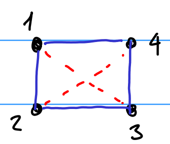
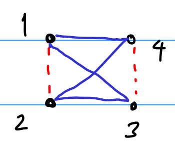
El problema del agente viajero
(Traveling Salesman Problem TSP)

(1832, Hamilton, Kirkman 1930)

Problema: Un vendedor de enciclopedias debe recorrer n ciudades y regresar a casa visitando cada ciudad exáctamente una vez. Sabemos los costos c_{ij} de viajar desde i hasta j. Cuál es el itinerario más económico posible?





$1423 = 4231 = 2314 = 3142$

y $1423 = 3241$

Cuántos caminos posibles hay? R: $\dfrac{n!}{2n} = \dfrac{(n-1)!}{2}$

ENORME!!

El algoritmo de Held-Karp, construido mediante pogramación dinámica nos dará un algoritmo $O(n^2 2^n)$, exponencial, pero MUUCHO mejor que fuerza bruta.

Planteamos el problema como uno de optimización rompiendo el ciclo para permitir una formulación recursiva.

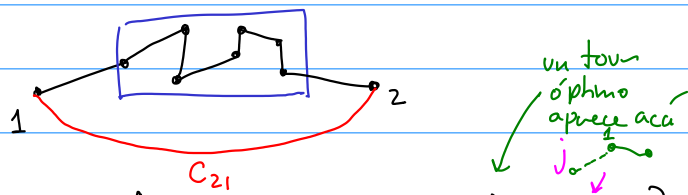
Def: Fijamos ciudad de inicio 1. y para $S \subseteq \{2,\dots,n\}$ y $e \neq 1$, $e \notin S$ definimos

$$g(S,e) = \min \{ c(T) :$$

- T es un camino
- T inicia en 1
- T termina en e
- T visita toda ciudad de S exáctamente una vez
- T no visita ninguna ciudad más (fuera de $S \cup \{1\} \cup \{e\}$).

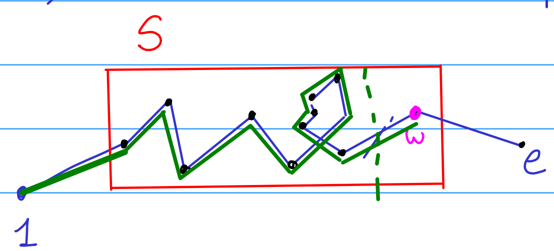
Si supiéramos $g(S,e)$ resolveríamos el TSP así:

$$g(\{3,\dots,n\},2) + C_{21}$$



luego $\mathrm{TSP} = \max\limits_{2\le j\le n} \left\{ g(\{2,\dots,j-1,j+1,\dots,n\}, j) + C_{j1} \right\}$

Ahora, sea T^* un camino óptimo para $g(S,e)$



Si w es el último vértice del camino dentro de S

entonces $\ell(T^*) = g(S\setminus\{w\}, w) + \ell(w,e)$

porque T' cumple:

- T' inicia en e
- T' termina en w
- Recorre solo las ciudades de $S\setminus\{w\}$ $\;(= S')$ cada una una vez
- No pasa por ningún otro vértice.

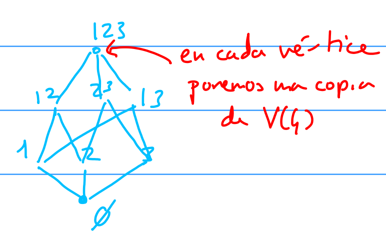
$\ell(T') \geq g(S\setminus\{w\}, w)$ y hay igualdad pues de lo contrario T^* no sería óptimo. Hemos demostrado

Teorema: Ecuación de Bellman para el TSP

$$g(S,e) = \min_{\substack{w\in S \\ (w,e)\in E(G)}} \left\{ g(S\setminus\{w\}, w) + C_{we} \right\}$$

Cuando $S=\emptyset$ $\quad g(\emptyset, w) = C_{1w}$ y vamos subiendo el cardinal de los subconjuntos.

ESPACIO DE ESTADOS:



Cuál es la complejidad? Dado e : $g(\underline{S}, e)$

$$\sum_{k=1}^{n}\Big[\underbrace{\binom{n-1}{k}}_{\text{Subconjunto } S}\ \underbrace{(n-1-k)}_{\substack{\text{escogemos}\\ \text{posibles para}\\ \text{cada dado } S}}\Big]$$

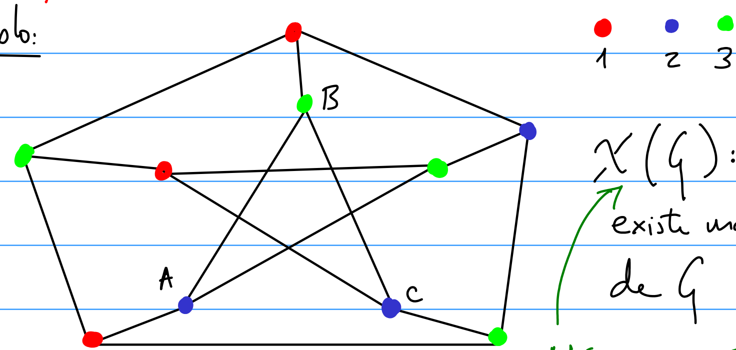
así que en total tenemos $n\left[\sum_{k=1}^{n}\binom{n-1}{k}(n-1-k)\right] = O(n^2 2^n)$

Es necesario enfatizar el ENORME avance

$$\wedge$$
$$\wedge$$
$$O(n!)$$

PROBLEMA: Una coloración de un grafo finito G con k colores es una función $f: V(G) \longrightarrow [k] := \{1, \dots, k\}$ con $f(u) \neq f(v)$ $\forall (u,v) \in E(G)$

Ejemplo:



$\chi(G) := \min\{k :$ existe una k-coloración de $G\}$

Número cromático

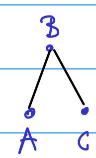
Cómo calcular $\chi(G)$? PROGRAMACIÓN DINÁMICA.

ALGORITMO DE LAWLER

Para cada $U \subseteq V(G)$ define el grafo inducido por U, $\langle U \rangle$ como:

$$V(\langle U \rangle) = U$$
$$E(\langle U \rangle) = \{(a,b) \in E(G) : a \in U, b \in U\}.$$

Ejemplo: $\langle A,B,C \rangle_G =$



Definimos, para $U \subseteq V$:

$$h(U) := \chi(\langle U \rangle)$$

Qué propiedades tiene una coloración óptima f^* de G?
con k colores

Entre todas las coloraciones óptimas escogemos una que tenga el color más grande de tamaño máximo y digamos que es el conjunto $A = f^{-1}(1)$

A es independiente y también maximal

porque de lo contrario podríamos encontrar otro vértice x independiente de A y colorearlo de azul contradiciendo la maximalidad.

Si $\langle V(G)\setminus A\rangle$ pudiera ser coloreado con $\leq k-2$ colores podríamos colorear A con $\leq k-1$ $\boxtimes$ luego

$$k = 1 + \chi(\langle V(G)\setminus A\rangle)$$

para algún subconjunto maximal independiente A.

ECUACIÓN DE BELLMAN

$$\chi(G) = \min_{A \subseteq V(G)} \{ 1 + \chi(\langle V(G)\setminus A\rangle)\}$$

[A independiente y maximal.]

Hay $\leq 3^{\frac{t}{3}}$ subconjuntos max indep. con t vértices

?

$O(2.44\ldots^n) << O(n^n)$

Como enumerar subconjuntos independientes maximales?

Siguiendo a Papadimitriou–Yannakakis 1988 lo hacemos mediante una combinación de DFS y Programación dinámica.

En resumen, hacemos los siguientes pasos:

(1) Enumeramos sobre los subconjuntos $S \subseteq V(G)$ en orden de cardinal creciente, empezando con $\chi(\{q\}) = 1 \quad \forall q \in V(G)$

Ahora, dado $S \subseteq V(G)$ de cardinal t

(i) Enumeramos TODOS los subconjuntos independientes maximales de $\langle S \rangle$ mediante el algo de Papadim.-Yannak. * que veremos a continuación creando una lista $\mathcal{L} = \{A_1, \ldots, A_M\}$

(M puede ser muy grande, $3^{\frac{|S|}{3}}$)

y usamos la ecuación de Bellman

$$\chi(\langle S \rangle) = \min_{A_i \in \mathcal{L}} \{ 1 + \underbrace{\chi(\langle S \setminus A_i \rangle)}_{} \}$$

estos valores ya se conocen por inducción.

Al final, el número cromático de G es

$$\chi(G) = \chi(\langle G \rangle)$$

Se puede demostrar que este algoritmo requiere $O(2.44^n)$, exponencial pero mucho mejor que fuerza bruta $O(n^n)$

* Cómo enumerar TODOS los subconjuntos independientes maximales de un grafo?

[Papadimitriou - Yannakakis]

El siguiente algoritmo los enumera todos en orden lexicográfico ($T_1, T_2 \subseteq [n]$

$T_1 \leq T_2$ si el primer índice en el que difieren está en T_1)

Lex

Ejemplo: $\overbrace{1\ 4\ 5\ 9}^{T_1} \overset{LEX}{\leq} \overbrace{1\ 4\ 7\ 8\ 9}$

Índice más pequeño en el que difieren.

Algoritmo: Q ← Priority queue con keys $\subseteq [n]$ ordenadas Lex.

S^* = Primer conjunto independiente maximal de $V(G)$ en orden lex

Q = heap(S^*), $L = [\,]$

while Q not empty:

  $S = Q$.get_minimum()

  L.append(S)

  for each j in $V(G)$ adyacent to a vertex $i \in S$ with $i < j$

    $S_j := S \cap \{1, .., j\}$

    If $S_j - \Gamma(j) \cup \{j\}$ is max indep. in $\langle \{1, .., j\} \rangle$ then

    neighbors of j

      $T :=$ lex-first maximal indep set containing $S_j - \Gamma(j) \cup \{j\}$

      Q.insert(T)

El algoritmo requiere que sepamos encontrar, dado un subconjunto independiente I, un conjunto independiente T con $T \supseteq I$ que sea el primero en orden lex. Esto es muy fácil:

Iniciamos con $T = I$ y
Leemos los vértices en orden $1, 2, \ldots, n$.
Agregamos j a T ssi al hacerlo T sigue siendo independiente.

Ejercicio: Demuestre que este método funciona siempre.