

TC2038. Análisis y diseño de Algoritmos A

M.C. Ramona Fuentes Valdéz

rfuentes@tec.mx

87

Grafos

Problema del vendedor ambulante Traveling Salesman Problem (TSP)

Dada una colección de ciudades y el costo de viaje entre cada par de ellas, el problema del vendedor ambulante (*traveling salesman problem* o *TSP*), es encontrar la forma más económica de visitar todas las ciudades y regresar a su punto de partida.

- La simplicidad del enunciado del problema es engañosa: el *TSP* es uno de los problemas más intensamente estudiados en la matemática computacional y, sin embargo, no se conoce ningún método de solución eficaz para el caso general.
- De hecho, la resolución del *TSP* resolvería el problema *P* vs *NP* y obtendría un premio de \$ 1,000,000 otorgado por el *Clay Mathematics Institute* (<http://www.claymath.org/>).

Definición

Dado

- Un conjunto de ciudades $\{c_1, c_2, \dots, c_N\}$.
- Para cada par de ciudades $\{c_i, c_j\}$ hay una distancia $d(c_i, c_j)$.

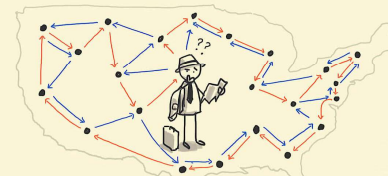
Encontrar:

La permutación $\pi: \{1, 2, \dots, N\} \rightarrow \{1, 2, \dots, N\}$ que minimiza

$$\sum_{i=1}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(N)}, c_{\pi(1)})$$

THE TRAVELLING SALESMAN PROBLEM

WHAT'S THE SHORTEST ROUTE TO VISIT ALL LOCATIONS AND RETURN?





ADDING MORE STOPS TAKES LONGER AND LONGER. AND LONGER TO FIGURE IT OUT

88

Grafos

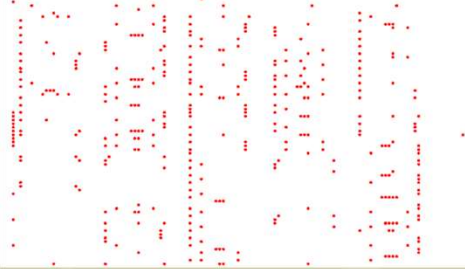
Problema del vendedor ambulante Traveling Salesman Problem (TSP)







Robert Bosch, February 2009
mona-lisa100K.tsp
New! \$1000 Prize Offered

Mona Lisa TSP Challenge
<https://www.math.uwaterloo.ca/tsp/data/ml/monalisa.html>





monuments.




How to visit 1.33 billion stars.

ITESM, Dr. Gildardo Sánchez Ante

89

Grafos

Problema del vendedor ambulante Traveling Salesman Problem (TSP)



Cristalografía de rayos X

- **Ciudades:** orientaciones de un cristal
- **Distancias:** tiempo para que los motores giren el cristal de una orientación a la otra.

Compresión de video de alta definición

- **Ciudades:** vectores binarios de longitud 64 que identifican los sumandos para una función en particular.
- **Distancias:** distancia de Hamming (*la cantidad de términos que se deben sumar/restar para obtener la siguiente suma*).

¿Qué tan complicado es el algoritmo TSP?

- **NP Completo** para las aplicaciones mencionadas:
- **Cantidad de caminos posibles:**
 - [Karp, 1972]
 - [Papadimitriou & Steiglitz, 1976]
 - [Garey, Graham, & Johnson, 1976]
 - [Papadimitriou & Kanellakis, 1978]
 - ...

$$N! = 1 \times 2 \times 3 \times \dots \times (N-1) \times N = \Theta(2^{N \log N})$$

10! = 3,628,200

20! ~ 2.43 × 10¹⁸ (2.43 quadrillion)

La cristalografía es la ciencia que estudia los cristales. La mayoría de los minerales, compuestos orgánicos y numerosos materiales, adoptan estructuras cristalinas cuando se han producido las condiciones favorables.

ITESM, Dr. Gildardo Sánchez Ante

90

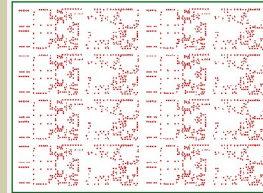
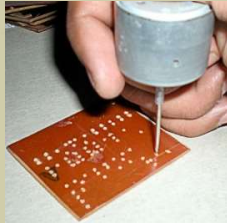
Grafos

Problema del vendedor ambulante Traveling Salesman Problem (TSP)

Aplicaciones

✓ Aplicación euclidiana plana

- **Ciudades:** Agujeros que tienen que ser perforados en la tableta de circuitos impresos.



N = 2392

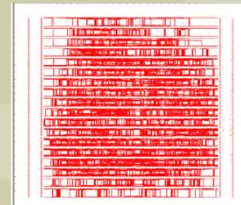
- **Ciudades:** Cables que van a ser cortados en un circuito programable "Láser Lógico"



N = 7397



N = 33,810



N = 85,900

ITESM, Dr. Gildardo Sánchez Ante

91

Grafos

Problema del vendedor ambulante Traveling Salesman Problem (TSP)

Resolviendo el TPS

A medida que aumenta el número de ciudades, una computadora no podrá resolver estos problemas en un período de tiempo razonable.

- → Resolver el problema del vendedor ambulante significa **buscar un algoritmo que realice un número de cálculos aceptable**.

Soluciones óptimas locales

Dado que no se ha encontrado una solución óptima a nivel mundial, existen muchos algoritmos que brindan soluciones óptimas a nivel local.

Definiciones

- Un **grafo completo** K_N es un grafo con N vértices y una arista que une cada dos de esos vértices.
- Un **circuito Hamiltoniano** es un circuito que utiliza cada vértice de un grafo en una sola ocasión.
- Una **grafo ponderado** es un grafo en el cual cada arista tiene asignado un peso (que puede representar tiempo, distancia o el costo de atravesar esa arista).
- El **TSP** es el problema de encontrar un circuito hamiltoniano de peso mínimo en K_N .

THE TRAVELLING SALESMAN PROBLEM

WHAT'S THE SHORTEST ROUTE TO VISIT ALL LOCATIONS AND RETURN?



ADDING MORE STOPS TAKES LONGER AND LONGER- AND LONGER TO FIGURE IT OUT

ITESM, Dr. Gildardo Sánchez Ante

92

Grafos



Problema del vendedor ambulante Traveling Salesman Problem (TSP)

Algoritmo del vecino más cercano

1. Elige un vértice de referencia para iniciar.
2. Camina a su vecino más cercano (por ejemplo: a lo largo de las aristas más cortas posibles).
3. En cada estado del recorrido, camina hacia el vecino más cercano que no se haya visitado.
4. Cuando se hayan visitado todos los vértices, regresa al vértice de inicio.

Algoritmo repetitivo del vecino más cercano

1. Elige un vértice de referencia para iniciar y utiliza el algoritmo del vecino más cercano para encontrar un **circuito Hamiltoniano**.
2. Repite el paso 1 para **cada** posible vértice inicial. Se deben tener un total de N circuitos hamiltonianos.
3. Elige el más barato.

Usualmente, no existe una forma de saber anticipadamente cual vértice de referencia trabajará mejor.

ITESM, Dr. Gildardo Sánchez Ante

93

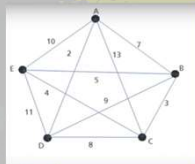
Grafos



Actividad: Problema del vendedor ambulante

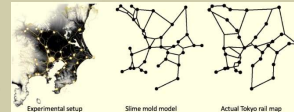
Instrucciones:

- 1) Para el siguiente grafo, realiza una propuesta de solución.



Algoritmo de enlace más barato (CLA, en inglés)

1. Agrega el vértice disponible más económico a tu recorrido.
2. Repite hasta que se tenga un circuito hamiltoniano.
3. Asegúrate de agregar exactamente dos aristas en cada vértice.
4. No cerrar el circuito hasta que todos los vértices se encuentren en él.



- 2) Revisa los siguientes enlaces de una propuesta de solución al TSP

- El "blob", la extraordinaria criatura que nos obliga a cuestionarnos si somos la especie más inteligente, https://www.bbc.com/espanol/noticias/2015/05/150522_bbc_blob
- Slime Mold Can Solve Exponentially Complicated Problems in Linear Time, <https://www.sciencedirect.com/science/article/pii/S0022006815000000>
- A Slime Mold-Ant Colony Fusion Algorithm for Solving Traveling Salesman Problem, https://www.researchgate.net/publication/244473771_A_Slime_Mold-Ant_Colony_Fusion_Algorithm_for_Solving_Traveling_Salesman_Problem

- a) Realiza un resumen (máximo una cuartilla) o una presentación (máximo 5 dispositivas) sobre los antecedentes, esencia, y desarrollo del algoritmo de *slime mold*.
- b) Agrega las conclusiones sobre este tipo de algoritmos alternos a los tradicionales para resolver el problema del TSP (vendedor viajero)
- c) Recuerda incluir las referencias en formato APA.

94

Grafos

Algoritmos

Algoritmo voraz de Kruskal (1956)

- El conjunto de aristas T está vacío inicialmente.
- Las aristas de G se ordenan por orden creciente de longitud.
- A medida que progresa el algoritmo, se van añadiendo aristas a T .
- Mientras no haya encontrado una solución, el grafo parcial formado por nodos de G y aristas de T consta de varios componentes conexos.
- Para añadir una arista se verifica que ésta no produzca un ciclo en alguna de las componentes conexas formadas por las aristas de T y los nodos de G .
- Los elementos de T que se incluyen en una componente conexas dada forman un árbol de recubrimiento mínimo para esa componente.
- Al final, sólo queda una componente conexas, así que T es un árbol de recubrimiento mínimo para todos los nodos de G .

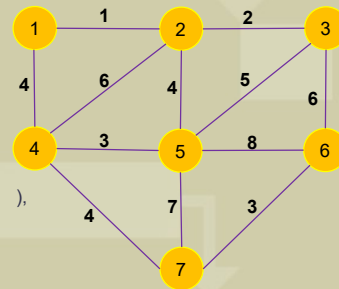
Ejemplo:

¿Cuál es el árbol de recubrimiento mínimo usando Kruskal?

■ Aristas ordenadas:

{1,2}	{4,7}
{2,3}	{3,5}
{4,5}	{2,4}
{6,7}	{3,6}
{1,4}	{5,7}
{2,5}	{5,6}

$T = [(,), (,), (,), (,), (,), (,), (,)]$



LIMTA, Dr. Alberto González

95

Grafos

Algoritmos

Algoritmo voraz de Kruskal

Pseudocódigo

```

función Kruskal( $G = \langle N, A \rangle$ : grafo; longitud:  $A \rightarrow R^+$ ): conjunto de aristas
{
    iniciación
    Ordenar  $A$  por longitudes crecientes
     $n \leftarrow$  el número de nodos en  $G$ 
     $T \leftarrow \emptyset$  {contendrá las aristas del árbol de recubrimiento mínimo}
    Iniciar  $n$  conjuntos, cada uno de los cuales
    contiene un elemento distinto de  $N$ 
    {bucle voraz}
    repetir
         $e \leftarrow \{u, v\} \leftarrow$  Arista más corta, aún no considerada
         $comp\_u \leftarrow$  buscar( $u$ )
         $comp\_v \leftarrow$  buscar( $v$ )
        si  $comp\_u \neq comp\_v$  entonces
            fusionar( $comp\_u$ ,  $comp\_v$ )
             $T \leftarrow T \cup \{e\}$ 
    hasta que  $T$  contenga  $n-1$  aristas
    devolver  $T$ 
}
    
```

LIMTA, Dr. Alberto González

96

Grafos

Algoritmos

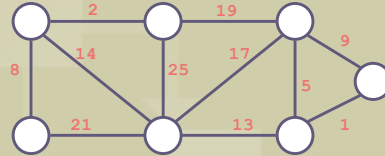
Algoritmo de Kruskal

Ejemplo

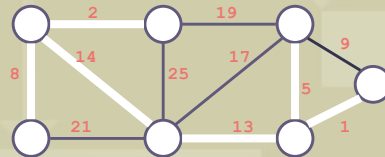
```

Kruskal()
{
  T = ∅;
  for each v ∈ V
    MakeSet(v);
  sort E by increasing edge weight w
  for each (u,v) ∈ E (in sorted order)
    if FindSet(u) ≠ FindSet(v)
      T = T ∪ {(u,v)};
      Union(FindSet(u), FindSet(v));
}
    
```

Run the algorithm:



Run the algorithm:



ITESM, Dr. Gildardo Sánchez Ante

97

Grafos

Algoritmos

Algoritmo voraz de Kruskal

Complejidad

En un grafo con n nodos y a aristas, el número de operaciones está en:

- $O(a \log(a))$ para ordenar las aristas, lo cual es equivalente a $O(a \log(n))$ porque $n-1 \leq a \leq n(n-1)/2$
- $O(n)$ para iniciar los n conjuntos disjuntos
- $O(a \log(n)) \rightarrow O(a \log(n))$ para las búsquedas
- $O((n-1)n) \rightarrow O(n^2)$ para las fusiones
- Todo lo demás es de orden $\Theta(1)$
- Complejidad: $O(n^2 + a \log n) \rightarrow O(n^2)$
- Algunos autores [brassard] consideran la complejidad de la fusión como $O(\log K)$ (considerando la operación de fusión de árboles que producen K nodos) con lo que la complejidad queda de $O(a \log(n))$.

UMTA, Dr. Alberto González

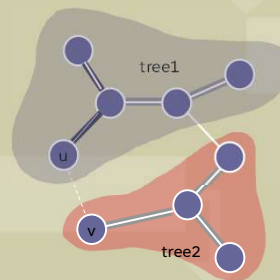
98

Grafos

Algoritmos

Algoritmo voraz de Kruskal vs Prim

- En Kruskal, la función de selección escoge las aristas por orden creciente de longitud, sin preocuparse por su conexión con las aristas seleccionadas anteriormente (*salvo la prevención de no formar un ciclo*).
 - El resultado es un bosque de árboles que crece al azar, hasta que al final forman un árbol único.
- En Prim, el algoritmo crece de forma natural, comenzando por una raíz arbitraria.
 - En cada fase se agrega una nueva rama al árbol, el algoritmo se detiene cuando se han alcanzado todos los nodos.



UMTA Dr. Alberto González

99

Grafos

Algoritmos

Algoritmo de Prim

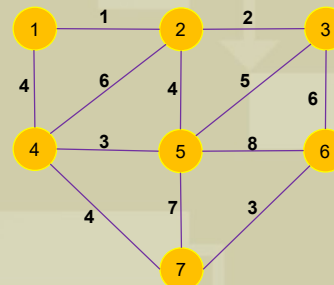
- Sea **B** un conjunto de **nodos**, y sea **T** un conjunto de **aristas**.
- Inicialmente, B contiene un único nodo arbitrario, y T está vacío.
- En cada paso, Prim busca la arista más corta posible $\{u,v\}$ tal que $u \in B$ y $v \in N \setminus B$. Entonces añade v a B y $\{u,v\}$ a T.
- De esta manera, todas las aristas de T forman en todo momento un árbol de recubrimiento mínimo para los nodos de B.
- Se continua mientras $B \neq N$.

Ejemplo:

¿Cuál es el árbol de recubrimiento mínimo usando Prim?

□ Aristas ordenadas: B = [__, __, __, __, __, __, __]

{1,2}	{4,7}	T = [(,), (,), (,), (,), (,), (,)]
{2,3}	{3,5}	
{4,5}	{2,4}	
{6,7}	{3,6}	
{1,4}	{5,7}	
{2,5}	{5,6}	



UMTA Dr. Alberto González

100

Grafos

Algoritmos

Algoritmo de Prim

- Las aristas del conjunto A siempre forman un árbol simple.
- Inicia de cualquier nodo arbitrario: $V_A = \{a\}$
- En cada paso:
 - Encuentra el nodo de menor peso cruzando $(V_A, V - V_A)$
 - Agrega esa arista en A
 - Repite hasta que el árbol se extienda para todos los vértices.

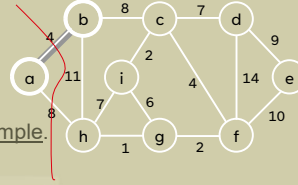
Pseudocódigo

función Prim($G = \langle N, A \rangle$: grafo; longitud: $A \rightarrow \mathbb{R}^+$): conjunto de aristas

```
{iniciación}
T ← ∅
B ← { se inicializa con un miembro arbitrario de N }
mientras B ≠ N hacer
    buscar e = {u,v} de longitud mínima tal que u ∈ B y v ∈ N \ B
    T ← T ∪ {e}
    B ← B ∪ {v}
fin_mientras
devolver T
```

MST-PRIM(G, w, r)

```
1 for each u ∈ G.V
2   u.key = ∞
3   u.π = NIL
4 r.key = 0
5 Q = G.V
6 while Q ≠ ∅
7   u = EXTRACT-MIN(Q)
8   for each v ∈ G.Adj[u]
9     if v ∈ Q and w(u, v) < v.key
10      v.π = u
11      v.key = w(u, v)
```



ITESM, Dr. Gildardo Sánchez Ante

IMTA, Dr. Alberto González

101

Grafos

Algoritmos

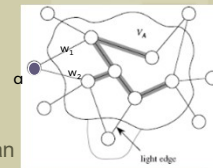
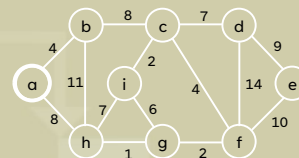
Algoritmo de Prim

¿Cómo encontrar las aristas de menor peso en forma rápida?

Utiliza una cola de prioridad Q:

- Contiene los vértices que no se han incluido en el árbol, es decir $(V - V_A)$
 - $V_A = \{a\}$, $Q = \{b, c, d, e, f, g, h, i\}$
- Se asocia una key con cada vértice v:

$key[v] = \text{peso mínimo a cualquier arista } (u, v) \text{ conectando } v \text{ to } V_A$ **Key[a] = min(w₁, w₂)**



- Después de agregar un nuevo nodo a V_A se actualizan los pesos para todos los nodos adyacentes a él

Ejemplo: después de agregar a al árbol, **k[b]=4** and **k[h]=8**
- La Key de v es ∞ , si v no es un nodo adyacente a cualquiera de los vértices en V_A

ITESM, Dr. Gildardo Sánchez Ante

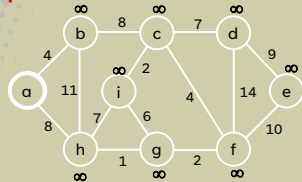
102

Grafos

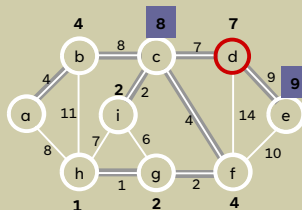
Algoritmos

Algoritmo de Prim

Ejemplo



0 ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
 $Q = \{a, b, c, d, e, f, g, h, i\}$
 $V_A = \emptyset$
 $\text{Extract-MIN}(Q) \Rightarrow a$



$\text{key}[e] = 9 \quad \pi[e] = d$
9
 $Q = \{e\} \quad V_A = \{a, b, c, i, f, g, h, d\}$
 $\text{Extract-MIN}(Q) \Rightarrow e$
 $Q = \emptyset \quad V_A = \{a, b, c, i, f, g, h, d, e\}$

ITESM, Dr. Gildardo Sánchez Ante

103

Grafos

Algoritmos

Algoritmo de Prim

Complejidad

- El bucle principal se ejecuta $n-1$ veces
- La búsqueda interna requiere de un tiempo n (dada la condición que verifica que $u \in B$ y $v \in MB$)
- Por lo tanto $O((n-1)n) \rightarrow O(n^2)$
- Si en promedio, Kruskal requiere un tiempo que está en $O(a \log(n))$ [brassard], considera que en el peor de los casos, $a = n(n-1)/2$ (para árboles muy densos), y la complejidad llega a $O(n^2 \log(n))$ por lo que en estos casos resulta más eficiente utilizar Prim.

UMTA, Dr. Alberto González, alberto.gonzalez@umta.mx

104