UNIDAD II. ESTRUCTURA DE DATOS AVANZADAS (GRAFOS NO DIRIGIDOS)

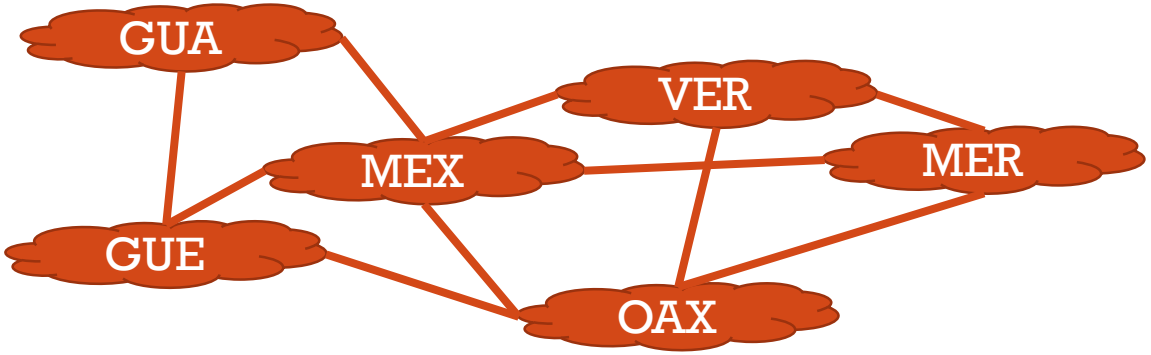
Francisco J. Hernández López

fcoj23@cimat.mx

GRAFOS NO DIRIGIDOS

- Sus aristas son pares de vértices no ordenados, es decir si hay un camino del vértice i al vértice j, será exactamente el mismo camino del vértice j al vértice i
- Modelan relaciones simétricas entre diferentes objetos
- Para su representación, se pueden utilizar:
  - Una matriz de adyacencia (simétrica)
  - Una lista de adyacencia

EJEMPLO DE UN GRAFO NO DIRIGIDO



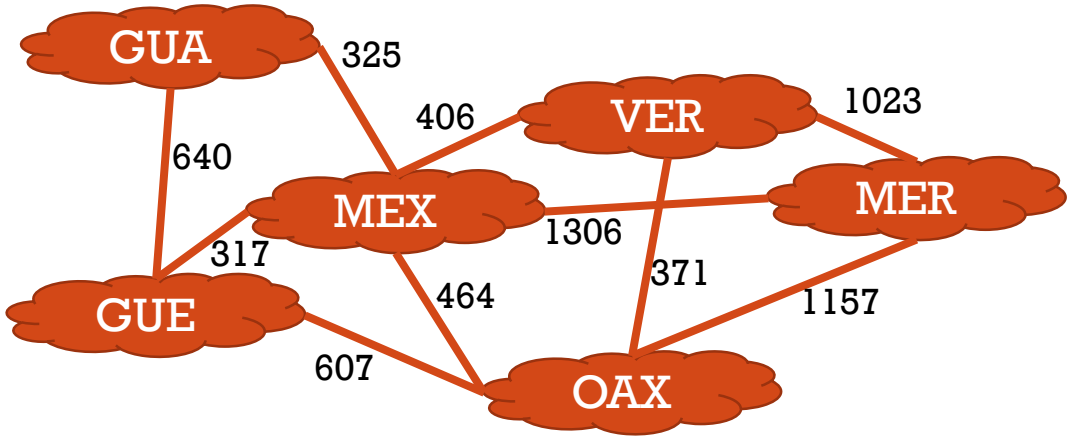
	GUA	GUE	MEX	OAX	VER	MER
GUA	0	1	1	0	0	0
GUE	1	0	1	1	0	0
MEX	1	1	0	1	1	1
OAX	0	1	1	0	1	1
VER	0	0	1	1	0	1
MER	0	0	1	1	1	0

MINIMUM SPANNING TREE

- O también conocido como:
  - Árbol de expansión mínima
  - Árbol expandido mínimo
  - Árbol abarcador de costo mínimo
  - Árbol recubridor mínimo
- Sea $G = (V, A)$ un grafo conexo en donde cada arista (u, v) tiene asociado un peso o costo $c(u, v)$, entonces un árbol de expansión mínima del grafo G se define como:
  - Un árbol libre que conecta todos los vértices de V por medio de las aristas que suman el menor costo

Árbol de expansión mínima

- Ejemplo: Diseño de redes de comunicación
  - Los vértices representan las ciudades
  - Las aristas representan los posibles canales de comunicación
  - El costo asociado a cada arista representa el costo de comunicar una ciudad con otra (tiempo, dinero, medios, etc.)



Un árbol de expansión mínima representará la red de comunicación que conecta a todas las ciudades a un costo mínimo.

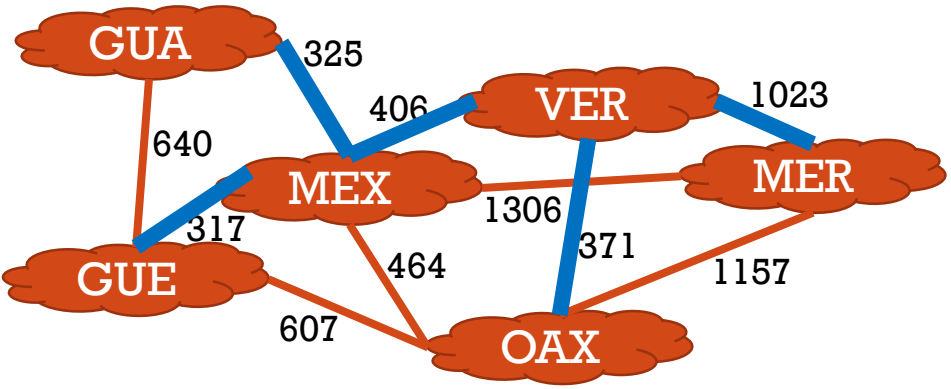
ALGORITMO DE PRIM

- Encuentra el árbol de expansión mínima de un grafo
- Tiempo de ejecución
  - Matriz de adyacencia: $O(N^2)$
  - Lista de adyacencia: $O(A * \log N)$

- Consideraciones:
  - $V = \{1,2, \ldots, n\}$ → Conjunto de vértices
  - U → Subconjunto propio de V, iniciando con $U = \{1\}$ el primer vértice de V
  - L → Lista de aristas que se va formando con las aristas de menor costo seleccionadas. Inicialmente $L = \{\emptyset\}$

ALGORITMO DE PRIM

1. Mientras $(V \neq U)$ repetir
   1. Elegir una arista $(u, v) \in A(G)$ tal que su costo sea mínimo, siendo $u \in U$ y $v \in (V - U)$
   2. Agregar la arista (u, v) a L
   3. Agregar el vértice v a U
2. Fin del ciclo



		1	2	3	4	5	6
		MER	OAX	VER	MEX	GUE	GUA
1	MER	--	1157	1023	1306	--	--
2	OAX	1157	--	371	464	607	--
3	VER	1023	371	--	406	--	--
4	MEX	1306	464	406	--	317	325
5	GUE	--	607	--	317	--	640
6	GUA	--	--	--	325	640	--

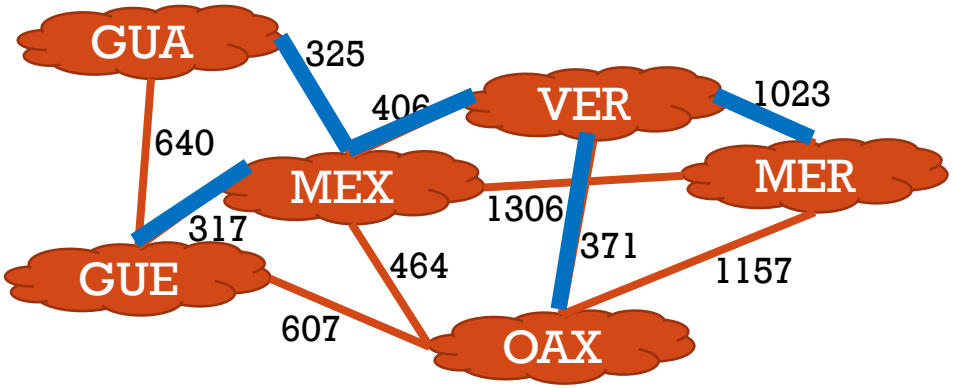
ALGORITMO DE KRUSKAL

- Al igual que el algoritmo de Prim, encuentra el árbol de expansión mínima de un grafo
- Tiempo de ejecución
  - $O(A * \log A)$, donde A es el número de aristas

- Consideraciones:
  - L → Conjunto formado por las aristas y sus respectivos costos
  - P → Representa las particiones generadas a partir de V.
  - Inicialmente $P = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$

ALGORITMO DE KRUSKAL

- Mientras haya vértices en P que pertenezcan a particiones distintas, repetir
  - De L seleccionar la arista (u, v) que tenga el menor costo
  - Si (u y v se encuentran en particiones diferentes) entonces
    - Unir las particiones a las cuales pertenecen u y v
  - Fin si
- Fin del ciclo



APLICACIONES

Araña web (web crawling): Programa que inspecciona las páginas web de forma automática:

- Crear el índice de una maquina de búsqueda
- Analizar los enlaces de un sitio para buscar links rotos
- Recolectar información de un cierto tipo: precios, descargas, etc.

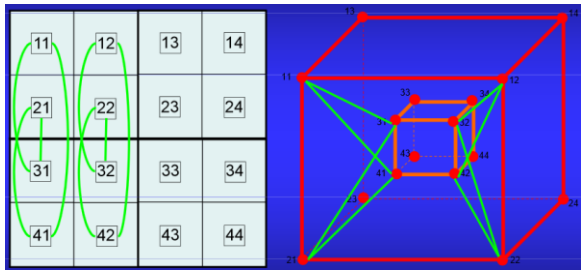
Red social (social networking): Forma de representar una estructura social. Si dos elementos de un conjunto de actores (personas, organizaciones, etc.) están relacionadas o conectadas deacuerdo a algún criterio (relación profesional, perentezco, negocios, etc.), entonces se construye una línea que conecta a dichos elementos. Ej: Facebook, Friend-finder, etc.

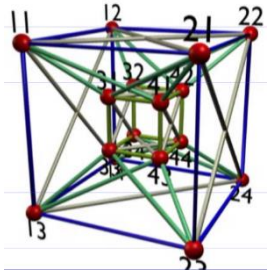
Transmisión a través de la red (network broadcast): Transmisión de datos (video, imágenes, audio, etc.) a través de la red. Ej: TV o radio en la red.

APLICACIONES

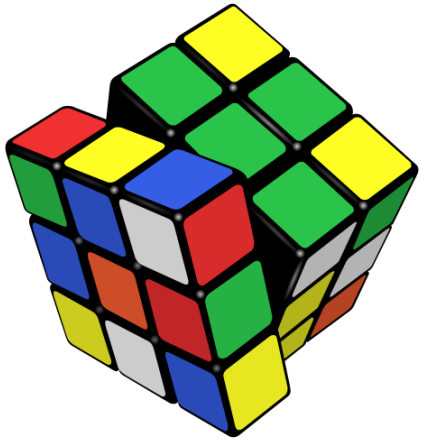
Recolector de basura (garbage collection): Mecanismo de gestión de memoria implementado en algunos lenguajes de programación de tipo interpretado o script (Python, Java, C#, Ruby, etc.)

Resolver juegos de rompecabezas (puzzles games):



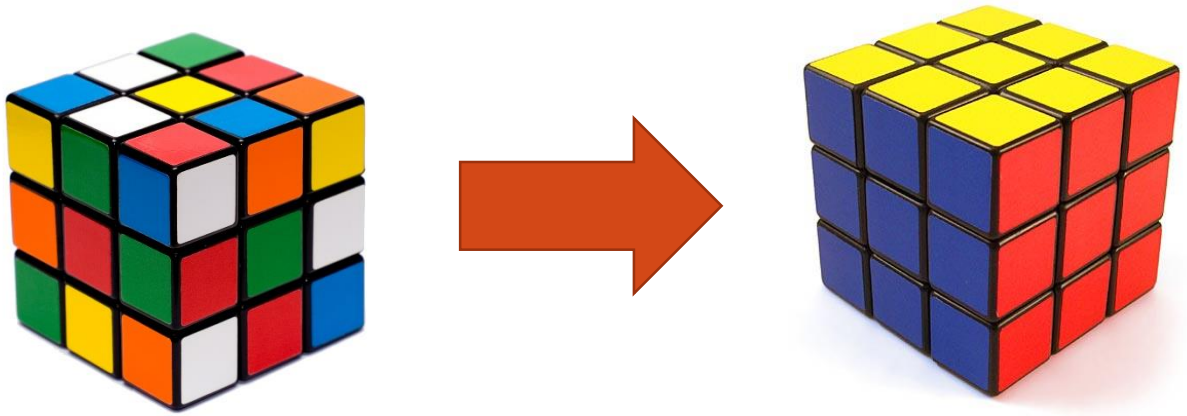


http://www.unizar.es/ttm/2010-11/Colores_sudoku_Huesca.pdf



MÉTODOS DE BÚSQUEDA

- Existen diversas aplicaciones en las cuales se pueden definir un estado inicial y entonces plantear el problema como la búsqueda de un estado final



  - Búsqueda a lo ancho (Breadth-first)
  - Búsqueda en profundidad (Depth-first)

BÚSQUEDA A LO ANCHO

- Los nodos se expanden en el orden en el que han sido generados
- Basado en la estructura FIFO (Cola)

- Consideraciones:
  - Se avanza por niveles hasta encontrar el estado meta
  - Utiliza dos listas auxiliares:
    - ABIERTO → Se utiliza para almacenar los nodos que están pendientes de ser expandidos. Cada nuevo nodo generado se coloca al final de la lista
    - CERRADO → Se colocan los nodos que ya han sido expandidos

Nota: Expandir a un nodo es calcular todos los sucesores del nodo

BÚSQUEDA A LO ANCHO (ALGORITMO)

1. Insertar el nodo inicial en la lista ABIERTO
2. Mientras (ABIERTO no esté vacía) y (no se haya alcanzado el estado final) repetir
   1. Quitar el primer nodo X de ABIERTO
   2. Si (el nodo X no se encuentra en CERRADO) entonces
      1. Poner el nodo X en la lista CERRADO
      2. Expandir el nodo X obteniendo todos sus sucesores
      3. Si (hay sucesores y no son el estado final) entonces
         1. Almacenar al final de ABIERTO y proveer apuntadores para regresar a X
      4. Fin si
   3. Fin si
3. Fin ciclo
4. Si (se generó el estado final) entonces
   1. Éxito: Desplegar la trayectoria del estado inicial al estado final
5. Si no
   1. Fracaso: No se alcanzó el estado final
6. Fin si

Complejidad del método:
$O(b^d)$, con:
b → factor de ramificación
d → profundidad del árbol

Tiempo de Ejecución:
$2A$ → Grafo no dirigido
A → Grafo dirigido

EJEMPLO

BÚSQUEDA EN PROFUNDIDAD

- Se expande el nodo más recientemente generado
- Basado en la estructura LIFO (Pila)

- Consideraciones:
  - La profundidad del nodo inicial es 1
  - La profundidad de un nodo no inicial es igual a uno más la profundidad de su padre
  - ABIERTO y CERRADO son dos listas simplemente ligadas
  - Se establece un límite de profundidad permitido P
  - Si se llega al límite establecido sin llegar al estado final (después de haber hecho backtracking), entonces se considera que el problema no tiene solución

BÚSQUEDA EN PROFUNDIDAD (ALGORITMO)

1. Insertar el nodo inicial en la lista llamada ABIERTO
2. Mientras (ABIERTO tenga elementos) y (no se haya llegado al estado final), repetir
    1. Quitar el primer nodo N de ABIERTO
    2. Si (N no está en CERRADO) y (su profundidad es $\leq P$) entonces
        1. Insertar el nodo N en la lista CERRADO
        2. Expandir el nodo N obteniendo todos sus sucesores
        3. Si (hay sucesores y no son el estado final) entonces
            1. Almacenarlos al inicio de la lista ABIERTO
        4. Fin si
    3. Fin si
3. Fin ciclo
4. Si (alguno de los nodos generados es el estado final) entonces
    1. Éxito: Desplegar la trayectoria desde el estado inicial al estado final
5. Si no
    1. Fracaso: No se encontró el estado final
6. Fin si

Complejidad del método:
$O(V + A)$ → #Vértices + #Aristas

Tiempo de Ejecución:
$2A$ → Grafo no dirigido
A → Grafo dirigido

EJEMPLO