la definiremos como el lenguaje asociado a Π, $\propto$

$L_\Pi = \{X \in \Sigma^*$: Σ alfabeto utilizado por $\propto$, x es la codificación de una entrada $E \in S_\Pi\}$

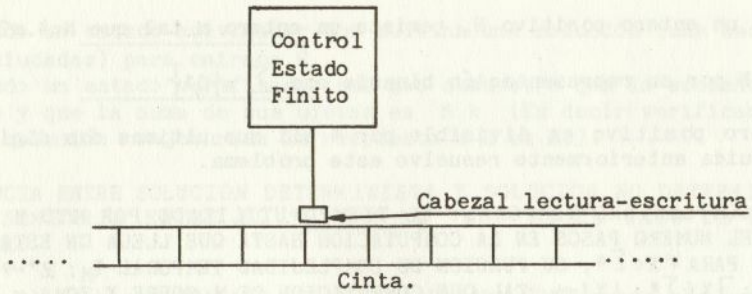
MAQUINA TURING

UNA MAQUINA TURING DETERMININISTA M:

$$< Q, \Sigma, q_0, F, \delta >$$

Q : Conjunto fínito de estados
Σ : Conjunto fínito de simbolos
q_0: Estado inicial
F : Conjunto estados finales ($\{q_s, q_n\}$)
δ : Funcion Transición: $\delta : (Q-F) \times \Sigma \rightarrow (Q-F) \times \Sigma \times \{D,I\}$

Esta máquina representa un programa (un procedimiento)



EJEMPLO: $\Sigma = \{0,1\} \cup \emptyset$
$Q = \{q_0, q_1, q_2, q_3, q_s, q_n\}$

δ	0	1	
q_0	$(q_0,0,D)$	$(q_0,1,D)$	$(q_1,0,I)$
q_1	$(q_2,0,I)$	$(q_3,0,I)$	$(q_n,0,I)$
q_2	$(q_s,0,I)$	$(q_n,0,I)$	(q_n,b,I)
q_3	$(q_n,0,I)$	$(q_n,0,I)$	(q_n,b,I)

Modelo de programa para la maquína de Turing.

Una MTD M, con alfabeto Σ, acepta $x \in \Sigma^*$, sii se para en estado final aceptador, (q_s), cuando se le aplica como entrada x.

Definiremos como el lenguaje reconocido por M:

$$L_M = \{x \in \Sigma^* : M \text{ ACEPTA } x\}$$

Si $x \in \Sigma^* - L_M$, la MTD M se puede parar en estado q_n o simplemente se puede no parar.

Pero para que una maquina turing determinista corresponda a la nocion de algoritmo, debe de pararse para todas sus posibles entradas !!!!

Diremos que una MTD M resuelve el problema decisional Π si M se para para todas sus entradas y ademas $L_M = L_\Pi$

EJEMPLO: Dado un entero positivo N, ¿existe un entero m tal que N=4.m?

$\propto$ codifica N por su representación binaria con $\Sigma = \{0,1\}$

Como un entero positivo es divisible por 4 sii sus ultimos dos digitos son 0, la MTD construida anteriormente resuelve este problema.

DEFINICION COMPLEJIDAD TEMPORAL: EL TIEMPO UTILIZADO POR MTD M SOBRE UNA ENTRADA X ES EL NUMERO PASOS EN LA COMPUTACION HASTA QUE LLEGA UN ESTADO FINAL. SI M, MTD, SE PARA $\forall X \in \Sigma^*$, SU FUNCION DE COMPLEJIDAD TEMPORAL $T_M: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$
$T_M(X)=\max \{m: \exists X \in \Sigma^*, |X|=m$, TAL QUE COMPUTACION DE M SOBRE X TOMA m PASOS $\}$

M ACTUA EN TIEMPO POLINOMICO SI EXISTE UN POLINOMIO P TAL QUE $\forall N \in \mathbb{Z}^+$, $T_M(n) \leq p(n)$.

****
* $P=\{L: \exists$ MTD M actuando tiemp. Polin., tal q. $L=L_M\}$ *
****

****
* DIREMOS $\Pi \in P$ SI $L_\Pi \in P$ *
****

Informalmente: intercambiaremos la MTD actuando en tiempo polinomico por Algoritmo (determinista) de tiempo polinomico.

P ≡ clase problemas para los que existen algoritmos deterministas polinomicos, que los resuelven.

MAQUINA TURING NO DETERMINISTA (MTND)

Intuitivamente un ALGORITMO NO DETERMINISTA resuelve un problema Π (decisional) si $\forall E \in E_\Pi$:

(a) Si $E \in S_\pi \Rightarrow$ existe una solucion adivinada S tal que el algoritmo comprobora que realmente es una solución valida y dará como resultado, SI.
(b) Si $E \notin S_\Pi$ no existe solución adivinada que nos produzca el resultado SI para E.

Un Algoritmo No Determinista consta de dos estados:

(I) Conjetura (Adivina) una solución.
(II) Verifica que realmente es una solución y si esta verificación se produce en tiempo polinomico, se dice que el algoritmo tiene complejidad polinomica.

VERIFICACION POLINOMICA $\not\Rightarrow$ SOLUCION POLINOMICA

Por ejemplo, un algoritmo no determinista con tiempo polinómico para el VC se puede construir:

1) Utilizando un estado adivinador que adivina una solución (una secuencia de las ciudades) para entrada E.
2) Utilizando un estado verificador que nos demuestre que la secuencia es un circuito y que la suma de sus costos es $\leq k$ (Es decir verificamos que para la solución conjeturada la respuesta a E es SI).

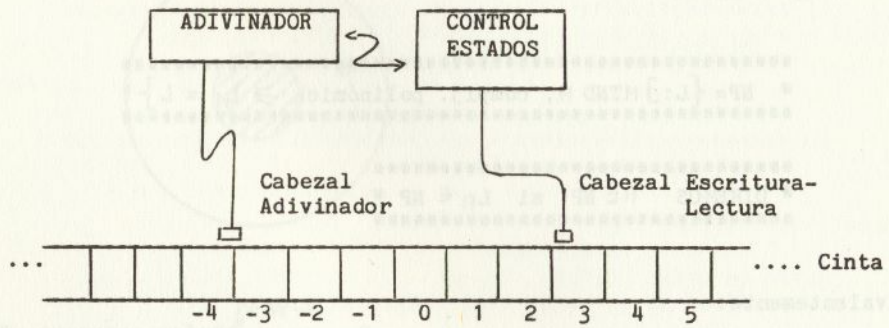
UNA DIFERENCIA ENTRE SOLUCION DETERMINISTA Y SOLUCION NO DETERMINISTA ES LA FALTA DE SIMETRIA ENTRE EL "SI" Y EL "NO" EN EL CASO DE SOLUCION NO DETERMINISTA:

Si tenemos el problema "DADA E, ES X VERDAD PARA E?" puede ser resuelto por un algoritmo determinista con complejidad polinomica, su problema complementario Π^c : "DADA E, ES X FALSO PARA E?" también puede ser resuelto por un algoritmo determinista con complejidad polinomica. (Esto es porque un algoritmo determinista se para para todos las entradas, por lo tanto lo único que tenemos que hacer es intercambiar las respuestas SI y NO (q_S, q_n).

PERO: No es obvio (M conocido en muchos casos) que la misma propiedad exista en el caso de algoritmos no deterministas.

Si consideramos el complementario del VC, "¿Es cierto que no existe ningún circuito de las m ciudades de costo total $\leq K$?" No se conoce ningun algoritmo no determinista CON COMPLEJIDAD POLINOMICA, que resuelva el problema.

MTND M



Especificación MTND identica a la MTD

$$< Q, \Sigma, q_0, F, \delta >$$

****
* excepto δ no es 1-1 *
****

Esto es equivalente a:

La computacion de una MTND M. sobre entrada X, difiere de la computacion MTD, en que se realiza en dos estados.

En el primero "el estado adivinador" el cabezal adivinador escribe una solucion conjeturada y activa el q_0 entonces viene "el estado verificador" en donde M actuando deterministicamente comprueba que verdaderamente lo escrito por el cabezal adivinador es una solucion para la entrada X. la computacion termina que q_s ó q_n, y es computacion que acepta X si termina en estado q_s

EL LENGUAJE RECONOCIDO POR M:

$$L_M = \{x \in \Sigma^* : M \text{ acepta } x\}$$

EL TIEMPO UTILIZADO POR MTND M PARA ACEPTAR UNA ENTRADA $X \in L_M$ SE DEFINE COMO EL NUMERO MINIMO DE PASOS QUE OCURREN DURANTE LA COMPUTACION DE M HASTA QUE ENTRA q_s

La funcion de complejidad temporal $T_M : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$

$$T_M(n) = \text{MAX}\left\{\{1\} \cup \left\{m: \exists\, x \in L_M,\ |x|=n,\ \ni \text{tiempo} \atop \text{comput. de M sobre x es m}\right\}\right\}$$

M actua en tiempo polinomico si existe un polinomio p tal que
$\forall\ n \geq 1$, $T_M(n) \leq p(n)$.

$$NP = \{L: \exists \text{ MTND M, complj. polinómica} \ni L_M = L\}$$

DIREMOS $\Pi \in NP$ si $L_\Pi \in NP$

Equivalentemente:

NP ≡ Clase problemas para los que existen algoritmos no deterministas de tiempo polinomico que los resuelven.

RELACION P y NP

Lema 1: $P \leq NP$

Teorema 1: SI $\Pi \in NP$, EXISTE UN POLINOMIO p TAL QUE Π PUEDE SER RESUELTO MEDIANTE UN ALGORITMO DE COMPLEJIDAD (TEMPORAL) $O(2^{p(n)})$

Demostración

$\Pi \in NP \Rightarrow \exists$ algoritmo ND A con complejidad polinómica q(n). Para cada entrada aceptada x, $|x|=n$, debe de existir una solución conjeturada de longitud $\leq q(n)$, que conduce a que el estado verificador de A responda SI en menos de q(n) pasos $\Rightarrow$ El numero de conjeturas posibles que puede realizar A es $\leq c^{q(n)}$, donde $|\sum| = C$.

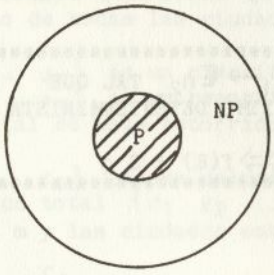
Podemos descubrir deterministicamente si A "tiene" una computación que acepte x, explorando cada una de los $c^{q(n)}$ posibles conjeturas, la simulación responde SI, cuando encuentra una computación que acepte x.

La complejidad total de la simulación es $O(q(n).\ c^{q(n)})$.

GRAN PROBLEMA

$$\begin{array}{l} P \subsetneq NP \\ \text{¿} \quad \text{ó} \quad ? \\ P = NP \end{array}$$

****
* conjetura $P \subsetneq NP$ *
****



Dado $\Pi \in NP$, queremos saber $\begin{array}{l} \nearrow \Pi \in NP\text{-}P \\ \searrow \Pi \in P \end{array}$?

ó

****
* SI $P \neq NP$ ENTONCES $\Pi \in NP\text{-}P$ *
****

Introducimos el concepto TRANSFORMACION POLINOMICA entre problemas

Un lenguaje $L_1 \subseteq \sum_1^*$ es reducible a $L_2 \subseteq \sum_2^*$ si existe una funcion $f: \sum_1 * \rightarrow \sum_2^*$ tal que:

1) Existe una MTD M que computa f con complejidad polinomica

2) $\forall x \in \Sigma_1^*$, $x \in L_1$ sii $f(x) \in L_2$

****
* $L_1 \propto L_2$ *
****

Lema 2: SI $L_1 \propto L_2$ ENTONCES $L_2 \in P \Rightarrow L_1 \in P$
($L_1 \notin P \Rightarrow L_2 \notin P$)

Demostración:

$L_1 \propto L_2$ $(L_1 \leq \Sigma_1^* \wedge L_2 \leq \Sigma_2^*) \Rightarrow \exists f: \Sigma_1^* \to \Sigma_2^*$ y una MTD M_f que computa f. Si M_2 es MTD que reconoce $L_2 \Rightarrow M_2$ o M_f es MTD que reconoce L_1. ■

INFORMAL:

SI Π_1, Π_2 SON PROBLEMAS DECISIONALES (con $\propto_1$, $\propto_2$)

DIREMOS $\Pi_1 \propto \Pi_2$ SI $L_{\Pi_1} \propto L_{\Pi_2}$

****
* EQUIVALENTEMENTE, SI $\exists f: E_{\Pi_1} \to E_{\Pi_2}$ TAL QUE *
* 1) f COMPUTABLE POR UN ALGORITMO DETERTERMINISTA *
* POLINOMICO *
* 2) $\forall E \in E_{\Pi_1}$, TENEMOS $E \in S_{\Pi_1} \Leftrightarrow f(E) \in S_2$ *
****

Ejemplo: Consideremos $\Pi_2 \equiv$ VC
$\Pi_1 \equiv$ CIRCUITO HAMILTONIANO
Dado un grafo G(V,A), $|V|=m$
G contiene un circuito Hamiltoniano?

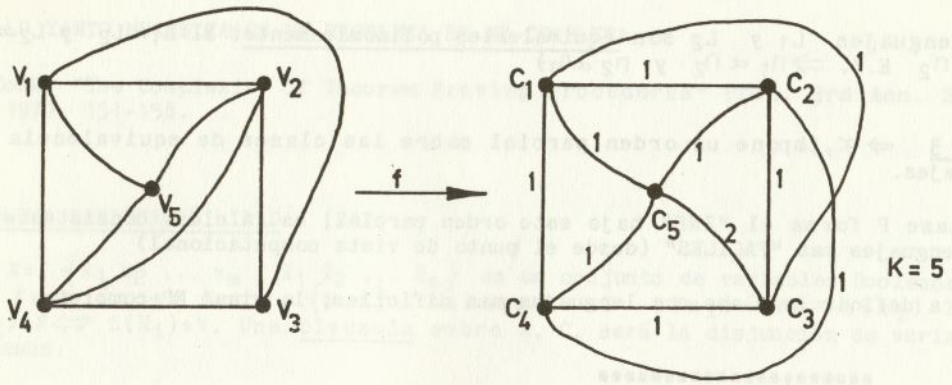
Vamos a demostrar $\Pi_1 \propto \Pi_2$

Necesitamos $f: E_{\Pi_1} \to E_{\Pi_2}$ $(G: \to C_1 \wedge d \wedge k)$

DEFINICION f: $C \equiv V$, $|V|=m$

$\forall v_i \; v_j \in C$, $d(v_i, v_j) =$ → 1 si $\overline{v_i \; v_j} \in A$
→ 2 si $\overline{v_i \; v_j} \notin A$

$K = m$



1) Conforme, f se puede computar con un algoritmo determinista polinomico, para cada una de los $\frac{m(m-1)}{2}$, posibles distancias, tenemos solo que examinar G, para ver si $(v_i\ v_j)$ pertenece o no a A.

Vamos a ver 2. Debemos demostrar que G contiene un circuito Hamiltoniano sii existe un recorrido de todas las ciudades en f(G), cuyo costo total sea $\leq K$.

($\Rightarrow$) Si $\{v_1\ v_2\ \ldots\ v_m\}$ es un circuito Hamiltoniano en G
$\Rightarrow$ $\{v_1\ v_2\ \ldots\ v_m\}$ es un recorrido de todas las ciudades y la longitud total de este recorrido es m = K.

($\Leftarrow$) Si $\{v_1\ v_2\ \ldots\ v_m\}$ es una solución al VC en f(G)
$\Rightarrow$ longitud total $\{v_1\ v_2\ \ldots\ v_m\} \leq K = m$. Como $|\{v_1\ v_2\ \ldots\ v_m\}| = m$ y las ciudades estan separadas por 1 ó 2 $\Rightarrow$

$\forall\ v_i,\ v_{i+1} \in \{v_1, v_2, \ldots v_m\}$, $d(v_i, v_{i+1}) = 1$

$\forall\ \overline{v_i, v_{i+1}} \in A$. y por lo tanto $\{v_1\ v_2\ \ldots\ v_m\}$ es un Circuito Hamiltoniano en G.

■

****
* El lema 2, nos dice que si $\forall C \in P \Rightarrow CH \in P$ y si *
* $CH \notin P \Rightarrow VC \notin P$ *
****

Lema 3: $\propto$ ES TRANSITIVA

$(L_1 \propto L_2)$ y $(L_2 \propto L_3) \Rightarrow L_1 \propto L_3$

Demostración

Si $\Sigma_1\ \Sigma_2\ \Sigma_3$ son los alfabetos de $L_1\ L_2\ L_3$
$f_i : \Sigma_i^* \rightarrow \Sigma_i^*$; $f_2 : \Sigma_2^* \rightarrow \Sigma_3^* \Rightarrow f_2 \cdot f_1 : \Sigma_1^* \rightarrow \Sigma_3^*$

■

Dos lenguajes L_1 y L_2 son equivalentes polinomicamente. SI $L_1 \propto L_2$ y $L_2 \propto L_1$ (Π_1 Π_2 E.P. $\Rightarrow \Pi_1 \propto \Pi_2$ y $\Pi_2 \propto \Pi_1$)

Lema 3 $\Rightarrow \propto$, impone un orden parcial sobre las clases de equivalencia de lenguajes.

La clase P forma el "INF" bajo este orden parcial, es la clase consistente de los lenguajes mas "FACILES" (desde el punto de vista computacional)

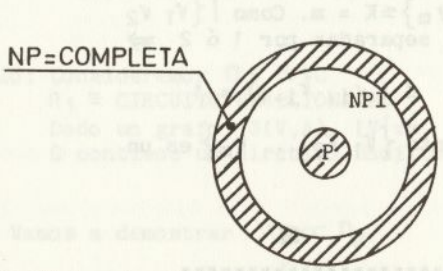
Vamos a definir la clase con lenguajes mas dificiles; la clase NP-completa

```
**********************
* L ∈ NP-COMPLETA SI *
* 1) L ∈ NP          *      DEFINICION NP-COMPLETA
* 2) ∀ L'∈NP, L'∝L   *
**********************
```

EQUIVALENTEMENTE: $\Pi \in$ NP- COMPLETA SI:
1) $\Pi \in$ NP
2) $\forall \Pi' \in$ NP, $\Pi' \propto \Pi$

El lema 2 nos indentifica la clase NP-Completa como la clase de problema "más dificiles" en NP. SI UN PROBLEMA $\Pi \in$NP-COMPLETA ES TAL QUE $\Pi \in P \Rightarrow P = NP$ Y SI $\exists \Pi \in$ NP-COMPLETA, $\Pi \notin P \Rightarrow \forall \Pi' \in$NP-COMPLETA $\Pi' \notin P$.

SI $P \neq NP$:



Lema 4: SI L_1, $L_2 \in$ NP, $L_1 \in$NP-COMPLETA, Y $L_1 \propto L_2$
$\Rightarrow L_2 \in$ NP-COMPLETA

Si enunciamos lema 4 en terminos de problemas decisionales este Lema nos da un método de producir nuevos problemas NP-Completos, UNA VEZ QUE TENGAMOS AL MENOS UN PROBLEMA EN LA CLASE NP-COMPLETA.

PARA DEMOSTRAR $\Pi \in$ NP-Completa, TENEMOS QUE COMPROBAR:

1) $\Pi \in$ NP
2) $\exists$ ALGUN Π' DEL QUE CONOCEMOS $\in$ NP-Completa, TAL QUE $\Pi' \propto \Pi$.

POR LO TANTO NECESITAMOS UN PROBLEMA EN NP-COMPLETA

S. Cook: "The Complexity of Theorem Proving Procedures" Proc. 3rd Ann. STOC, ACM 1971, 151-158.

PROBLEMA SATISFACIBILIDAD (SAT)

Si $X = \{x_1\ x_2\ \dots\ x_m\ \bar{x}_1\ \bar{x}_2\ \dots\ \bar{x}_m\}$ es un conjunto de variables Booleanas, y una función $t: X \longrightarrow \{V, F\}$ tal que si $t(x_i) = V \Leftrightarrow t(\bar{x}_i) = F$ y $t(x_i) = F \Leftrightarrow t(\bar{x}_i) = V$. Una clausula sobre X, C, será la disjunción de variables boolenas.

SAT Dados conjunto de literales $X = \{x_1\ x_2\ \dots\ x_n\ \bar{x}_1\ \dots\ \bar{x}_n\}$
y colección de clausulas $C_1\ C_2\ \dots\ C_m$

Ver si existe una asignación $t: X \rightarrow \{V, F\}$, tal
$(1,0)$

que $\bigwedge_{i=1}^{m} C_i = V$ (1)

Ejm: $X = \{x_1\ x_2\ x_3\ x_4\ \bar{x}_1\ \bar{x}_2\ \bar{x}_3\ \bar{x}_4\}$
$C_1 = \{x_1 \vee \bar{x}_2 \vee \bar{x}_3\}$
$C_2 = \{\bar{x}_1 \vee x_2 \vee x_4\}$
$C_3 = \{x_3 \vee \bar{x}_2\}$

ENTONCES SI $S = \{x_1, x_3, x_4\} \leftarrow V$ (1)
tendremos $C_1 \wedge C_2 \wedge C_3 \equiv V$ (1)

Para que el problema SAT tenga una solución S necesitamos dos condiciones

1) $|S \cap \{x_i, \bar{x}_i\}| \leq 1 \qquad \forall i,\ 1 \leq i \leq n$
2) $S \cap C_j \neq \emptyset \qquad \forall j\ \ 1 \leq j \leq m$

****
* TEOREMA COOK (1971) *
* *
* SAT $\in$NP-COMPLETA *
****

EJEMPLO REDUCIBILIDAD:

SAT 3:

ENTONCES: $\bar{X}$, $|\bar{X}| = 2n$, $C = \{C_1, C_2\ \dots\ C_m\}$ Clausulas tales que $|C_i| = 3$ $1 \leq i \leq m$

PROPIEDAD $\exists\, S \leq \bar{X}$, $\forall x_i \in S$, $t(x_i) = 1 \Rightarrow \bigwedge_{J=1}^{m} C_j = 1$?

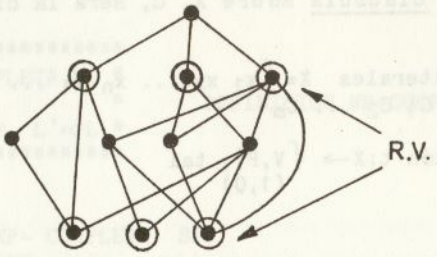
RECUBRIMIENTO DE VERTICES (RV)

ENTRADA: Grafo $G(V,A)$, $K \in N$, $k \leq n = |V|$

PROPIEDAD: Existe un recubrimiento de Vertices de cardinalidad $\leq K$

[Existe $V' \leq V$; $|V'| \leq K$ y tal que para cada arista $\{u,v\} \in A$ al menos uno entre u y v pertenece a V']

Ejemplo:




```
*********************************************
* VAMOS A DEMOSTRAR   SAT 3 ∈NP-Completa *
*                     RV ∈ NP-Completa    *
*********************************************
```

SAT 3 $\in$ NP-COMPLETA

1) SAT 3 $\in$ NP
2) SAT $\propto$ SAT3

Si entrada de SAT = $X = \{x_1 \ldots x_{2n}\}$

Tenemos que construir, mediante f, una entrada de SAT 3 es decir un conjunto X' de literales y un conjunto C' de clausulas. Para construir C' reemplazaremos cada clausula individual $c_j \in C$ por una colección equivalente C'_j de clausulas con 3 literales, para esto utilizaremos extra literales X'_j

Si $c_j = \{z_1 \vee z_2 \vee \ldots \vee z_k\}$, en donde $\forall z_i \in X$, vamos a ver como formamos C'_j y X'_j

Caso 1: K = 1 $X'_j = \{y_j^1, y_j^2\}$
$C'_j = \{\{z_1, y_j^1, y_j^2\}, \{z_1, y_j^1, \bar{y}_j^2\}, \{z_1, \bar{y}_j^1, y_j^2\}, \{z_1, \bar{y}_j^1, \bar{y}_j^2\}\}$

K = 2 $X'_j = \{y_j^1\}$
$C'_j = \{\{z_1, z_2, y_j^1\}, \{z_1, z_2, \bar{y}_j^1\}\}$

K = 3 $X'_j = \emptyset$
$C'_j = \{\{c_j\}\}$

K = 3 $X'_j = \{y_j^i : 1 \leq i \leq k-3\}$
$C'_j = \{\{z_1, z_2, y_j^1\}\} \cup \{\{y_j^i, z_{i+2}, y_j^{i+1}\} : 1 \leq i \leq K-4\}$
$\cup \{\{\bar{y}_j^{k-3}, z_{k-1}, z_k\}\}$

Entonces: $X' = X \cup \left[\bigcup_{i=1}^{m} X'_j \right]$

$C' = \bigcup_{j=1}^{m} C'_j$

($\Rightarrow$) Si t: $X \to \{0,1\}$ sat C podemos extenderlo a un $t' : X' \to \{01\}$ tal que SAT C'. Solo necesitamos que demostrar que t puede extenderse a los conjuntos X'_j, uno a uno, y verificar las correspondientes C'_j son 1. Si X'_j se construyó como caso 1 ó 2, la extensión es arbitraria, $t'(y)=1$ (por ejem). Si X_j construida caso 3 $\Rightarrow$ $X'_j=\emptyset$ y C'_j SAT.

Si X'_j construida caso 4, como t es SAT para C $\Rightarrow \exists$ l tal que $t(Z_l)=1$. Si l=1 ó 2, escribimos $t'(y^i_j)=0$, $1 \leq i \leq k-3$.

Si l=k-1 ó l=K ponemos $t'(y^i_j)=1$ para $1 \leq i \leq K-3$, en caso contrario ponemos $t'(y^i_j)=1$ $1 \leq i \leq l-2$ y $t'(y^i_j)=0$ para $l-1 \leq i \leq K-3$.

$\Rightarrow$ Todas clausulas en C' seran SAT por t'.

Ejemplo: SAT $\propto$ SAT3

Consideremos SAT: X =

$C^1 = \{x_1\}$

$C_2 = \{\bar{x}_1, \bar{x}_3\}$

$C_3 = \{x_2, \bar{x}_3, \bar{x}_4\}$

$C_4 = \{\bar{x}_1, x_2, x_3, x_4\}$

f:

$C_1 \xrightarrow[K=1]{f} X'_1 = \{\{y^1_1, y^2_2\} \cup \{\bar{y}^1_1, \bar{y}^2_2\}\}$

$C'_1 = \{\{x_1 \vee y^1_1 \vee y^2_1\} \vee \{x_1 \vee y^1_1 \bar{y}^2_1\} \vee \{x_1 \vee \bar{y}^1_1 \vee y^2_1\} \vee \{x_1 \vee \bar{y}^1_1 \vee \bar{y}^2_1\}\}$

$C_2 \xrightarrow[K=2]{f} x_1 = \{y^1_2, \bar{y}^1_2\}$

$C'_2 = \{\{\bar{x}_1, \vee \bar{x}_3 \vee \bar{y}^1_2\} \vee \{\bar{x}_1 \vee \bar{x}_3 \vee \bar{y}^1_2\}\}$

$C_3 \xrightarrow[K=3]{f} X'_3 = \{\ \}$

$C'_3 = \{x_2 \vee \bar{x}_3 \vee \bar{x}_4\}$

$C_4 \xrightarrow{f} U'_4 = \{y^1_4, \bar{y}^1_4\}$

$C'_4 = \{\{\bar{x}_1 \vee x_2 \vee y^1_4\} \cup \{\{\bar{y}^1_4 \vee x_3 \vee x_4\}\}$

Lo que queda

$X' = \{x_1, x_2, x_3, x_4, y^1_1, y^2_1, y^1_2, y^1_4, \bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{y}^1_1, \bar{y}^2_1, \bar{y}^1_2, \bar{y}^1_4\}$

$C' = \{\{x_1\ vy^1_1\ vy^2_1\} v \{x_1, vy^1_1 v\bar{y}^2_1\} v \{x_1, v\bar{y}^1_1\ vy^2_1\} v \{x_1\ v\bar{y}^1_1$

$v\bar{y}^2_1\} v \{\bar{x}_1\ v\bar{x}_3\ vy^1_2\} v \{\bar{x}_1\ v\bar{x}_3\ vy^1_2\} v \{x_2\ v\bar{x}_3\ v\bar{x}_4\} v \{\bar{x}_1$

$vx_2\ vy^1_4\} v \{y^1_4\ vx_3\ vx_4\}\}$

$\Rightarrow S = \{x_1, \bar{x}_3, x_2\}$ sol. SAT i.e. $t(x_1)=1$; $t(x_2)=1$; $t(\bar{x}_3)=1$;
$t'(y^1_1)=1$; $t'(y^2_1)=1$; $t'(y^1_2)=1$; $t'(y^1_4)=0$

$\Leftarrow$ t' SAT

($\Leftarrow$) Si t' es una asignación que SAT C', $\Rightarrow$ la restricción a X SAT C

****
∴ * C' es SAT sii C es SAT *
****

∴ SAT3 $\in$ NP - Completa

RV $\in$ NP-Completa
1) RV $\in$ NP
2) SAT3 $\propto$ RV

ENTRADA: SAT3 : $\bar{X} = \{x_1 \dots x_n\}$ $C = \{c_1 \dots c_m\}$, $|C_i| = 3$

Debemos transformarla en ENTRADA RV : Un grafo G(V,A) y $K \in N$
$K \leq |V|$

f: $\forall x_i \in X \longrightarrow T_i = (V_i, A_i)$ donde $V_i = \{x_i, \bar{x}_i\}$
$A_i = \{x_i, \bar{x}_i\}$

$\forall C_j \in C \longrightarrow S_j(V'_j, A'_j)$ donde
$V'_j = \{a_1[j], a_2[j], a_3[j]\}$
$A'_j = \{a_1[j], a_2[j]\}, \{a_1[j], \{a_3[j]\}, \{a_2[j], a_3[j]\}\}$

La única parte de la construcción que depende de qué literales ocurren en qué clausulas, es la comunicación entre las aristas

Si $C_j \in C$, llamemos a los tres literales de C_j: x_j, y_j, z_j

Las extras aristas seran:

$A''_j = \{\{a_1[j], x_j\}, \{a_2[j], y_j\}, \{a_3[j], z_j\}\}$

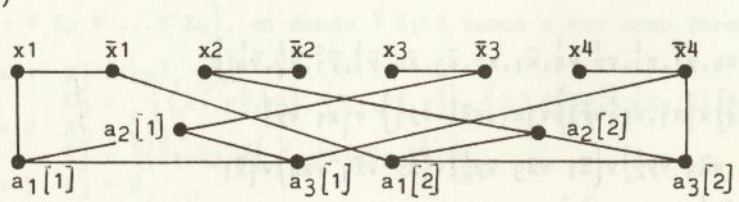
LA ENTRADA DE RV SERA

$K = n+2m$
$G(V,A)$ donde: $V = (\bigcup_{i=1}^{n} V_i) \cup (\bigcup_{j=1}^{m} V'_j)$

$A = (\bigcup_{i=1}^{n} A_i) \cup (\bigcup_{j=1}^{m} A'_j) \cup (\bigcup_{j=1}^{m} A''_j)$

Ejemplo: SI entrada SAT: $X = \{x_1\ x_2\ x_3\ x_4\}$
$C = \{\{x_1, \bar{x}_3, \bar{x}_4\}, \{\bar{x}_1, x_2, \bar{x}_4\}\}$

(Π, S_j, E'')



Vamos a ver C es SAT $\Leftrightarrow$ G tiene RV de cardinalidad $\leq K$
($\Leftarrow$) Si $V' \leq V$ es RV de G, $|V'| \leq K \Rightarrow V'$ debe contener al menos un vertice de cada T_i y dos vertices de cada S_j.
Pero esto nos da un total $\geq n+2m=k$
$\Rightarrow$ V' debe contener exactamente un vertice de cada T_i y 2 de cada S_j.

Vamos a obtener $t:X \rightarrow \{0,1\}$ $t(x_i)=1$ si $x_i \in V'$
$t(x_i)=0$ si $\bar{x}_i \in V'$

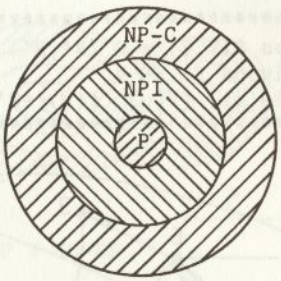
t SAT todas clausulas $C_j \in C$, porque si consideramos E_j''. Sólo dos aristas de E_j'' puede ser recubiertas por $V_j' \cap V_j$
$\Rightarrow$ una arista debe ser recubierta por un vertice de V_i que también pertenece a V' $\Rightarrow$ literal correspondiente, x_i o $\bar{x}_i$ de la c_j es tal que $t(x_i)=1$ ó $t(\bar{x}_j)=1 \Rightarrow C_j=1$. Como esto sucede $\forall c_j \in C \Rightarrow \wedge c_j=1 \Rightarrow$ Si $t:X \rightarrow \{01\}$ hace $\wedge C_i=1$, el correspondiente V' incluye un vertice de cada T_i y dos de cada S_j.

El vertice de Ti en V' es x_i si $t(x_i)=1$ y es $\bar{x}_i$ si $t(x_i)=0$. Esto asegura al menos una de cada 3 aristas en E_j'' esta recubierta.

$\therefore$ Si incluimos en V' los extremos de Sj de las otras aristas en E_j'' nos produce el RV deseado. ■

ESTRUCTURA DE NP

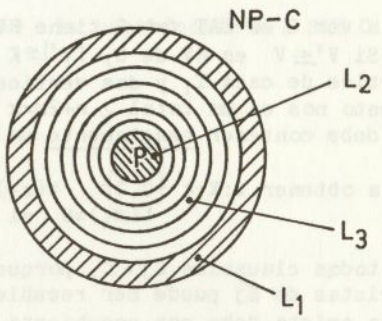
****
* Si P≠NP ⇒ ∃NPI≡NP-(P ∪ NPC) *
****



R.E. Ladner: "On the structure of Polynomial Time Reducibility". J.Assoc. Comput. Mach. 22, 155-171, 1975

TEOREMA: Si L_1 es un lenguaje recursivo tal que $L_1 \notin P$. Esto implica que existe un lenguaje reconocible en tiempo polinomico $L_2 \in P$ tal que si $L_3=L_1 \wedge L_2 \Rightarrow L_3 \notin P$, $L_3 \propto L_1$ pero $L_1 \not\propto L_3$

SI $L_1 \in$ NP-C Y SI P≠NP
$\Rightarrow L_1 \notin P$ Y SI $L_2 \in P$
$\Rightarrow L_3 \in$ NP pero como $L_1 \propto L_3$
$\Rightarrow L_3 \notin$ NP-C y como $L_3 \notin P$
$\Rightarrow L_3 \in$ NPI.



Pero además la clase NPI esta constituida por una colección infinita de clases de equivalencia. Porque si $L_1 \in$ NPI, el teorema de Ladner nos construira un problema L_3 "más facil que" L_1 pero tal que $L_3 \notin P$.

CONSTRUCCION DE LA "CEBOLLA" DENTRO NPI

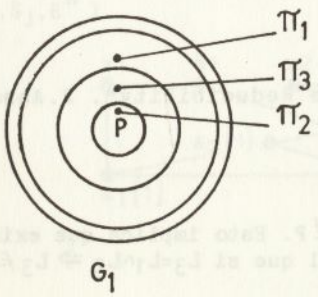
Si P≠NP, el teorema de Ladner $\Rightarrow$ NPI≠$\emptyset$ que:
Supongamos $\Pi_1 \in$ NPI $\Longrightarrow \exists \Pi_2 \in P$ tal que si $\Pi_3 \equiv "\Pi_1 \wedge \Pi_2"$, $\Pi_3 \propto \Pi_1$, $\Pi_1 \not\propto \Pi_3$, $\Pi_3 \notin P$

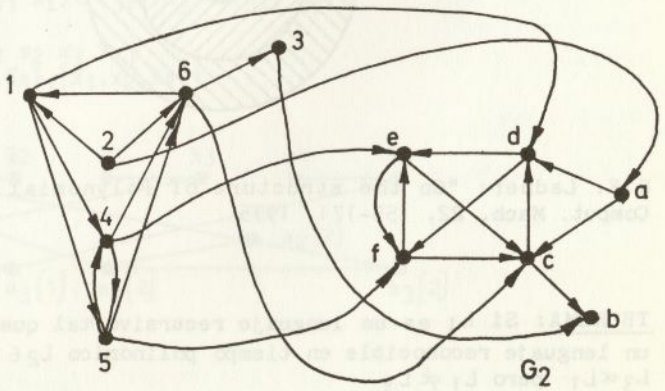
Haciendo esto repetidamente podemos construir unas capas concentricas de clase de complejidad.

¿Se conoce algun problema "natural" que pertenezca a esta clase? No

Cualquier problema en NP pero no NPC es un posible candidato.
Uno de ellos es el problema de encontrar un isomorfismo entre dos grafos.

```
*******************************************************
* Es decir, dados dos grafos G(V,E) y G'(V',E')       *
* ¿existe una función biunivoca f: V→V'               *
* tal que {u,v}∈E si y solo si {f(u), f(v)}∈E' ?      *
*******************************************************
```





Si $L_1 \propto_p L_2$ y $L_2 \propto_p L_1 \Rightarrow L_1 \equiv_p L_2$

TEOREMA: ISOMORFISMO AUTOMATAS FINITOS
Isomorfismo automatas finitos $\equiv_p$ Isomorfismo grafos
Isomorfismo semigrupos $\equiv_p$ Isomorfismo grafos.

Es decir los problemas de isomorfismo de grafos, isomorfismo de semigrupos, isomorfismo de automatas finitos, tienen la misma complejidad.

TARJAN, ha demostrado que el decidir si dos grupos son isomorficos, se puede realizar en $O(n^{2\lg_2 n})$ e donde n es el orden de los grupos.

G.L. Miller: "On the $n^{\log n}$ Isomorphism Technique"
Proc. 10th STOC, 51-58, 1978

Conjetura (1): El isomorfismo de grupos es mas facil que el isomorfismo de grafos.

Conjetura (2): Isomorfismo grafos $\notin$ NPI

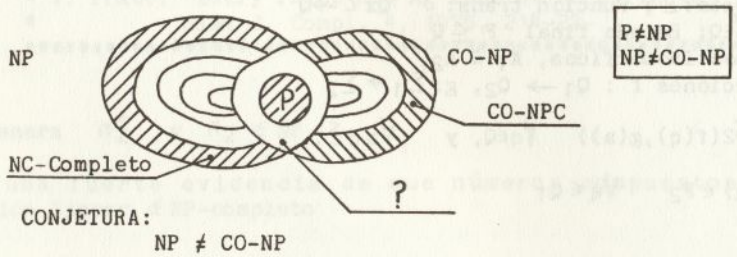
Si (1) Falsa $\Rightarrow$ existe algoritmo $O(n^{c\log n})$ para isomorfismo Grafos; Si (1) Verdad $\Rightarrow$ P $\neq$ NP

Recordemos que si teniamos un problema en forma decisional , el problema complementario Π^c, era el resultado de cambiar las respuestas. $S_{\Pi^c} = \mathbb{E}_\Pi - S_\Pi$

DEFINICION:

$$\text{CO-NP} = \{\Pi^c \mid \Pi \in \text{NP}\}$$
$$\text{CO-NP} = \{\Sigma^* - L : L \text{ lenguaje sobre } \Sigma,\ L \in \text{NP}\}$$

Como ya indicamos, parece que muchos problemas en CO-NP no pertenecen a NP.
Como también indicamos P=CO-NP



Como P=CO-P, si NP≠CO-NP $\Rightarrow$ P≠NP aunque puede suceder que P≠NP incluso en el caso de que NP=CO-NP!

TEOREMA: Si existe problema $\Pi \in$ NP-completa tal que $\Pi^c \in$ NP, entonces NP=CO-NP

Demostración:
Facil $\Pi \in$ NPC $\Rightarrow \Pi^c \in$ CO-NPC = $\forall \Pi'^c$, $\Pi'^c \propto \Pi^c$
pero si $\Pi^c \in$ NP $\Rightarrow \Pi^c \propto \Pi$ Trans. $\Rightarrow$ $\Pi'^c \propto \Pi \Rightarrow \Pi^c \in$ NP

Algoritmo más fácil es utilizar backtracking y examinar todas las posibles correspondencias entre los vertices de los grafos. Si $|V_1'| = |V| = n$, tendremos que la complejidad de este algoritmo es $n! \cong O(2^n)$.

Algoritmo más "inteligente":

Si definimos como cardinalidad de entrada de un nodo el número de aristas que llegan a ese nodo, y cardinalidad de salida, backtracking restringido a nodos igual cardinalidades.

Si el grafo es planar, Hopcroft, Wong: "Linear Time Algorithm for Isomorphism of Planar Graphs" Proc. 6th STOC, 172-184, 1974. Consiguieron un algoritmo en tiempo O(n)!!!

Semigrupo: S=(x,*) tal que * es asociativa y def.

Dos semigrupos: $S,(x_1,*_1) \cong S_2(x_2,*_2)$ sí

$\exists$ biyección $f: x_1 \rightarrow x_2$ tal que

$$x *_1 y = Z \Leftrightarrow f(x) *_2 f(y) = f(z)$$

El Problema del isomorfismo entre semigrupos es determinar si existe una biyección del tipo anterior

Automata Finito: $A=(Q, \Sigma, \delta, S, F)$
Estado: Q, Alfabeto: Σ ; Función trans: $\delta : Q \times \Sigma \rightarrow Q$
Estado inicial $S \in Q$; Estado final $F \subseteq Q$
Dos automatas son Isomorficos, $A_1 \cong A_2$,
si existen biyecciones $f : Q_1 \rightarrow Q_2$, $g: \Sigma_1 \rightarrow \Sigma_2$
tales que
(a) $f(\delta,(q,a)) = \delta_2(f(q),g(a))$ $\forall q \in Q$, y $\forall a \in \Sigma$,
(b) $f(S_1) = S_2$
(c) $q \in F_1 \Leftrightarrow f(q) \in F_2$ $\forall q \in Q_1$

El problema del isomorfismo entre automatas es el determinar si dados dos automatas existen biyecciones del tipo anterior

K.S.Booth "Isomorphism Testing for Graphs Semigroups and finite automata are Polynomially equivalent Problems", SIAM J. Compt. 7,3, 273-279, 1978.

Consecuencia de este teorema: Si tenemos un problema Π tal que Π y $\Pi^c \in NP \Rightarrow \Pi \notin$ NP-Completa excepto en el caso de que NP=CO-NP.

Por lo tanto si un problema Π no sabemos si es NP-Completo ó si es P, pero Π y $\Pi^c \in NP$, eso indica una fuerte evidencia (tan fuerte como $P \subsetneq NP$) de que $\Pi \notin$ NP-Completa.

PROBLEMA:

$$P \overset{?}{=} (NP \cap CO\text{-}NP)$$
$$\text{ó} \qquad ??$$
$$P \subsetneq (NP \cap CO = NP)$$

Vamos a ver dos problemas que tienen dicha propiedad:

Π_1: Números compuestos
Entrada: $K \in \mathbb{Z}+$
Propiedad: ¿Existen enteros m,n>1 tales que k=m.n.?

Π_2* Programacion linear
Entrada: Vectores de enteros $V_i=(v_i[1] \vee_i [2] .- v_i[n])$ $1 \leq i \leq m$
$D = (d_1, d_2, \ldots d_m)$, $C=(C_1, C_2, \ldots C_n)$, $B \in \mathbb{Z}$
Propiedad: Existe un vector racional $X=(x_1, x_2, \ldots x_n)$
tal que $V_i.X \leq d_i$ $1 \leq i \leq m$ y tal que $C.X \geq B$?

$\Pi_1 \in NP \Leftarrow$ SENCILLO
Π_1^c : PRIMOS
: DADO $K \in \mathbb{Z}+$
PROP: ¿ES K PRIMO?

****
* ESTA DEMOSTRADO PRIMOS $\in$ NP *
* V. Pratt: "Every Prime has a succint certificate" *
* SIAM J. Compl. 4, 1975, 214-22- *
****

De igualmanera Π_2^c y $\Pi_2 \in NP$

Esto es una fuerte evidencia de que números compuestos $\notin$ NP-completo programacion linear $\notin$ NP-completo

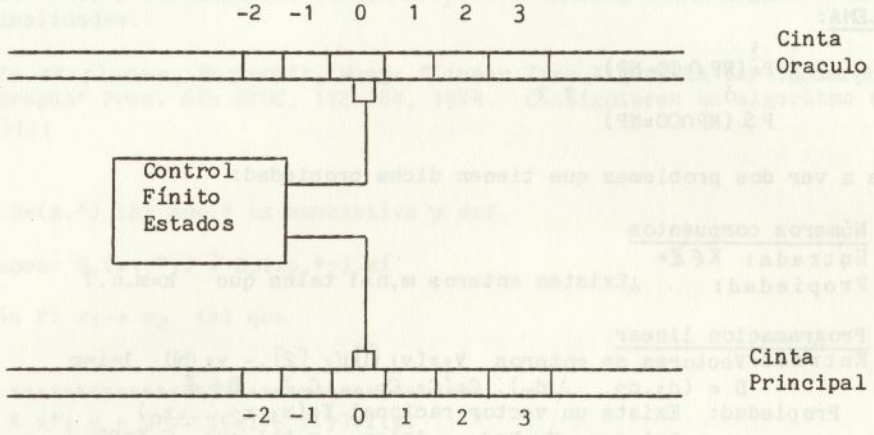
****
* Recientemente: PROGRAMACION LINEAR $\in$ P *
* (Algoritmo Elipsoidal de Katchian) (*) *
****

Números compuestos podia ser candidato $\in$ NPI

(*) L. Lovaz, P. Gacs: The Katchian Algorithm.
Tech Report Stanford Uni. (1979)

TEOREMA LADNER

Una Máquina Turing Oráculo "MTO" consiste en una MTD con una extracinta "cinta Oraculo" con su cabezal lectura-escritura.



La MTO viene especificada:

1.- $\sum$: Alfabeto fínito

2.- Conjunto fínito estados Q: con:

q_0: estado inicial

q_p: estado parada $\rightarrow q_s$, q_n

q_c: estado consulta al oraculo

q_r: estado de volver a comenzar la computación

3.- Función Transición:

$$\delta : \left\{Q - \left\{q_p, q_c\right\}\right\} \times \sum^2 \longrightarrow Q \times \sum^2 \times \left\{D, I\right\} \times \left\{D\ I\right\}$$

La computación es similar a la de la MTD excepto cuando el control estado fínito está en q_c, lo que suceda en el siguiente paso, depende de una función oráculo $g: \sum^* \rightarrow \sum^*$

Comenzamos la computación con la entrada x, $|x|=n$, en la cinta principal, y la máquina en estado q_0. Durante la computación pueden ocurrir tres movimientos de la MTO:

1.- Si se llega al estado q_p, la computación termina

2.- Si el estado es $q \in Q - \left\{q_p, q_c\right\}$, el movimiento siguiente depende del simbolo explorado sobre las dos cintas y de δ. Actua como una MTD pero explorando y moviendo dos cintas.

3.- Si máquina llega a pararse en estado q_c, el próximo paso dependerá del contenido de la cinta oraculo y de la función oraculo g.

Si $y \in \Sigma^*$ es la cadena en cinta oráculo y todo lo demás blanco, y si $g(y)= z$, $z \in \Sigma^*$, en un paso.

* La cinta oraculo pasa a contener Z en ....?.... a $|z|$. El resto de la cinta en blanco).

* El cabezal oraculo se situa sobre "1"

* El control estados fínito cambia de q_c a q_p.

LA DIFERENCIA PRINCIPAL ENTRE MTD Y MTO ES EL PUNTO 3, LA CONSULTA AL ORACULO.

Si la MTO escribe $y \in \Sigma^*$ sobre la cinta oraculo y entra el estado q_c, la respuesta $z=g(y)$, es equivalente a tener una subrutina que nos compute g, PERO EN UN SOLO PASO

LA COMPUTACION DE UNA MTO SOBRE ENTRADA DEPENDE DE "x" Y DE LA FUNCION ORACULO g.

Si tenemos $L_{\Pi 1}$, $L_{\Pi 2}$; $L_1 \leq \Sigma_1^*$, $L_2 \leq \Sigma_2^*$ asociados A Π_1 Π_2
L_1 Turing reducible L_2 ($L_1 \propto_T L_2$)
SI $\exists$ MTO M_{L2} que actuando en tiempo polinómico $x \in L_1 \Leftrightarrow M_{L2}$ se para estado aceptador con entrada x y oraculo L_2 ( ))$x(L_2)$

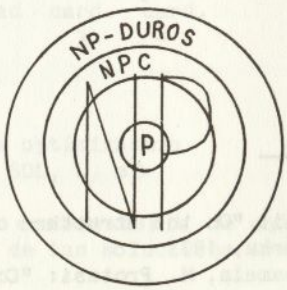
(Referencias: Cook (1971), Turing-Pst (1937))

Si $L_{\Pi 1}$, $L_{\Pi 2}$ (L_1,L_2) L_1 Es KARP reducible L_2
($L_1 \propto L_2$) SI $\exists f: \Sigma_1^* \longrightarrow \Sigma_2^* \ni$ $x \in L_1 \Leftrightarrow f(x)$ L_2 y f puede ser computada MTD tiempo polinómico

(Referencias: Karp (1972), Post (193?))

DEFINICION $L1 \in$ NPDURA SI $\exists L_2 \in$ Np-Completa TAL QUE $L_2 \propto_T L_1$

QUE $\Pi(L_\Pi) \in$ NP-DURA
$\not\Rightarrow$(NECESARIAMENTE) QUE
$\Pi \in$ NP. !!!!



Teorema: Si $B \notin P$, pero B es computable existe un A computable tal que $A \notin P$, $A \not\leq B$, $B \propto_T A$

Corolario: Si $P \neq NP \exists L \in NP\text{-}P$ y tales $L \notin NP\text{-}Compl$.

Demostración: Si $L \in NP - P$ Teor $\Rightarrow L' \notin P \; \exists L' \propto L' \Rightarrow L' \in NP$
pero como $L \not\propto L' \Rightarrow L' \notin NP\text{-}Compl$.

Corolario: Es similar a la solución del problema de POST: Todo conjunto no recursivo pero recursivamente enumerable, es Turing completo ??

La respuesta fue no (Friedberg)

La versión polinímica del problema POST:

Es todo miembro de NP-P, NP-Completo?

REDUCCIONES QUE PRESERVAN LA ESTRUCTURA *

Definición: UN problema de optimizacion A:

$$A \equiv \langle ENTR, SAL, SOL, Q, m \rangle$$

en donde

ENTR: CONJUNTO ENUMERABLE (ENTRADAS DATOS) (Σ_A)
SAL: CONJUNTO ENUMERABLE (SALIDAS)
SOL: $ENTR \longrightarrow \{A \mid A < SAL \wedge |A| < \infty\}$
(es una función recursiva que nos da soluc. aproximadas para cada elemento de ENTR)
Q: CONJUNTO ORDENADO TOTALMENTE ($Q \equiv N$)
m: $SAL \longrightarrow Q$
(m nos da una medida de las salidas sobre las soluciones)

Definición:

El VALOR OPTIMO $m^*(x)$ de UNA ENTRADA x DE UN PROBLEMA DE OPTIMIZACION A ES
$$m^*(x) = MEJOR \{ m(y) / \; y \; SOL(x) \}$$

LA SOLUCION OPTIMA $A^*(x)$ DE UNA ENTRADA x:
$$A^*(x) = \{ y \in SOL(x) / \; m(y) = m^*(x) \}$$

---

* Este material está sacado de:
- G. Ausiello, D. Atri, M.Protasi: "On the Structure of Combinatorial Problems" Proc. th ICALP, Turku, 1977
- G. Ausiello, A. Marchetti Spacamela, M. Protasi: "Combinatorial Problems over Power Sets". TR 79-43, CNR, Roma 1980

EJEMPLO: MAX-CLIQUE=A

Dado un grafo G(V,A) encontrar un clique de cardinalidad máxima.

Entr: Conjunto grafos finitos ($\mathbb{E}_A$)

Sal: Conjunto de grafos completos finitos

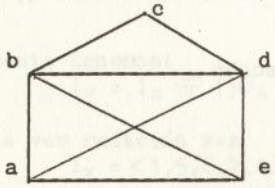
Sol(x): Conjunto de subgrafos completos de un grafo dado x

Q: $\mathbb{N}$ (con la ordenacion usual)

m: Numero de nudos de un grafo completo

Consideremos $x \in ENTR.$:

X:



SOL(X)= {{cbd}, {bda}, {bae}, {deb}, {ade}, {bdae}, {cb}, {cd}, {bd}, {ab}, {ad}, {be}, {ae}, {de}} ∪ {{a}, {b}, {c}, {d}, {e}}

(Todos los subgrafos completos)

m ({ade}) = 3
m ({de}) = 2
m*(x)= m({bdac}) = 4

****
* A*(x) = {bdae} *
****

Estructura Combinatoria:

lx = ⟨1, 5, 8, 5, 1⟩

∅	Soluc. Cardinalidad	soluc. cardinalidad	soluc. card.	sol. card.
	1	2	3	4

Definicion: Dado un problema optimizacion
A = ⟨ENTR, SAL, SOL, Q, m⟩

Definiremos la estructura combinatoria de un entrada $x \in ENTR$, (lx), como la lista de las cardinalidades de las soluciones aproximadas de x.

****
* Formalmente: *
* $l_x = \langle a_1, a_2, \ldots a_n \quad n \geq 0$ tal que *
* $\exists u_1, \ldots, u_n \in Q$, con $u_i < u_{i+1} \quad \forall i<n$ y que *
* (1) $(\forall i)$ $[a_i = \mid A_i(x) \mid]$ *
* (2) $A_i(x) = \{y \mid y \in SOL(x) \wedge m(y) = u_i\}$ *
* (3) $\bigcup A_i(x) = SOL(x)$ *
****

****
NOTA: * $A_n(x) = A^*(x)$ *
****

Definicion: Un problema de optimizacion se define como convexo si para cada x ∈ entr, para cada $a_i \in l_x$, (correspondiente a un valor "u_i" de m), y para cada q $u_i \leq q \leq m^*(x)$, el conjunto de soluciones aproximadas de x, con medida q, no es vacio.

Algunos problemas convexos: max-clique, max-sat, min-recubr

Probl. no convexo max. suma subconj.

Definición: SI A ES UN PROBLEMA OPTIMIZACION Y SI x, y ∈ENTR DIREMOS QUE x ES EQUIVALENTE A Y EN A ($x \equiv_A y$) SII TIENEN LA MISMA ESTRUCTURA ($l_x = l_y$)

$[x]_A$ REPRESENTARA LA CLASE EQUIVALENTE DE x BAJO $\equiv_A$

Si l_r, l_s son estructuras de r,s ∈ENTR, diremos $l_r \leq l_s$ si $l_r = \langle a_1, \ldots a_n \rangle$ y $l_s = \langle b_1, \ldots, b_m \rangle$ y $\exists i_1, \ldots i_n$ con $i_1 < \ldots < i_n \leq m$ y $(\forall j<n)\ a_j \leq b_{ij}$

Claramente es un orden parcial.

Si r,s ∈ entrada, diremos r es un subproblema de S en A ($r \leq_A s$) si la estructura de r es una subestructura de s, $l_r \leq l_s$.

SI $[r]_A$, $[s]_A \in$ ENTRADA/$\equiv_A$. $[r]_A \leq_A [s]_A$ SI $r \leq_A S$

Definición: Si A=⟨ENTR., SAL, Q, m es un problema optimizacion, EL PROBLEMA DECISIONAL ASOCIADO A. Es el conjunto $A_d \subseteq$ ENTRxQ donde $A_d = \{\langle x,k \rangle \mid K \leq m^*(x)\}$ (la desigualdad respecto a la ordenación de Q).

Ejemplo: SI A=MAX CLIQUE, A_j: Encontrar o no si un grafo tiene un CLIQUE de orden ≥ K.

Definición:. Si A_d, B_d son problemas decisionales asociados a A y B

$A_d \propto B_d$ si ∃: $f_1 : \mathcal{L}_A \to \mathcal{L}_B$

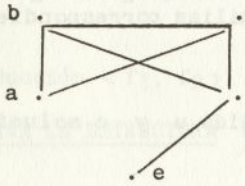
$f_2 : \mathcal{L}_A \times Q_A \to Q_B$

tal que $\langle x_1 k \rangle \in A_d \Leftrightarrow \langle f_1(x), f_2(x,k) \rangle \in B_d$

y $f_1 f_2$ computable tiempo polinómico con Algoritmo determ.

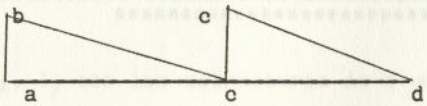
Ejemplo: Consideremos MAX-CLIQUE.=A.

Y:



SOL (y)= {Ø,{a},{b},{c},{d},
{e},{ab}{ac}{bd}{bc}
{cd}{de}{abc}{bcd}
∴ l_y= {1,5,6,2}

Z:



SOL(z)= {Ø, {a}, {b}, {c}, {d},
{e}, {ab}{ae}{be}{ec}
{ed}{cd} {abc} {ecd}
l_z= {1,5,6,2}

Claramente tenemos:

$l_y = l_z \Rightarrow [y]_A = [z]_A$

Vamos a ver relación x≤z

$l_x = \langle 1,5,8,2,1 \rangle$
$l_z = \langle 1,5,6,2 \rangle$
$\Rightarrow l_x \leq l_a$

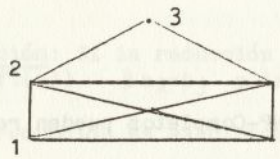
$\therefore [y]_A = [z]_A \leq [x]_A$

Ejemplo: EMPAQUETAMIENTO CONJUNTOS

Entrada: Colección C. de conjuntos finitos; $K \in Z^*$, $Kc|c|$
Propiedad: ¿C contiene K conjuntos mutuamente disjuntos?

Clique $\propto_{pe}$ EMPAQ. CONJUNTOS
$X = G(V,A)$ $|V|=n$
$f_1(x) = \{C_1, C_2, \ldots C_n\}$ $C_i = \{\{i,j\} \quad \{i,j\} \notin A\}$
$f_2(x,k) = k$

Si x



$C_1 = \{\{1,1\} \quad \{1,3\}\}$
$C_2 = \{\{2,2\}\}$
$C_3 = \{\{3,3\}, \{1,3\}, \{3,5\}\}$
$C_4 = \{\{4,4\}\}$
$C_5 = \{\{5,5\} \{5,3\}\}$
$l_{f1(x)} = \langle 1,5,8,5,1 \rangle$

$l_x = \langle 1,5,8,5,1 \rangle$ 0 conj.entre 1conj. 2conj.excl.

$l_x = l_{f1(x)}$

La única manera de que $C_u \cap C_v \neq \emptyset \Rightarrow \{u,v| \in C_u \wedge \{u,v| \in C_v\}$
$\Rightarrow \forall$subgrafo G_i de x, $\forall$arco $\{u,v\} \in C_i'$ los conjuntos entre $C_u C_v = \emptyset$ (y viceversa)
Esto demuestra $\forall$k-clique en x $\exists$k conjuntos disjuntos entre $s_1\ s_2\ \ldots\ s_n$
Si dos cliques-k difieren al menos en 1 vertice $\Rightarrow$ las familias correspondientes de K-conjuntos disjuntos son diferentes (y viceversa)
$\Rightarrow$ El número soluciones de medida K par CLIQUE =
= número soluciones medida K para EMPAQUETAMIENTO
Como $f_2(x,k)=k \Rightarrow$ si en l_x tenemos b soluciones de medida u y c soluciones de medida v, v>u, en $l_{f_1}(x)$
b estará delante de c

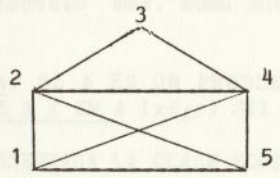
****
* Por lo tanto la reducción preserva estructura *
****

CLIQUE $\propto_{pe}$ RV

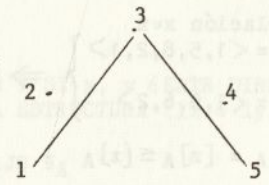
$X \equiv G(V,A)$

$f_1(x)=\bar{x}=\bar{G}(\bar{V},\bar{A})$
$f_2(x,k)=|V|-K$

X



f(x)



$l_x=(1,5,8,5,1)$

R.V. Recubrimiento medida:
1.- {3}
2.- {1,5} {1,3} {3,5} {2,3} {4,3}
3.- {123} {125} {234} {235}
{134} {135} {145} {345}
4.- {1234} {1235} {1345}
{2345} {1245}
5.- {12345}
$\therefore l_{f1}(x)=\{1,5,8,5,1\}$

$\Rightarrow l_x=l_{f_i}(x)$

Esta definición de reducción $\Rightarrow$ Karp
Casi todas las reducciones Karp $\Rightarrow \propto$

Nuestro interes reside en estudiar que problemas NP-Completos pueden reducirse de manera que preserven estructura.

Definición: SI A_d, B_d son problemas decisionales asociados a problemas de Optimización A y B.

La reducción $<f_1, f_2>$ de $A_d \propto B_d$ se dice que PRESERVA EL ORDEN $\propto_{po}$

SI $[x]_A \leq [z]_A \Rightarrow [f_1(x)]_B \leq [f_1(z)]_B$

La reducción $< f_1, f_2 >$ de $A_d \propto B_d$ se dice que

PRESERVA LA ESTRUCTURA $\propto_{pe}$

SI $\forall x \in \mathbb{E}_A , \forall y \in \mathbb{E}_B , f_1(x)=y \Rightarrow l_x = l_y \; * \; l_x = l_{f_1(x)}$

en donde l_x, l_y son las estructuras de x en A y de y en B

Lema SI $\propto$ PRESERVA ESTRUCTURA $\Rightarrow \propto$ PRESERVA ORDEN

Definición: Una reducción $f=\langle f_1,f_2\rangle$ $A_d \propto B_d$ se denomina PARSIMONIOSA SI $\forall x \in \mathbb{E}_A$, $\forall K \in Q_A$

$|\{y \in SOL_A(x) | m_A(y)=K\}| = |\{y \; SOL_B(f_1(x)) | m_B(y)=f_2(x,k)\}|$

Es decir preservan soluciones (el número soluciones).

Como nosotros estamos interesados en considerar el número de soluciones aproximadas:

Una reducción $< f_1, f_2 >: A_d \rightarrow B_d$ se denomina FUERTEMENTE PARSIMONIOSA SI

1) Es parsimoniosa
2) $\forall z \in f_1(\mathbb{E}_A) \; \forall h \in Q_B : |\{y \in SOL_B(z) | m_B(y)=h\}| \neq 0 \Rightarrow \exists x \in \mathbb{E}_A, \exists K \in Q_A$
$[|\{y \; SOL_A(x) | m_A(y)=K\}| \neq 0 \wedge Z = f_1(x) \wedge h=f_2(x,k)]$

Una reducción A^c se dice ESTRICTAMENTE MONOTONICA
Si $\forall x$, $k_1 < K_2$ en $Q_A \Rightarrow f_2(x,K_1) < f_2(x_1,k_2)$ en Q_B

Teorema: Si una reduccion es fuertemente parsimoniosa y estrictamente monotonica $\Rightarrow$ preserva la estructura.

Demostración: Si la reducción es fuertemente parsimoniosa $\Rightarrow \forall a_i \in l_x$ $\exists b_j \in l(f1(x))$ $\ni a_i=b_j$ y la longitud $l(x) \cong$ longitud $l(f1(x))$

Como la reducción es estrictamente monotonica $\Rightarrow$

Dados a_{i_1}, $a_{i_2} \in l_x$ y los correspondientes a_{j_1}, $a_{j_2} \in l_{f1(x)}$

Si $i_1 < i_2 \Rightarrow j_1 < j_2$

$\forall i \quad a_i=b_i \Rightarrow l_x \equiv l_{f_1}(x)$

■

CARACTERIZACION DE REDUCCIONES QUE PRESERVAN ESTRUCTURA

Teorema: Si A es un problema Optimización Convexo

$f = \langle f_1, f_2 \rangle$ $A_b \propto B_b$ tal que:

1) f es Parsimoniosa

2) $f_2(x,K) = \begin{cases} a(x)+K & \text{si A y B son problemas Max-Min.} \\ a(x)-K & \text{en otro caso} \end{cases}$

3) f_2 es tal que $f_2(x, \min\{m_A(y) \mid y \in SOL(x)\}) = \min\{m_B(z) \mid z \in SOL(f_1(x))\}$

ENTONCES f PRESERVA LA ESTRUCTURA

Demostracion: 2 $\Rightarrow$ f es estrictamente monotonica
1+2+3+A_B convexa $\Rightarrow$ f Fuertemente Parsimoniosa

El teorema anterior completa la demostración ■

Vamos a listar algunas reducciones que preservan estructura:

CLIQUE $\propto$ R.V.
CLIQUE $\propto$ Almacenamiento en Conjunto
PARTICION $\propto$ SUMA SUBCONJUNTOS

PROBLEMA CORTE SIMPLE

ENTRADA: GRAFO G(V,A), Pesos $p(a) \in \mathbb{Z}^+$, $\forall a \in A$. $K \in \mathbb{Z}^+$
PROPIEDAD: Existe una particion de $V = V_1 \cup V_2$ $(V_1 \cap V_2 = \emptyset)$ tal que $\sum_{\{u,v\}} p(a_i) \leq K$ en donde $a_i = (u,v)$ $u \in V1, v \in V_2$?

PARTICION GRAFOS

ENTRADA: Grafo $G' \equiv (V', A)$ pesos $p'(v) \in \mathbb{Z}^+$ $\forall v \in V$
y $l(a) \in \mathbb{Z}^+$ $\forall a \in A'$, enteros K, $J \in \mathbb{Z}^+$
PROPIEDAD: Existe una participación de $V \equiv V_1 \cup V_2 \cup \ldots V_m$
$(V_1 \cap V_2 \cap \ldots \cap V_m)$ tal que $\sum_{v \in V_i} p(v) \leq k$ $1 \leq i \leq m$

Y tal que si $A'' \subseteq A$, es el conjunto aristas con sus extremos en diferentes conjuntos V_i
$\Rightarrow \sum_{a \in A''} l(a) \leq J$?

Teorema: CORTE SIMPLE $\not\propto_{pe}$ PARTICION GRAFO

Demostr. X=C.S, Y=P.G.

$l_X = \langle a_1 \ldots a_k \rangle \Rightarrow \sum_{i=1}^{k} a_i = 2^{n-1}$; $l_Y = \langle b_1 \ldots b_k \rangle \Rightarrow \sum_{i=1}^{k} b_i = \frac{1}{2}\binom{2m}{n}$

METODOS PROBABILISTICOS EN EL ANALISIS DE ALGORITMOS MAS EFICIENTES PARA LOS PROBLEMAS NP-COMPLETOS

Dado un grafo $G(V,A)$, un $I \subseteq V$ es un conjunto independiente en G si $\nexists e \in A \ni l = (v_1, v_2)$ y $v_1, v_2 \in I$.

El problema del conjunto independiente máximo consiste en encontrar un conjunto independiente I_M en G, que tenga cardinalidad máxima.

$A\,|I_M| = \alpha(G)$ mle llamaremos el "número estable" de G.

En forma decisional:

Dado $G(V,A)$, $|V| = n$, y una cte. k, $k \in N$, $k \leq v$ encontrar si G tiene un conjunto independ. de cardinalidad k.

ALGORITMO

1.- Construir las familias $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \cdots \mathcal{F}_k$
en donde $\mathcal{F}_i$ es la familia de todos los conjuntos independientes de cardinalidad i.

2.- Si $\mathcal{F}_k \neq \emptyset$, La respuesta será SI.
Si $\mathcal{F}_k = \emptyset$, La respuesta será NO.

Para construir $\mathcal{F}_{i+1}$ partiendo de $\mathcal{F}_i$, verificaremos que para cada $C_j \in \mathcal{F}_i$ y para cada $v_1 \notin C_j$, si $C_j \cup \{v_1\}$ continua siendo independiente ó no lo es.

COMPLEJIDAD DEL ALGORITMO

Cada paso de $\mathcal{F}_i$ a $\mathcal{F}_{i+1}$ requiere

$$|\mathcal{F}_{i+1}|(n-(i+1)(i+1))$$

Por lo tanto complejidad total será

$$\sum_{i=1} |\mathcal{F}|_i \,(n-i)\; i < n^2 \,.\, I(G)$$

en donde $I(G)$ es el número total de conjuntos independientes en G.

Vamos a demostrar el siguiente teorema:

Teorema: (Phan Dinh Diêu)(1977)
Cuando $n \to \infty$, para "casi todos" los grafos G de n vertices, tenemos que:

$$n^{\frac{1}{2}\log n - \log\log n - 1} < I(G) < n^{\frac{1}{2}\log n + \log\log n + 2}$$

Este teorema nos indica que el algoritmo de antes tiene una complejidad de
$O(n^{c}.\log n)$ donde $c>0$

Para casi todas las entradas del problema.

DEMOSTRACION TEOREMA P.D.D.

(a) Si $\bar{X}=\{x_1,x_2,\ldots x_n\}$, $|X|=n$, el número de grafos diferentes que se pueden formar tomando $\bar{X}$ como conjunto de Vertices es $p=2^{C_n^2}=2^{\binom{n}{2}}$.

Denotemos estos p grafos por $G_1\ G_2\ G_3\ \ldots\ G_p$.

(b) Si $0\leq K\leq n$, $K\in\mathbb{N}$, el número de subconjuntos de $\bar{X}$ con cardinalidad K es: $q=C_n^K=\binom{n}{K}$. Llamemos estos q subconjuntos de $\bar{X}$ por $E_1\ E_2\ E_3\ \ldots\ E_q$.

Dados un grafo G, y ctes. $n,K\in N$. Definiremos:

$I_k(G)$: numero conjuntos independientes en G, de cardinalidad K.

$\bar{I}_k(G)$: valor medio de $I_k(n)$ (sobre todos los grafos G de n nodos)

$P_{n,K}(m)$: NUMERO DE GRAFOS G, CON $\bar{X}$ COMO CONJUNTO DE VERTICES Y QUE CONTIENE EXACTAMENTE UN NUMERO m DE conjuntos INDEPENDIENTES DE CARDINALIDAD K.

$\mathcal{E}_{n,k}$: VARIABLE ALEATORIA QUE TOMA EL VALOR m CON PROBABILIDAD $p_{n,k}(m)=\frac{P_{n,k}(m)}{2^{\binom{n}{2}}}$

$M\mathcal{E}_{n,k}$: ESPERANZA MATEMATICA DE $\mathcal{E}_{n,k}$

LEMA 1 $M\,\mathcal{E}_{n,k}=\bar{I}_k(n)$

Demostración: El $\#$ subconjunto de K nodos que se pueden formar con $\bar{X}$ es $q=C_n^K$

$$\Rightarrow \forall m>q,\ P_{n,k}(m)=0 \text{ y } p_{n,k}(m)=0 \Rightarrow M\mathcal{E}_{n,k}=\sum_{m=0}^{\infty} m\,p_{n,k}(m)=$$

$$\sum_{m=1}^{q} m\,p_{n,k}(m)+\sum_{m=q+1}^{\infty} m\,p_{n,k}(m)=\frac{1}{P}\sum_{m=1}^{q} m\,P_{n,k}(m)=\frac{1}{P}\sum_{i=1}^{P} I_K(G_i)=\bar{I}_K(n)$$

■

LEMA 2 $M\mathcal{E}_{n,k}=C_n^K\,2^{-C_K^2}$

Demostración: Para todo grafo G_i $(i=1,2,\ldots,p_j\ p=2^{C_n^2})$, llamemos $I_k(G_i)$ al $\#$ subconjunto de k nodos que son independientes en G_i; y para cada conjunto de K nodos Ej. $(j=1,\ldots,q;\ q=C_n^K)$ llamamos $d_k(Ej)$ al número de grafos de los que es independiente.

Ej. Se cumple que : $\sum_{i=1}^{P} I_k(G_i)=\sum_{j=1}^{q} d_k(Ej)$

Vamos a calcular la parte derecha:

Si Ej independiente en $G_i \Rightarrow G_i$ no tiene aristas que se apoyen en Ej.

Sea W el conjunto de todos los posibles arcos construidos sobre X excepto los que se apoyen en Ej.

Tendremos que $|W| = C_n^2 - C_k^2$

Por tanto cada grafo G_i del que Ej sea independiente correspondera a un subconjunto de W.

$$\Rightarrow d_k(E_j) = |P(W)| = 2^{C_n^2 - C_k^2}$$

$$\Rightarrow M\mathcal{E}n,k = \bar{I}_k(n) = \frac{1}{P}\sum_{i=1}^{P} I_k(G_i) = \frac{1}{P}\sum_{j=1}^{q} d_k(Ej)$$

$$= (2^{-C_n^2}) \sum_{j=1}^{q} 2^{Cn - C_K^2} =$$

$$= 2^{-C_n^2} \cdot 2^{C_n^2 - C_K^2} \sum_{j=1}^{q} 1 = q \cdot 2^{-C_K^2} = C_n^K \cdot 2^{-C_K^2}$$

■

LEMA 3: (CONDICION "CASI TODOS"). Si Π es una función de los p grafos con n nodos en los $\mathcal{N}$, y sea $\bar{\Pi}$ la media de Π. Si llamamos A al número de grafos G_i tales que $\Pi(G_i) > \bar{\Pi} \cdot z$ se cumple que $\frac{A}{P} < \frac{1}{z}$ siendo $z \in R$, $z \geq 0$

Demostración :

$$\bar{\Pi} = \frac{1}{P}\sum_{i=1}^{P} \Pi(G_i) = \frac{1}{P}\left(\sum_{\Pi(G_i) > \bar{\Pi}\cdot z} \Pi(G_i) + \sum_{\Pi(G_i) \leq \bar{\Pi}\cdot z} \Pi(G_i)\right) \geq$$

$$\geq \frac{1}{P}\sum_{\Pi(G_i) > \bar{\Pi}\cdot z} \Pi(G_i) > \frac{1}{P}\sum_{\Pi(G_i) > \bar{\Pi}\cdot z} \bar{\Pi}\cdot z =$$

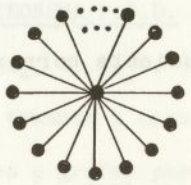
$$= \frac{1}{P} \cdot \bar{\Pi} \cdot z \sum_{\Pi(G_i) > \bar{\Pi}\cdot z} 1 = \frac{1}{P}\bar{\Pi} z \cdot A \Leftrightarrow \frac{1}{z} > \frac{A}{P}$$

■

Teorema 4: Si $h = \lceil 2 \log n+1 \rceil$

Cuando $n \to \infty$, para casi todos los grafos G, con n vertices, todo conjunto independiente I de G, contiene como máximo h vertices. Por lo tanto $\alpha(G)=h$.

Ejm. de un grafo que no cumple el teorema:



$n \to \infty$
$|I| = n-1$

Demostración: Utilizaremos el lema 3 con

$$\Pi(G_1) = I_{h+1}(G_i)$$
$$\overline{\Pi}_{z=h} = \overline{I}_{h+1}(n)$$

tendremos $\frac{A}{P} < \frac{1}{h}$ Cuando $n \to \infty$ A <<<< p

$$\frac{A}{P} < \frac{1}{h} \to 0$$

Tenemos $A = \#$ grafos $\ni I_{h+1}(G) \geq \overline{I}_{h+1}(n)-h$

$\Rightarrow$ Cuando $n \to \infty$, casi todos los grafos cumplen que $I_{h+1}(G) \leq \overline{I}_{h+1}(n).h$

Es decir

$$I_{h+1}(G) \leq \overline{I}_{h+1}(n).h = C_n^{h+1} \cdot 2^{-C_{h+1}^2} \cdot h$$

$$= \frac{n!}{(h+1)!(n-h-1)!} \cdot \frac{1}{2^{\frac{(h+1)h}{2}}} \cdot h$$

$$= \frac{h}{(h+1)!} \cdot \frac{n(n+-1) \ldots (n-h+1)(n-h)}{2^{\frac{(h+1)h}{2}}} = \frac{h}{(h+1)!} \cdot \frac{A}{B}$$

donde: $A=n(n-1) \ldots (n-h+1)(n-h) \leq n. \ldots . n=n^{h+1}$

y $B=2^{\frac{(h+1)h}{2}} \geq n^{h+1}$ (*)

(*) Sacando log queda: $\frac{(h+1)}{2} \cdot h \geq (h+1) \log n$

$\Rightarrow \frac{h}{2} \geq \log n, \lceil 2 \log n+1 \rceil \geq 2 \log n$

Substituyendo

$$I_{h+1}(G) \leq \frac{h}{(h+1)!} \quad \frac{A}{B} \leq \frac{h}{(h+1)!} \cdot \frac{n^{h+1}}{N^{h+1}} = \frac{h}{(h+1)!}$$

$$= \frac{h}{(h+1)h!} \leq \frac{h}{h.h!} = \frac{1}{h!}$$

Cuando $n \to \infty$, casi todos los grafos cumplen que:

$$I_{h+1}(G) \leq \frac{1}{h!}$$

■

Teorema: Cuando $n \to \infty$, para casi todos los grafos G, con $V \equiv \bar{X}$, $|\bar{X}| = n$, el número de conjuntos independientes I(G) satisface:

$$I(G) < n^{\frac{1}{2}\log n + \log\log n + 2}$$

Demostración:

* $\forall k < n$ se tiene $\dfrac{I_{k+1}(n)}{I_k(n)} = \dfrac{n-K}{2^{k(k+1)}}$

en efecto $\dfrac{\bar{I}_{k+1}(n)}{\bar{I}_k(n)} = \dfrac{C_n^{k+1} \cdot 2^{-C_{k+1}^2}}{C_n^k \cdot 2^{-C_k^2}} =$

Lema 2 $\dfrac{n(n-1)(n-2)\ldots(n-k+1)(n-k)}{(k+1)!} \cdot 2^{-\frac{(k+1).K}{2}}$

$$= \frac{n(n-1)\ldots(n-k+1)}{K!} \cdot \frac{2}{2^{-\frac{k\cdot(k-1)}{2}}}$$

$$= \frac{n-k}{k+1} \cdot \frac{2^{-\frac{k^2}{2}-\frac{k}{2}}}{2^{-\frac{k^2}{2}+\frac{k}{2}}} = \frac{n-k}{2^{k(k+1)}}$$

** Si $K \leq \log n - 1$ se cumple $\dfrac{\bar{I}_{k+1}(n)}{I_k(n)} \geq \dfrac{2(n-\log n+1)}{n.\log n} > \dfrac{1}{\log n}$ (*)

Por (*) $\dfrac{\bar{I}_{k+1}}{I_k} = \dfrac{n-k}{2^{k(k+1)}} \geq \dfrac{n-\log n+1}{2^{\log n-1}(\log n-1+1)}$

$$= \frac{2(n-\log n+1)}{n \log n} = \frac{1}{\log n} \cdot \frac{2n-2\log n+2}{n} > \frac{1}{\log n}$$

ya que $2n-2\log n+2 > n \Leftrightarrow n+2 > 2\log n$
lo que es cierto para n suf. grande.

Para n suficientemente grande si K>logn-1 se cumple $\frac{\bar{I}_{k+1}}{\bar{I}_k} < \frac{2}{\log n}$

De * : $\frac{\bar{I}_{k+1}}{\bar{I}_k} \quad \frac{n-k}{2^k(k+1)} < \frac{n-\log n-1}{2^{\log n-1}(\log n-1+1)}$

$= \frac{2}{\log n} \cdot \frac{n-\log n-1}{n} < \frac{2}{\log n}$

Tomando $h = \overline{\log n}$ **se cumple**

$\bar{I}_k(n) < (\log n)^{h-k} \cdot \bar{I}_h(n)$ si $h > k$

$\bar{I}_k(n) < \left(2/\log n\right)^{k-h} \cdot \bar{I}_h(n)$ si $h \leq k$

en efecto

$$\begin{cases} h>k \Rightarrow k \leq \log n-1 \Rightarrow \bar{I}_k \leq I_{k+1} \cdot \log n < \bar{I}_{k+2}(\log n)^2 \quad \dots \quad I_{k+(h-k)}(\log n)^{n-k} \\ h \leq k \Rightarrow k > \log n-1 \Rightarrow \bar{I}_k < \bar{I}_{k-1} \cdot \frac{2}{\log n} < \bar{I}_{k-2}\left(\frac{2}{\log n}\right)^2 < \dots < I_{k-(k-h)}\left(\frac{2}{\log n}\right)^{k-h} \end{cases}$$

$$\sum_{K<h} (\log n)^{h-k} \leq n \log \log n$$

en efecto $\sum_{k<h} (\log n)^{h-k} \leq \log n \; h \sum_{K<h} \left(\frac{1}{\log n}\right)^k$ (sumando la prog. geometrica)

$$\leq \log n \log n \; \frac{\left(\frac{1}{\log n}\right)^h \left(\frac{1}{\log n}\right) - \left(\frac{1}{\log n}\right)}{\left(\frac{1}{\log n}\right)-1} =$$

$$= n \log \log n \; \frac{\left(\frac{1}{\log n}\right)^h - 1}{1-\log n} = n \log\log n \cdot \frac{(\log n)^{h-1}}{(\log n)^h (\log n-1)} \leq n^{\log\log n}$$

$$\sum_{K \geq h} \left(\frac{2}{\log n}\right)^{K-h} \leq 2$$

en efecto $\sum_{k\geq h} \left(\frac{2}{\log n}\right)^{k-h} = \left(\frac{2}{\log n}\right)^{-h} \sum_{k\geq n} \left(\frac{2}{\log n}\right)^{k} \leq$ ($\sum$ prog. geometrica)

$$\leq \left(\frac{2}{\log n}\right)^{-h} \frac{\left(\frac{2}{\log n}\right)^{n}\left(\frac{2}{\log n}\right) - \left(\frac{2}{\log n}\right)^{h}}{\left(\frac{2}{\log n}\right) - 1} \leq$$

$$\leq \left(\frac{2}{\log n}\right) \cdot \frac{2}{\log n - 1 - \left(\frac{2}{\log n}\right)} = \frac{\log n}{\log n - 2} \leq$$

ya que $\log n \leq 2 \log n - 4 \Leftrightarrow \log n \geq 4$

****
*** * Para n suf. grande, como $\log n - 1 < h \leq \log n$, *
*** * $\Rightarrow C_n^h \leq n^{\log n - 1}$ *
****

en efecto $C_n^h = \frac{n(n-1)\ldots(h-h+1)}{h!} \leq \frac{n,n\ldots n}{h!} = \frac{n^h}{h!} \leq$

$\leq \frac{n^{\log}}{2^{\log n}} = n^{\log n - 1}$; ya que para n grande $(\sqrt{\log n})! \geq 2^{\log n}$

****
*** * Como $\log n - 1 < h \leq \log n$, para n sufic. grande *
*** * $C_n^2 > \log n \ (\frac{1}{2}\log n - 2)$ *
****

$$C_n^2 = \frac{h(h-1)}{2} \quad \frac{(\log n - 1)(\log n - 2)}{2} = \frac{(\log n)^2 - 3\log n + 2}{2} >$$

$$> \frac{(\log n)^2 - 4\log n}{2} = \log n \left(\frac{1}{2}\log n - 2\right)$$

Si $\bar{I}(n)$ es la media de I(G) para $\forall$G de n nodos, si n es sufientemente grande:

$$\bar{I}(n) = \frac{1}{2^{\bar{C}_n^2}} \sum_{i=1}^{P} \bar{I}(G_i) = \sum_{k=1}^{n} \bar{I}_k(n)$$

$$= \sum_{k<h} \bar{I}_k + \sum_{k\geq h} I_k \underset{**}{\leq} \bar{I}_n \left(\sum_{k<h} (\log n)^{h-k} + \sum_{k\geq h} \left(\frac{2}{\log n}\right)^{k-h} \right)$$

$$\overset{***}{\leq} \bar{I}_h (n^{\log \log n + 2}) = C_n^h 2^{-C_n^2} (n^{\log\log n + 2}) < n^{\log - 1} (2^{\log n}) 2^{-\frac{1}{2}\log n} (n^{\log\log n + 2}) =$$

$$= n^{\frac{1}{2}\log n + 1} \cdot (n^{\log\log n + 2}) \underset{***}{\underset{**}{+}} n$$

$$< n^{\frac{\log n + 1}{2}} \left(2n^{\log\log n}\right) < 2 \cdot n^{\frac{1}{2}\log n + \log\log n + 1}$$

Utilizando lema 3 con $\pi = I$
$\eta = I$:
$\ell = \frac{n}{2}$

LA FRACCION DE GRAFOS QUE CUMPLEN

$$I(G) > \bar{I}(n) \cdot \binom{n}{\frac{n}{2}}$$

ES MENOR QUE $\frac{2}{n}$

CUANDO $n \to \infty$ CASI TODOS GRAFOS

$$I(G) \leq \bar{I}(n) \cdot \frac{n}{\frac{n}{2}} \quad n^{\frac{1}{2}\log n + \log\log n + 2}$$

■

ALGUNAS CONSECUENCIAS DEL TEOREMA 4

Corolario 1 Si $h = \lceil 2 \log n + 1 \rceil$. Cuando $n \to \infty$, para casi todos los grafos G con n vertices, cada recubrimiento de vertices (vertex Cover) de G, contiene un minimo de n-h.

Corolario 2 Si $\chi(G)$ es el número cromático de G. Cuando $n \to \infty$, para casi todos los grafos G de n nodos, $\chi(G) \geq \frac{n}{\lceil 2 \log n + 1 \rceil}$

Trivial si consideramos que

$$\alpha(G) \cdot \chi(G) \geq n. \quad \text{C.Berge.}$$

Corolario 3 Cuando $n \to \infty$, para casi todos los grafos G de n vertices, cada clique de G contiene un máximo de $h = \lceil 2 \log n + 1 \rceil$ vertices.

BIBLIOGRAFIA

1.- Angluin, D.; L.G.Valiant: "Fast probabilistic algorithms for Hamiltoniam circuits and matchings". Proc. 8 th STOC, 1-30, ACM, 1977

2.- Berenguer, X; J.Diaz: "The weighted Sperner´s set Problem" Proc, 9 th MFCS, Ed. P.Dembinski, 137-142, Springer-Verlag, 1980

3.- Cook, S.A.: "The Complexity of Theorem-proving procedures" Proc. 3rd STOC, 151-158, ACM, 1971

4.- Diaz, J.: "Complejidad Problemas Combinatorios", Questiió, 2,1,35-51, 1978.

5.- Edmonds, J.:"Path, trees and flowers" Canod, J.Math. 17, 449-469, 1965.

6.- Erdos, P.; J. Spencer: "Probalitistic Methods in Combinatorics" AP, 1974

7.- Garey, M.; S.J.Johson: "Computers and Intractability: A guide to the theory of NP-Completeness", Ed. Freeman, 1979

8.- Hopcroft, J.E.; J.D.Ullman: "Introduction to Automata Theory, languages and Computation"; Addison-Westley, 1979

9.- Johnson, D.S.: "Approximation algorithms for combinatorial problems". J. Compt. System Science, 9, 256-278, 1974

10.- Karp, R.: "Reducibility among combinatorial problems" in Complexity Computer Computations, Ed. Miller and Thatcher, Plenun Press, 85-104, 1972

11.- Karp, R.: "On the Computational Complexity of Combinatorial Problems"; Networks, 5,45-48, 1975

12.- Karp, R.: "Some Probabilistic Analysis of Some Combinatorial Research Algorithms" en Algorithms and Complexity, Edited J. Tranb, AP, 1976

13.- Ladner, "On the structure of Polynomial time Reducibility" J. Assoc. Comput. Mach. 22, 155-171, 1975

14.- Penttonen, M.: "Complejidad de los algoritmos" Boletin CCUCM, 34, 15-44, 1979.

15.- Phan Dinh Dieu: "Some investigations of finite graphs and their applications", Proc. IV FCT, Ed.L.Budach, 343-348, Akademie-Verlag, 1979

16.- Turing, A.M.: "On computable numbers with an application to the entescheidungs problem", Proc. London Math, Soc. 42, 230-265, 1936 y 43, 544-546, 1937