Parte de Algoritmos
de la asignatura de Programación
Master de Bioinformática

Grafos

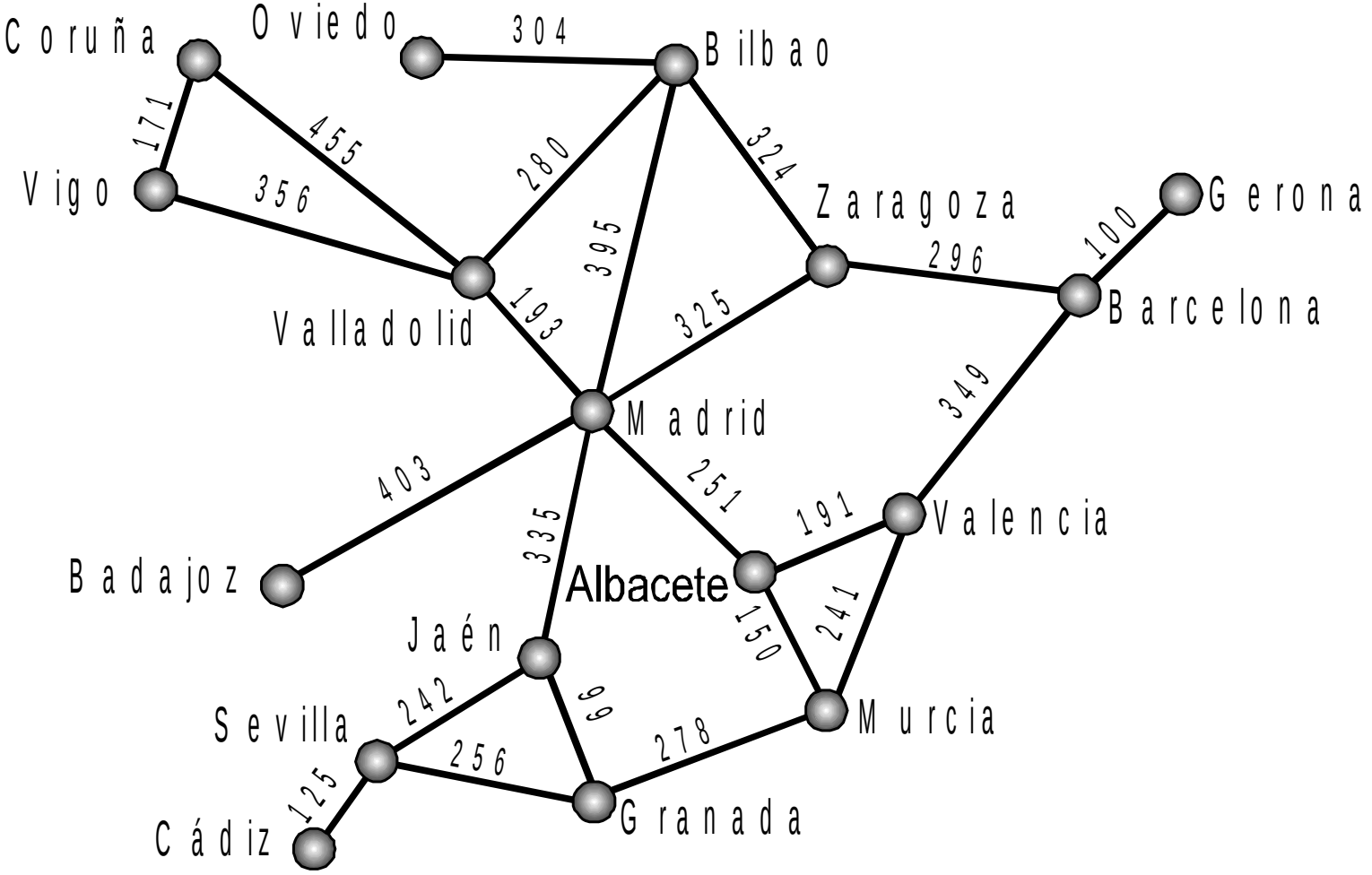
Web asignatura: http://dis.um.es/~domingo/algbio.html

E-mail profesor: domingo@um.es

Transparencias preparadas a partir de las Ginés GarcíEstructuras de Datos I, del Grado de Ingeniería Informática y
An Introduction to Bioinformatics Algorithms

Ejemplos de grafos

- **Ejemplo:** Grafo de carreteras entre ciudades.



Ejemplos de grafos

- **Ejemplo:** Grafo de carreteras entre ciudades.

Problemas

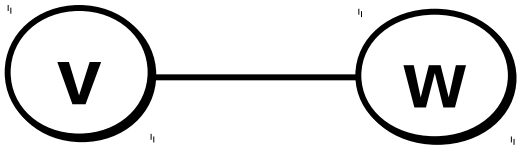
- ¿Cuál es el camino más corto de Murcia a Badajoz?
- ¿Existen caminos entre todos los pares de ciudades?
- ¿Cuál es la ciudad más lejana a Barcelona?
- ¿Cuál es la ciudad más céntrica?
- ¿Cuántos caminos distintos existen de Sevilla a Zaragoza?
- ¿Cómo hacer un tour entre todas las ciudades en el menor tiempo posible?

Introducción y definiciones

- **Un grafo G** es una tupla G= (V, A), donde V es un conjunto no vacío de **vértices** o **nodos** y A es un conjunto de **aristas** o **arcos**.
- Cada **arista** es un par (v, w), donde v, w $\in$ V.

Tipos de grafos

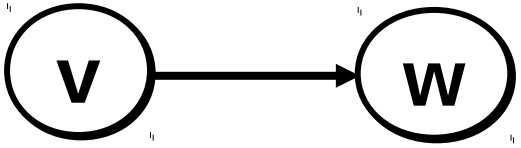
- **Grafo no dirigido.**



Las aristas no están ordenadas:

(v, w) = (w, v)

- **Grafos dirigidos (o digrafos).**



Las aristas son pares ordenados:

<v, w> $\neq$ <w, v>

<v, w> $\Rightarrow$ w = cabeza de la arista, v = cola.

Terminología de grafos

- **Nodos adyacentes a un nodo v:** todos los nodos unidos a v mediante una arista.
- En grafos dirigidos:
  - **Nodos adyacentes a v:** todos los w con $\langle v, w \rangle \in A$.
  - **Nodos adyacentes de v:** todos los u con $\langle u, v \rangle \in A$.

- Un grafo está **etiquetado** si cada arista tiene asociada una etiqueta o valor de cierto tipo.
- **Grafo con pesos:** grafo etiquetado con valores numéricos.
- **Grafo etiquetado:** G= (V, A, W), con **W**: A $\rightarrow$ TipoEtiq

Terminología de grafos

- **Camino de un vértice w_1 a w_q:** es una secuencia $w_1, w_2, ..., w_q \in V$, tal que todas las aristas $(w_1, w_2), (w_2, w_3), ..., (w_{q-1}, w_q) \in A$.
- **Longitud de un camino:** número de aristas del camino = nº de nodos -1.
- **Camino simple:** aquel en el que todos los vértices son distintos (excepto el primero y el último que pueden ser iguales).
- **Ciclo:** es un camino en el cual el primer y el último vértice son iguales. En grafos no dirigidos las aristas deben ser diferentes.
- Se llama **ciclo simple** si el camino es simple.

Terminología de grafos

- Un **subgrafo** de G=(V, A) es un grafo G'=(V', A') tal que V'⊆V y A'⊆A.
- Dados dos vértices v, w, se dice que están **conectados** si existe un camino de v a w.
- Un grafo es **conexo** (o **conectado**) si hay un camino entre cualquier par de vértices.
- Si es un grafo dirigido, se llama **fuertemente conexo**.
- Un **componente (fuertemente) conexo** de un grafo G es un subgrafo maximal (fuertemente) conexo.

Terminología de grafos

- Un grafo es **completo** si existe una arista entre cualquier par de vértices.

- **Grado de un vértice v:** número de arcos que inciden en él.
- Para grafos dirigidos:
  - **Grado de entrada de v:** nº de aristas con <x, v>
  - **Grado de salida de v:** nº de aristas con <v, x>

Operaciones elementales con grafos

- Crear un **grafo vacío** (o con **n** vértices).
- **Insertar** un nodo o una arista.
- **Eliminar** un nodo o arista.
- **Consultar** si existe una arista (obtener la etiqueta).
- **Iteradores** sobre las aristas de un nodo:

  para todo nodo *w* adyacente a *v* **hacer**

    acción sobre *w*

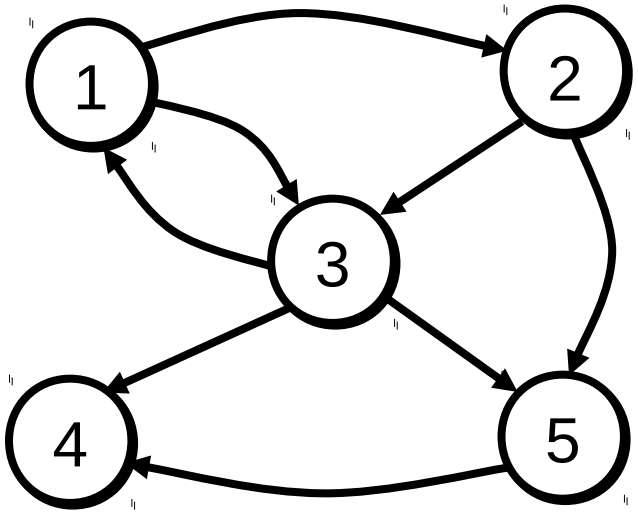
  para todo nodo *w* adyacente de *v* **hacer**

    acción sobre *w*

Mucho menos frecuente

Representación de grafos

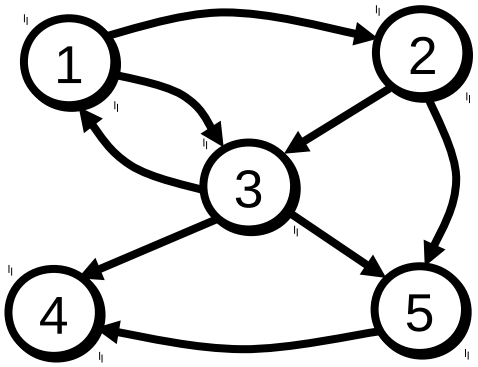
- **Representación de grafos**:
  - Representación del conjunto de nodos, V.
  - Representación del conjunto de aristas, A.



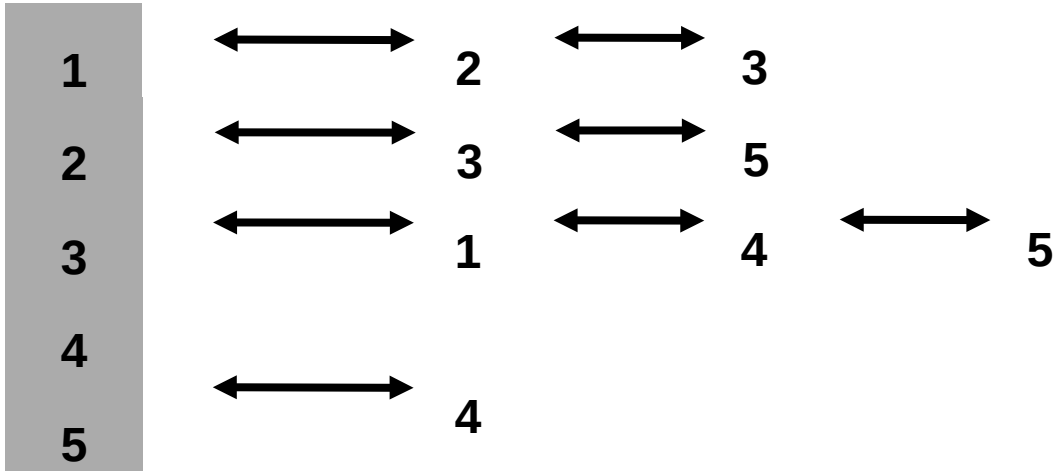
Representación de grafos

- **Representación del conjunto de aristas, A.**
  - Mediante **matrices de adyacencia**.

M	1	2	3	4	5
1		T	T		
2			T		T
3	T			T	T
4					
5				T	



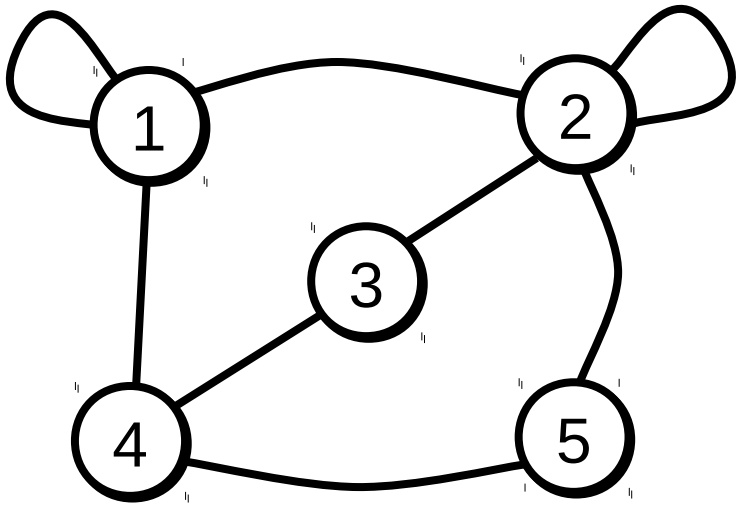
  - Mediante **listas de adyacencia**.



Matrices de adyacencia

tipo GrafoNoEtiq= **array** [1..n, 1..n] **de** booleano

- Sea M de tipo GrafoNoEtiq, G= (V, A).
- M[v, w] = cierto $\Leftrightarrow$ (v, w) $\in$ A



M	1	2	3	4	5
1	T	T		T	
2	T	T	T		T
3		T		T	
4	T		T		T
5		T		T	

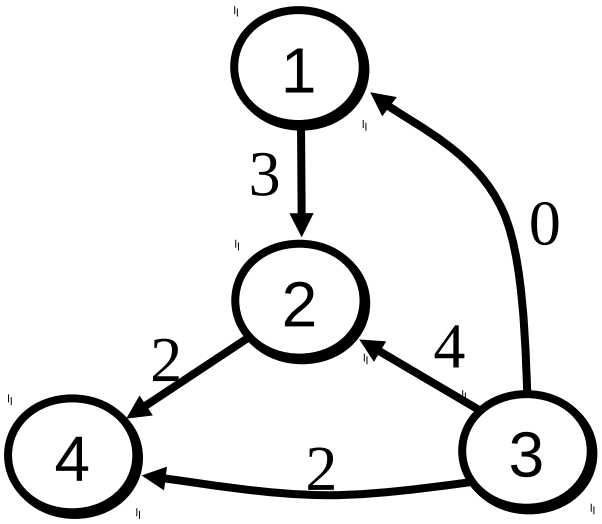
- Grafo no dirigido → Matriz simétrica: M[i, j] = M[j, i].
- **Resultado:** se desperdicia la mitad de la memoria.

Matrices de adyacencia

- **Grafos etiquetados:**

  tipo GrafoEtiq[E]= array [1..n, 1..n] **de** E

- El tipo E tiene un valor NULO, para el caso de no existir arista.



M	1	2	3	4
1		3		
2				2
3	0	4		2
4				

Recorridos sobre grafos

- Idea similar al recorrido en un árbol.
- Se parte de un nodo dado y se visitan los vértices del grafo de manera ordenada y sistemática, *moviéndose* por las aristas.

- **Tipos de recorridos**:
  - **Búsqueda primero en profundidad.** Equivalente a un recorrido en preorden de un árbol.
  - **Búsqueda primero en amplitud o anchura.** Equivalente a recorrer un árbol por niveles.

- Los recorridos son una **herramienta** útil para resolver muchos problemas sobre grafos.

Recorridos sobre grafos

- El recorrido puede ser tanto para grafos dirigidos como no dirigidos.
- Es necesario llevar una cuenta de los nodos visitados y no visitados.

```
var
    marca: array [1, ..., n] de (visitado, noVisitado)

operación BorraMarcas
    para i:= 1, ..., n hacer
        marca[i]:= noVisitado
```

Búsqueda primero en profundidad

```
operación bpp (v: nodo)
   marca[v]:= visitado
   para cada nodo w adyacente a v hacer
        si marca[w] == noVisitado entonces
            bpp(w)
   finpara

operación BúsquedaPrimeroEnProfundidad
   BorraMarcas
   para v:= 1, ..., n hacer
        si marca[v] == noVisitado entonces
            bpp(v)
   finpara
```

Problemas de caminos mínimos

- **Definición:** Dado un grafo ponderado G= (V, A) (dirigido o no) y un camino w_1, w_2, ..., w_q en G, el **costo del camino** será la suma de los costos asociados a las aristas (w_1, w_2), ..., (w_{q-1}, w_q).

- Si el grafo es no ponderado, normalmente el costo se asocia con la longitud del camino.

- **Problema de los caminos más cortos por un origen:**
  Encontrar los caminos más cortos entre un nodo origen dado a todos los demás nodos.

Algoritmo de Dijkstra

- Supongamos un grafo ponderado G (con pesos ≥ 0) y un nodo origen v.
- El algoritmo trabaja con dos conjuntos:
  - **S: conjunto de nodos escogidos**, para los cuales se conoce el camino de distancia mínima al origen.
  - **C: conjunto de nodos candidatos**, pendientes de calcular el camino mínimo. Conocemos los caminos mínimos al origen pasando por nodos de S.
- En cada paso coger del conjunto de candidatos el nodo con distancia mínima al origen. Recalcular los caminos de los demás candidatos pasando por el nodo cogido.
- Un **camino especial** del origen a otro nodo cualquiera es un camino que sólo pasa por nodos ya escogidos.
- Supongamos que el nodo origen es el 1.

Algoritmo de Dijkstra

- En un array **D[2, ..., N]** se guarda la longitud del camino especial más corto a cada vértice. Cuando todos los nodos estén en S, todos los caminos son especiales y D contiene las distancias mínimas al origen.
- En otro array **P[2, ..., N]** se almacena el camino por el que pasa cada nodo v. El camino de 1 a v pasa por P[v].
- Inicialmente D contendrá los caminos directos de 1 a los restantes nodos, es decir d[1, x]. Si no existe la arista (1, x) el costo será ∞.

  P contendrá el valor 1 (el camino es directo). S contendrá sólo el nodo 1.
- Buscar el nodo v de C=V-S con mínimo valor de D. Añadir v a S. Para el resto de nodos comprobar si el camino al origen es más corto pasando por el nodo v:

```
si D[v]+d[v, w] < D[w]
    D[w]:= D[v] + d[v, w]
    P[w]:= v
```

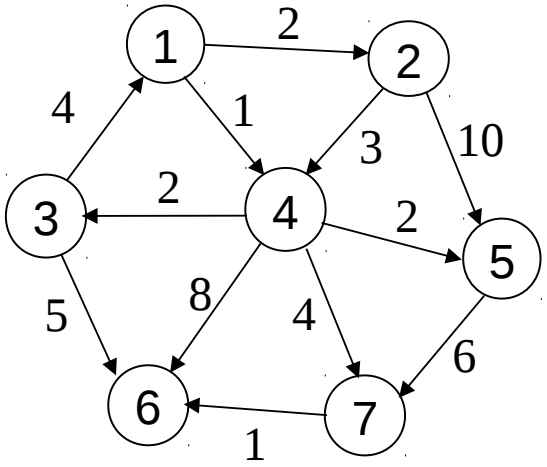
Algoritmo de Dijkstra

```
para i:= 2, ..., N
   S[i]:= FALSO
   D[i]:= d[1, i]
   P[i]:= 1
para i:= 1, ..., N-1
   v:= vértice con D[v] mínimo y
   S[v]=FALSO
   S[v]:= VERDADERO
   para cada nodo w adyacente a v
      si S[w]=FALSO
         si D[v]+d[v, w]<D[w]
      D[w]:= D[v]+C[v, w]
            P[w]:= v
```

```
Operación ImprimeCamino
   (v:entero)
   si v != 1
      ImprimeCamino(P[v])
   escribir v
```

Algoritmo de Dijkstra

- **Ejemplo:**



Nodo
2
3
4
5
6
7

Inicializ.

S	D	P
F	2	1
F	~~1~~	~~1~~
F	1	1
F	~~1~~	~~1~~
F	~~1~~	~~1~~
F	~~1~~	~~1~~

v = 4

S	D	P
F	2	1
F	**3**	**4**
T	1	1
F	**3**	**4**
F	**9**	**4**
F	**5**	**4**

v = 2

S	D	P
T	2	1
F	3	4
T	1	1
F	3	4
F	9	4
F	5	4

v = 3

S	D	P
T	2	1
T	3	4
T	1	1
F	3	4
F	**8**	**3**
F	5	4

5, 7

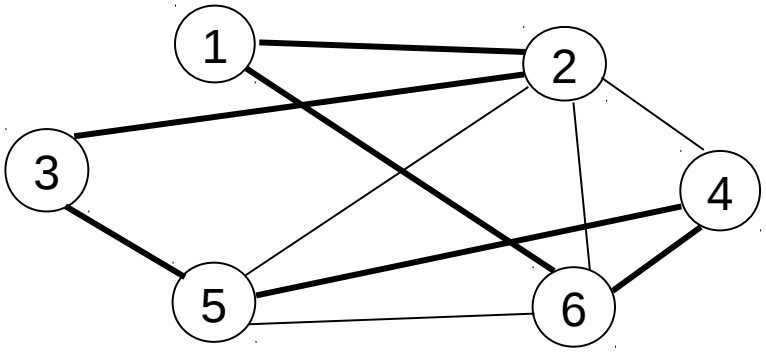
.....

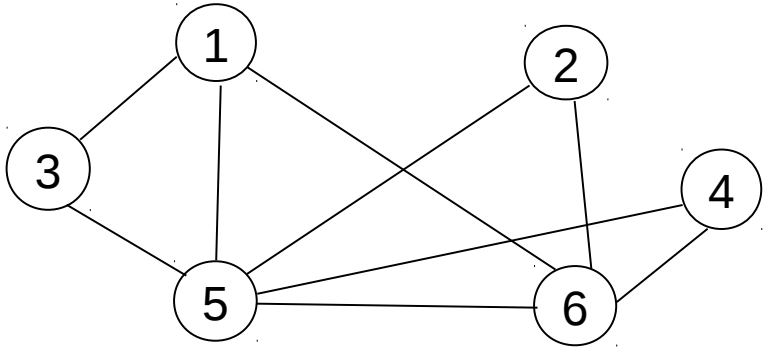
v = 6

S	D	P
T	2	1
T	3	4
T	1	1
T	3	4
T	6	7
T	5	4

Problema del ciclo hamiltoniano

- Dado un grafo no dirigido G, un **ciclo hamiltoniano** es un ciclo simple que visita todos los vértices.
- **Problema del ciclo hamiltoniano.**
  Determinar si un grafo no dirigido dado tiene un ciclo hamiltoniano.

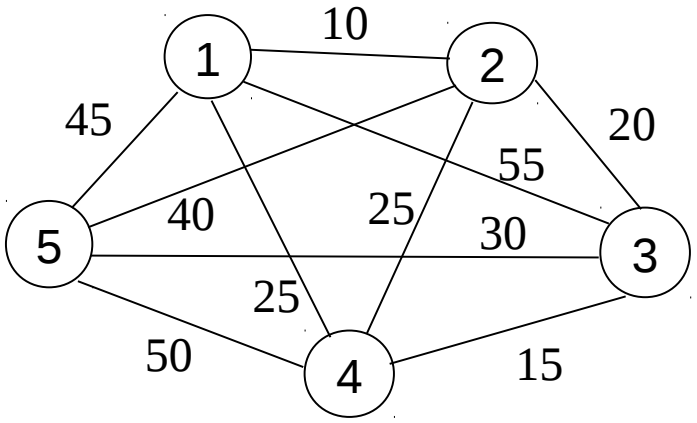




- No se conoce ningún algoritmo para resolverlo en tiempo polinomial.

Problema del viajante

- Dado un grafo no dirigido, completo y ponderado G = (V, A), encontrar un ciclo simple de costo mínimo.



- **Ejemplos:** Un repartidor de determinadas mercancías tiene encargos en varias ciudades. ¿Qué ruta debe seguir para que el costo de desplazamiento sea mínimo?
- El problema del viajante es un problema **NP-completo**, con un orden de complejidad exponencial. No existe una solución polinómica.
- Podemos aplicar heurísticas, obteniendo soluciones aproximadas, no necesariamente óptimas.

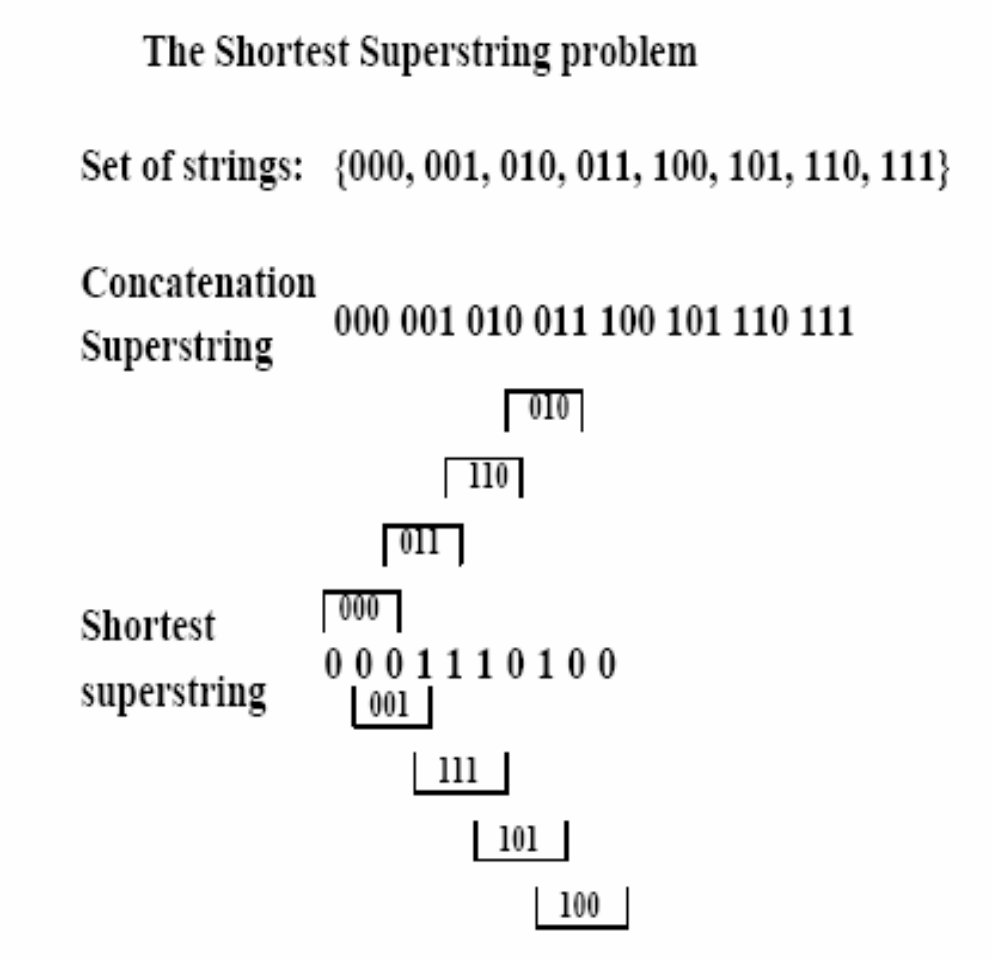
Problema de la Supercadena más Corta

Dado un conjunto de cadenas, encontrar la cadena más corta que contiene a todas las cadenas.

- Entrada: Cadenas s_1, s_2,…., s_n
- Salida: Una cadena s que contiene a todas las cadenas s_1, s_2,…., s_n como subcadenas y tal que la longitud de s es mínima

- **Complejidad:** NP – completo (no hay algoritmos eficientes para este problema)

Problema de la Supercadena más Corta

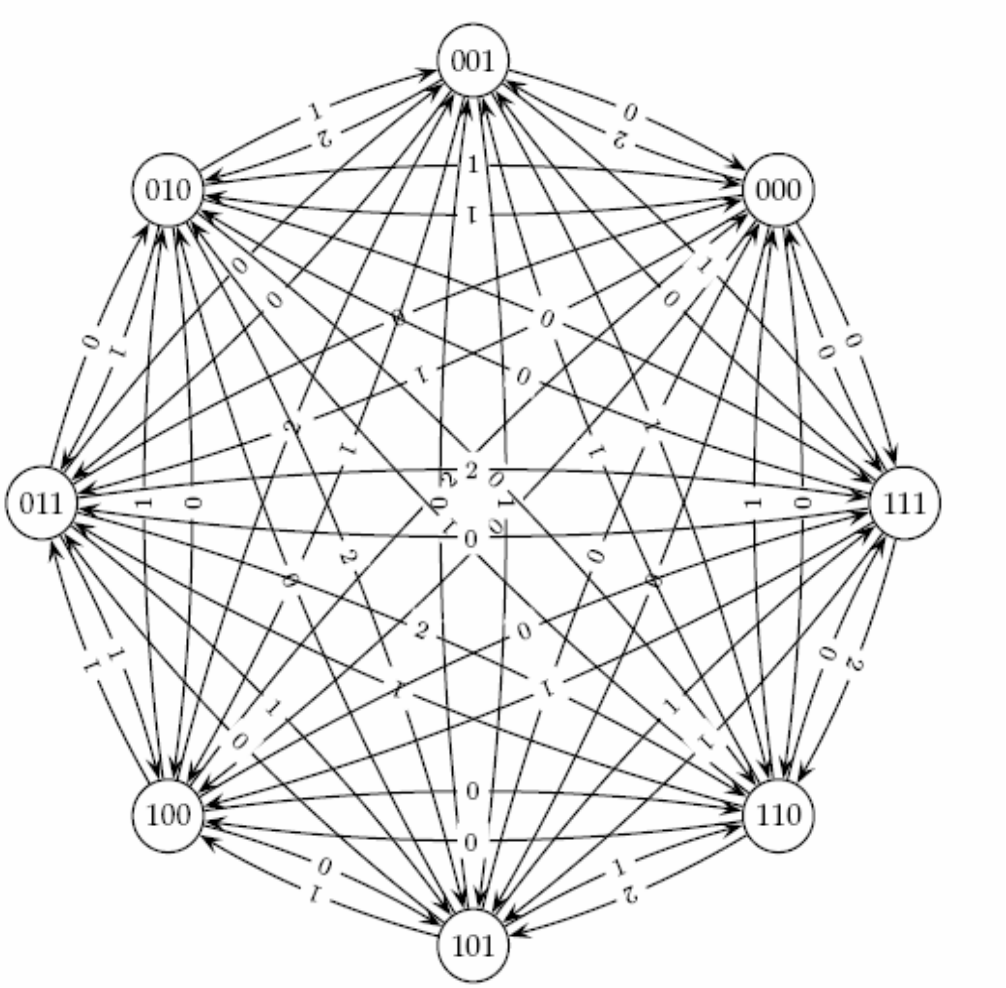


Problema de la Supercadena más Corta

Se puede reducir al Problema del Viajante de Comercio (TSP)

- Se define *overlap* (s_i, s_j) como la longitud del prefijo más largo de s_j que coincide con un sufijo de s_i.
- Se construye un grafo con n vértices que representan las n cadenas s_1, s_2,…., s_n.
- Con aristas con peso *overlap* (s_i, s_j) del vértice s_i al s_j.
- Se trata de encontrar el camino más largo que visita cada vértice una vez: variante del **Problema del Viajante de Comercio** (TSP)

Problema de la Supercadena más Corta



Problema de la Supercadena más Corta

S = { ATC, CCA, CAG, TCC, AGT }

SSP

AGT

CCA

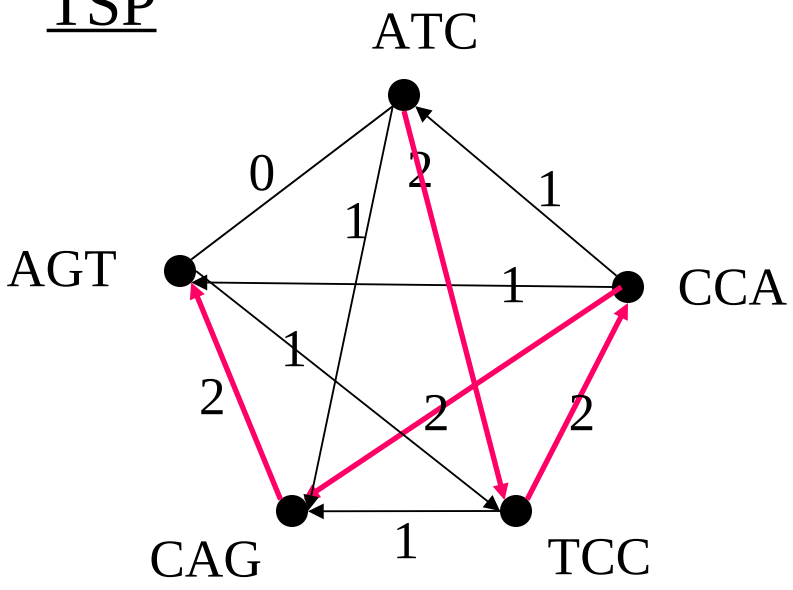
ATC

ATCCAGT

TCC

CAG

TSP



ATCCAGT