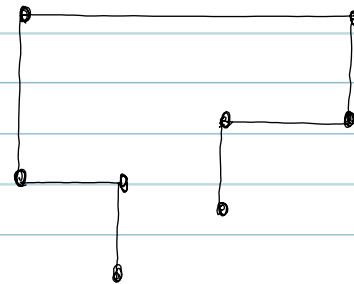
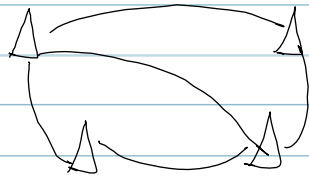


Introducción a la teoría de grafos



También en programación paralela:

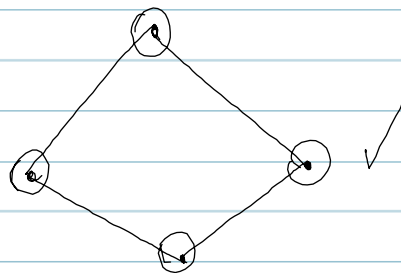
n -cubo

2^n

1-cubo

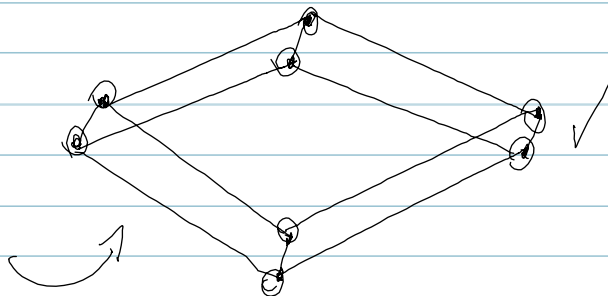


2-cubo



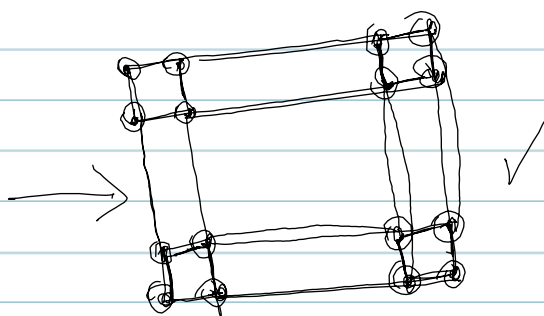
3-cubo

2^3



4-cubo

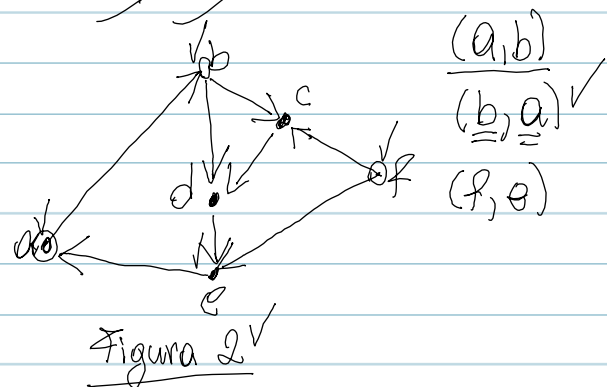
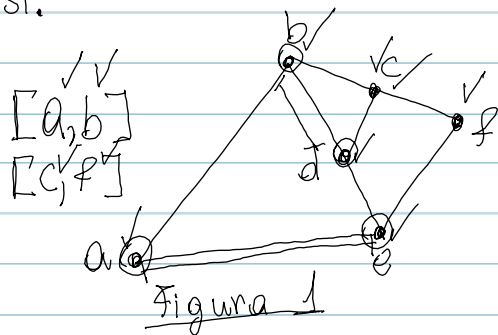
2^4



<< Combinatoria! ✓

Table[ShowGraph[✓✓
Hypercube[i], { i , 1, 4}],

Definición Sean V y E dos conjuntos no vacíos. V un conjunto de elementos llamados nodos o vértices y E un conjunto de aristas o lados. Un grafo $G = (V, E)$ es una relación R sobre el conjunto V tal que un nodo a está relacionado con otro b , si existe una arista $e \in E$ cuyos extremos son a y b . Si las aristas de G son dirigidas, es decir, tienen un orden en la dirección que relaciona los vértices de V , entonces a G se le llama grafo dirigido o digrafo, en caso contrario, se dice que G es un grafo no dirigido. Si G es dirigido y aRb al lado e de G que relaciona a y b , se le denota: $e = (a, b)$. Si G es un grafo no dirigido y aRb , la arista e que los une, se representa como: $e = [a, b]$. Dos vértices relacionados por R se llaman adyacentes. Se dice además, que una arista e incide sobre dos nodos a y b , si e los conecta entre sí.



$V = \{a, b, c, d, e, f\}$

El software Mathematica → Combinatorica → 450

→ "Graph" ✓
 → "ShowGraph" ✓

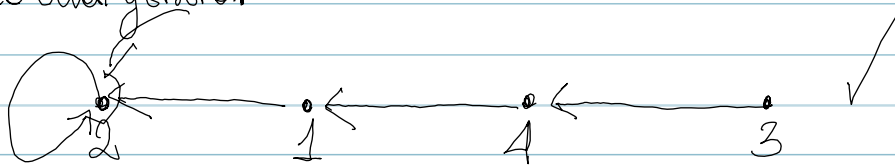
Por ejemplo: $G = (V, E)$, $V = \{1, 2, 3, 4\}$ y $E = \{(1, 2), (2, 2), (3, 4), (4, 1)\}$

→ ✓ → DirectedEdge y UndirectedEdge

Graph[$\{1 \rightarrow 2, 2 \rightarrow 2, 3 \rightarrow 4, 4 \rightarrow 1\}$, VertexLabels → "Name", ImagePadding → 10, EdgeStyle → Arrowheads[Medium], EdgeShapeFunction → Arrow]

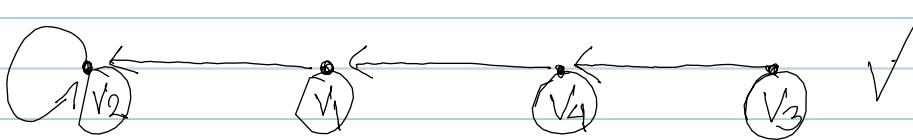
Palettes / Special Characters / Symbols / X

Lo cual genera:



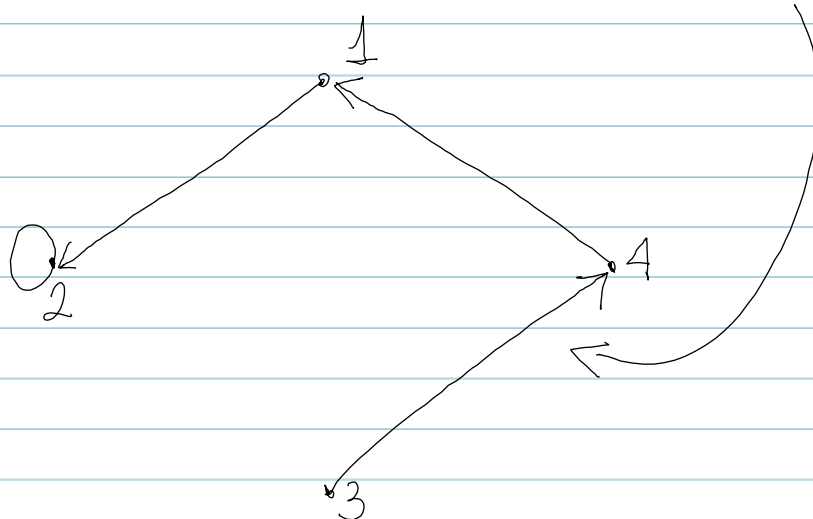
Por otra parte: VertexLabels → Table

Graph [{ 1 → 2, 2 → 2, 3 → 4, 4 → 1 },
VertexLabels → Table [{ i → v_i, { 1, 4 } }], ImagePadding → 10,
 EdgeStyle → Arrowheads [Medium], EdgeShapeFunction →
 "Arrow"]



Al recurrir a ShowGraph, se obtiene algo similar:

LL Combinatoria! ✓
 h = { { 1, 2 }, { 2, 2 }, { 3, 4 }, { 4, 1 } };
 ShowGraph [SetGraphOptions [FromOrderedPairs [h], VertexColor → Gray,
 EdgeColor → Black, VertexLabel → True, PlotRange → 0.1]



✓ Graph 1. Evaluation / Out Kernel / Local 2. Quit []

Can Graph $\left\{ \begin{array}{l} \text{Vertex List} \checkmark \rightarrow V \\ \text{Edge List} \checkmark \rightarrow E \end{array} \right.$
 \downarrow
 $G = (V, E)$

For example: ✓

\downarrow
 $G = \text{Graph} [\{ 1 \rightarrow 2, 2 \rightarrow 2, 3 \rightarrow 4, 4 \rightarrow 1 \} \checkmark,$
 $\text{Vertex Labels} \rightarrow \text{Table} [i \rightarrow \text{Vi}, \{i, 1, 4\}], \text{Image Padding} \rightarrow 10,$
 $\text{Edge Style} \rightarrow \text{Arrowheads} [\text{Medium}],$
 $\text{Edge Shape Function} \rightarrow \text{"Arrow"}];$
 $\rightarrow \text{Vertex List} [G] \checkmark \leftarrow$
 $\rightarrow \text{Edge List} [G] \checkmark \leftarrow$

out []

$\{ 1, 2, 3, 4 \} \checkmark \leftarrow$

$\{ 1 \rightarrow 2, 2 \rightarrow 2, 3 \rightarrow 4, 4 \rightarrow 1 \}$

La sentencia Edges del paquete "combinatorica", $G = (V, E)$
Vertices $\{ 1, 2, \dots, n \}$

For example:

$\ll \text{Combinatorica}$
 $n = \{ \{ 1, 2 \}, \{ 2, 2 \}, \{ 3, 4 \}, \{ 4, 1 \} \};$
 $\text{ShowGraph} [$
 $\text{Set GraphOptions} [\text{From Ordered Pairs} [h],$
 $\text{VertexColor} \rightarrow \text{Gray}, \text{EdgeColor} \rightarrow \text{Black}, \text{VertexLabel} \rightarrow \text{True},$
 $\text{Plot Range} \rightarrow 0.1];$
 $\checkmark \text{Edges} [G]$
 $\checkmark \text{Vertices} [G]$

$\{ \{ 1, 2 \}, \{ 2, 2 \}, \{ 3, 4 \}, \{ 4, 1 \} \}$

$\{ \{ 0, 1. \}, \{ -1., 0 \}, \{ 0, -1. \}, \{ 1., 0 \} \}$

Otros conceptos:

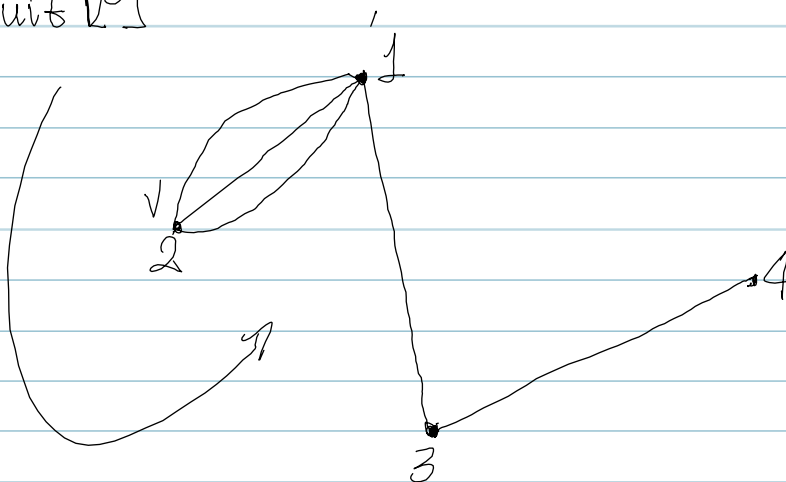
Si en un grafo $G = (V, E)$ una arista (e) conecta a un nodo (a) consigo mismo, a dicha arista se le denomina lazo o ciclo de G .

Mathematica \rightarrow Simple Graph Q = $\begin{cases} \text{True} \\ \text{False} \end{cases}$
 \downarrow
 Graph

se dice que dos nodos a y b poseen lados múltiples en un grafo (G) si existe más de una arista entre a y b.

LL Combinatoria! ✓✓
 $\rightarrow h = \{ \{1, 2\}, \{1, 2\}, \{1, 2\}, \{2, 3\}, \{2, 3\}, \{2, 3\}, \{3, 4\} \}$

Show Graph []
 $G = \text{SetGraphOptions [FromUnorderedPairs [h],$
 VertexColor \rightarrow Gray, EdgeColor \rightarrow Black], VertexLabel \rightarrow True,
 PlotRange \rightarrow 0.1]
 Quit []

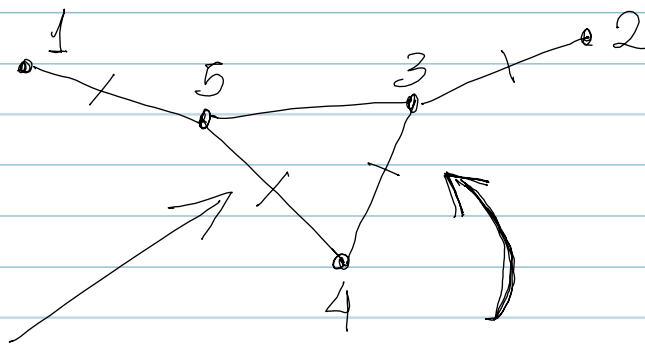


Graph \leftrightarrow Show Graph

un grafo (G) sin lazos y lados múltiples recibe el nombre de 'grafo simple'.

El grafo G mostrado a continuación, es simple:

Random Graph [$\{1, 5, 5\}$, VertexLabels \rightarrow "Name", ImagePadding \rightarrow 10]



Una ruta, camino o trayectoria entre dos nodos a y b de un grafo $G = (V, E)$ es una sucesión de vértices de V que conecta al nodo a con el vértice b , por medio de las aristas de E sobre G , sin repetición de lados. La longitud de la ruta se define como la cantidad de aristas de G utilizadas para conectar a con b .

1 a 2
 $1 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2$: (4)
 $t \rightarrow t-1$

Matematica:

NetworkFlow

<<Combinatoria

$h = \{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 2\}, \{3, 4\}, \{4, 5\}, \{3, 5\}$;

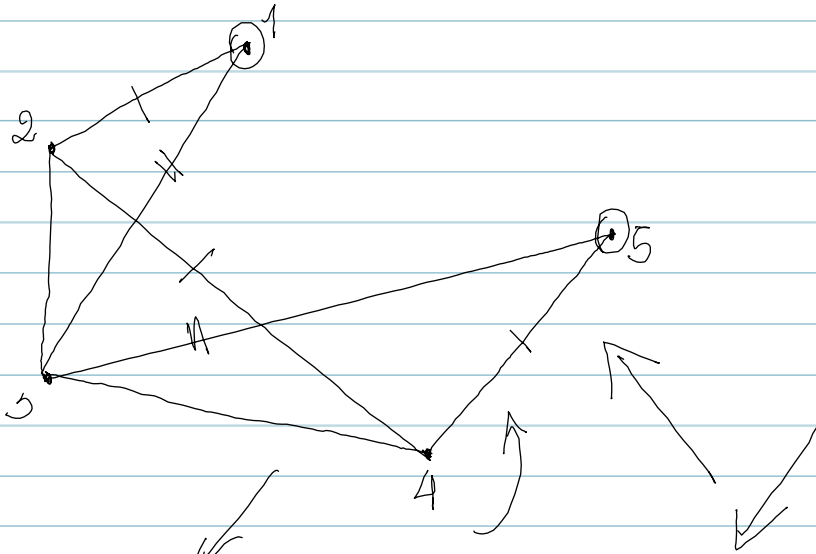
showGraph [$G = \text{SetGraphOptions}[\text{FromUnorderedPairs}[h],$
 VertexColor \rightarrow Gray, EdgeColor \rightarrow Black], VertexLabel \rightarrow True,
 PlotRange \rightarrow 0.1]

NetworkFlow [$G, \{1, 5\}$]

NetworkFlow [$G, \{1, 5\}, \text{Edge}$]

Quit[]

Out []



②

→ d[2,2], 2, 1, d[2,3], 1, d[2,4], 1, d[3,5], 1, d[4,5], 1

↳ 1 → 2 → 4 → 5
 1 → 3 → 5

El siguiente programa automatiza la elaboración de las rutas:

Combinatoria

$n = \{2, 1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 2\}, \{3, 4\}, \{4, 5\}, \{3, 5\}$

Show Graph [G] = Set GraphOptions [From Unordered Pairs [n],
 VertexColor → Gray, EdgeColor → Black, VertexLabel → True,

PlotRange → 0.1];

ContiRutas = NetworkFlow [G, 1, 5];

L = NetworkFlow [G, 1, 5, Edge];

For [i = 1, i ≤ ContiRutas, v = L[[i, 1]]];

camino = ToString [v[[1]] → v[[2]]]; elemento = v[[2]];

→ For [j = i + 1, j ≤ Length [L], w = L[[j, 1]]];

If [elemento == w[[1]],

camino = StringJoin [camino, "→", ToString [w[[2]]];

elemento = w[[2]]; j++]; Print [ToExpression [camino];

i++];

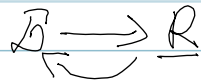
Quit []

1 → 2 → 4 → 5

1 → 3 → 5

Número de rutas posibles sobre un grafo simple

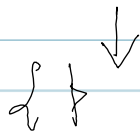
"0"



$$R \circ R \rightarrow (a, b) \rightarrow 2 \rightarrow B$$

$$R \circ (R \circ R) \rightarrow (a, b) \rightarrow 3 \rightarrow B$$

$$R \circ [R \circ (R \circ R)] \rightarrow (a'', b'') \rightarrow 4 \rightarrow B$$



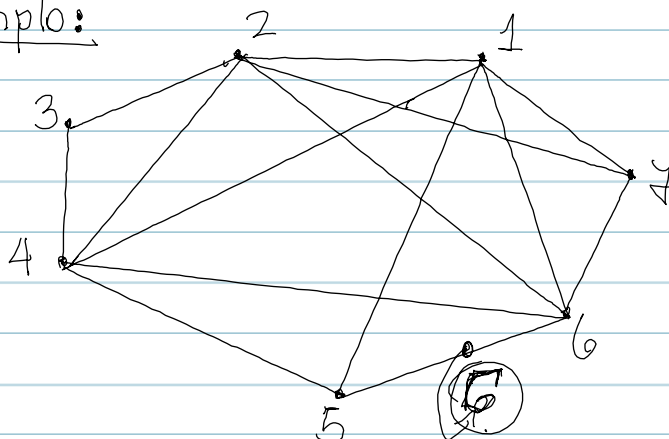
Mathematica: ✓ R o H ✓

```
Composiciones [R-List, H-List] := Module [ d[Composicion = {}],
  For [i = 1, i ≤ Length [H],
    For [j = 1, j ≤ Length [R],
      If [ H[[i, 2]] == R[[j, 1]],
        Composicion = Append [Composicion,
          { H[[i, 1]], R[[j, 2]] }]; j++]; i++];
  Return [DeleteDuplicates [Composicion]]]
```

→ CantRutas [G, Graph] := Module [d[Lr = 2], R = Edges [G]; c = R;

→ While [c ≠ {}, c = Composiciones [R, c];
 If [c ≠ {}, Print ["El grafo tiene ", Length [c],
 " rutas de longitud ", Lr, ", con extremos: "];
 Print [c]; Lr++];

Por ejemplo:



"CantRutas"

Entrega la salida:

El grafo tiene (14) rutas de longitud (2), con extremos:

→ 1,4,1,4, 1,1,7, 1,1,3, 1,1,6, 1,1,5, 1,3,5, 1,3,6, 1,2,5, 1,2,6, 1,4,6, 1,2,4, 1,2,7, 1,5,7, 1,4,7

El grafo tiene (10) rutas de longitud (3), con extremos:

→ 1,1,5, 1,1,6, 1,1,4, 1,1,7, 1,3,6, 1,3,7, 1,2,6, 1,2,7, 1,4,7, 1,2,5

El grafo tiene (6) rutas de longitud (4), con extremos:

→ 1,1,6, 1,1,7, 1,1,5, 1,3,7, 1,2,7, 1,2,6

El grafo tiene (3) rutas de longitud (5), con extremos:

→ 1,1,7, 1,1,6, 1,2,7

El grafo tiene (1) rutas de longitud (6), con extremos:

→ 1,1,7

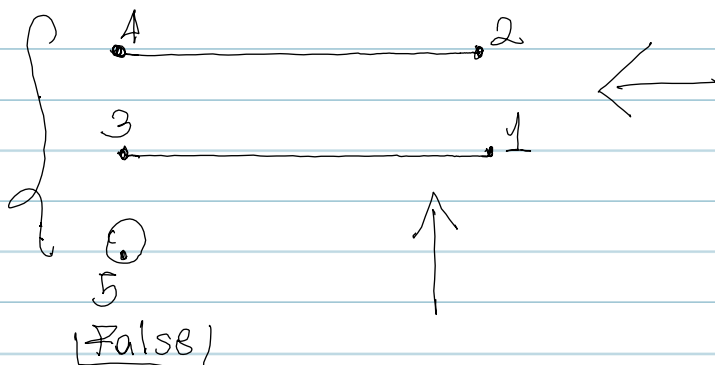
Un grafo (5) se dice que es conexo, cuando entre cada par de nodos cualesquiera siempre existe una ruta.

Connected Graph [5]

Por ejemplo:

"un grafo no dirigido con cinco nodos y dos aristas siempre es disconexo"

→ $G = \text{Random Graph}[5, 2]$ VertexLabels → "Name", ImagePadding → 10
Connected Graph[5] ✓



↓
Teorema sea $G = (V, E)$ un grafo no dirigido, entonces la relación R definida sobre V , tal que: $\forall a, b \in V, (aRb)$ si y solo si existe una trayectoria que une a con b , es una relación de equivalencia. Las clases de equivalencia de R se llaman componentes o piezas de G .

"Piezas"
↳ G

Ejemplo Verifique el teorema anterior por medio de un grafo pseudorandom generado en Mathematica.

TipoRelacion ✓

ConnectedComponents ✓

```
→ G = RandomGraph[ {20, 18}, VertexLabels -> "Name",  
    ImagePadding -> 10]  
→ A = VertexList[G];  
→ MC = ConnectedComponents[G];  
  << Combinatorica`  
  { R = {};  
    For [ i = 1, i <= Length[MC],  
      L = CartesianProduct[ MC[[i]], MC[[i]] ]; R = Union[L, R];  
      i++ ];  
    Print [ "R es : " ];  
→ TipoRelacion[R, A];  
Quit[]
```

En ciertos tipos de problemas a cada una de las aristas que determinan un grafo se le suele asociar un número (real), usualmente no negativo.

En Mathematica:

```

G = Graph[{1 -> 2, 2 -> 3, 3 -> 1}];
G = Graph[{1 -> 2, 2 -> 3, 3 -> 1},
  EdgeWeight -> RandomReal[{1, 5}], EdgeCount -> 5];

```

Weighted Adjacency Matrix [G]

(i, j)
 $\downarrow \quad \downarrow$
 $v_i \quad v_j$

También, Mathematica cuenta con:

→ Weighted Graph Q [G]

Usando Show Graph:

<< Combinatorica

n = {2, 2, 3, 2, 3, 1};

Show Graph [

G = Set Graph Options [From Unordered Pairs [n],

Vertex Color -> Gray, Edge Color -> Black],

Vertex Label -> True, Plot Range -> 0.1];

G = Set Edge Weights [G, RandomReal[{1, 5}, M[G]]];

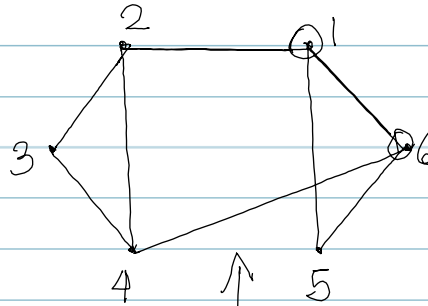
→ Set Edge Weights [G]

Quit [G]

Set Edge Weights [G, {2, 2, 3, 2, 3, 1}]

V[G], Cost of Path -> ∞

Ejemplo Encuentre la longitud de los caminos construidos por Network Flow que unen los vértices ① y ⑥ en el grafo:



En Mathematica:

<< Combinatorica

$h = \{1, 2\}, \{1, 6\}, \{2, 3\}, \{3, 4\}, \{2, 4\}, \{5, 1\}, \{5, 6\}, \{6, 4\};$

ShowGraph [

$G = \text{SetGraphOptions} [\text{FromUnorderedPairs} [h],$
 VertexColor \rightarrow Gray, EdgeColor \rightarrow Black, VertexLabel \rightarrow True,
 PlotRange \rightarrow 0.1]

ContiRutas = NetworkFlow [G, 1, 6];

L = NetworkFlow [G, 1, 6, Edge];

$G = \text{SetEdgeWeights} [G, \text{RandomReal} [\{1, 5\}, \text{Length}[E]]];$

For [i=1, i ≤ ContiRutas, v = L[[i, 1]];

$T_i = \{v[[1]], v[[2]]\}$; elemento = v[[2]];

For [j=i+1, j ≤ Length [L], w = L[[j, 1]];

If [elemento == w[[1]], $T_i \rightarrow \text{Append} [T_i, w[[2]]]$;

elemento = w[[2]]; j++]; i++]

For [i=1, i ≤ ContiRutas,

Print [T_i, " tiene longitud: ", CostOfPath [G, T_i]; i++]

Quit[]

Out[]

{1, 2, 4, 6} tiene longitud: 8.99443

{1, 5, 6} tiene longitud: 4.45207

{1, 6} tiene longitud: 4.83835

Otras opciones del software:

Graph Distance [G, a, b]

Graph Distance [G, a]

Otro concepto fundamental es el de valencia \rightarrow grado

Definición Sea $G = (V, E)$ un grafo. Si G es no dirigido y $a \in V$, se llama valencia o grado del nodo a , al número de lados de E que son incidentes sobre a , este se representa como $\theta(a)$. Si en a ocurre un bzo, el ciclo suma dos unidades a su grado. Por otra parte, si G es dirigido, se llama valencia o grado interno de a , denotado $\theta_*(a)$, al número de aristas de E que llegan al vértice a . La cantidad de lados de G que salen del nodo a , recibe el nombre de valencia o grado externo de a y se representa así: $\theta^*(a)$.

\rightarrow Teorema Sea $G = (V, E)$ un grafo no dirigido con $V = \{v_1, v_2, \dots, v_n\}$ y m su número de aristas, entonces:

$$\sum_{i=1}^n \theta(v_i) = 2m$$

Ejemplo Elabore una rutina en Mathematica que verifique el teorema anterior.

Solución 1:

```

→ For [i=2, i ≤ 11, Print [G = RandomGraph [2*i, 2*i], VertexLabels
→ "Name", ImagePadding → 10]];
Print ["El valor de verdad de la propiedad en este grafo es: "];
If [Total [VertexDegree [G]] == 2 EdgeCount [G],
Print ["True"], Print ["False"]]; i++]
```

Solución 2:

```
(n) = Input ["Digite la cantidad de nodos y aristas de los grafos: "];  
(m) = Input ["Digite la cantidad de grafos a generar: "];  
LG = RandomGraph [UniformGraphDistribution [n,n], (m),  
VertexLabels -> "Name", ImagePadding -> 10];  
→ For [i=1, i ≤ Length [LG], i++, G = LG [[i]]; Print [G];  
Print ["El valor de verdad de la propiedad en este grafo es: "];  
{ If [Total [VertexDegree [G]] == 2 EdgeCount [G], Print ["True"],  
Print ["False"]]; i++]
```

En Mathematica:

↓

Degrees [G] (1)

(2)

En un grafo (5) dirigido:

Graph } Vertex In Degree ✓
 } Vertex Out Degree ✓

ShowGraph } In Degree ✓
 } Out Degree ✓

Ejemplo conjetura que ocurre con el teorema:

"Sea $G = (V, E)$ un grafo con $V = \{v_1, v_2, \dots, v_n\}$ y m su número de aristas, entonces:

$$\sum_{i=1}^n \text{deg}(v_i) = 2m$$

Si G es dirigido.

Random Graph atributo DirectedEdges \rightarrow True

En Mathematica:

```
(n) = Input["Digite la cantidad de nodos y aristas de los grafos: "];
(m) = Input["Digite la cantidad de grafos a generar: "];
→ LG = RandomGraph[UniformGraphDistribution[(n), (n)], (m),
DirectedEdges → True, VertexLabels → "Name", ImagePadding → 10];
→ For[i = 1, i ≤ Length[LG], i++, G = LG[[i]]; Print[G];
Print["El valor de verdad de la propiedad en este grafo es: "];
→ If[Total[VertexInDegree[G]] == Total[VertexOutDegree[G]],
Print["True"], Print["False"]];
i++]
```

Se concluye la conjetura:

"en un grafo dirigido la suma de los grados internos y externos es el doble de la cantidad de lados y a su vez, la suma de las valencias internas es igual a la suma de los grados externos"

Concepto de excentricidad

Definición Sea $G = (V, E)$ un grafo conexo y $(a) \in V$. Se llama excentricidad del vértice (a) , al valor máximo de las longitudes mínimas de los caminos desde (a) a los demás nodas de G . Al valor mínimo de todas las excentricidades en G , se le denomina radio del grafo. Al subgrafo que contiene los vértices donde ocurre la excentricidad mínima, se le define como el centro de G . Por otra parte, al valor máximo de las excentricidades se le llama diámetro del grafo.

Ejemplo Utilizando el software Mathematica, encuentre el centro, el radio y el diámetro de distintos grafos conexos pseudoaleatorios.

En el software: ✓

```
GrafoSAconexo [n_] := Module[{d}, G = RandomGraph[{n, n}],  
VertexLabels -> "Name", ImagePadding -> 10];
```

```
While[ConnectedGraphQ[G] == False,  
G = RandomGraph[{n, n}], VertexLabels -> "Name",  
ImagePadding -> 10]; Return[G]
```

```
→ For[i = 1, i ≤ 10, LE = {}; Centro = {};
```

```
Print[G = GrafoSAconexo[n]]; ✓ ↓ ↓
```

```
→ For[j = 1, j ≤ 20, LE = Append[LE, VertexEccentricity[G, j]]];  
j++]; vmin = Min[LE]; vmax = Max[LE];
```

```
→ For[h = 1, h ≤ Length[LE],
```

```
If[vmin == LE[[h]], Centro = Append[Centro, h]; h++]; ✓
```

```
→ Print["El centro del grafo anterior es el subgrafo con vértices "  
Centro, " su radio es " , vmin, " y su diámetro corresponde a "  
vmax]; i++]; ↑
```

Por otra parte: Graph Radius Graph Diameter

Graph Center

(5)

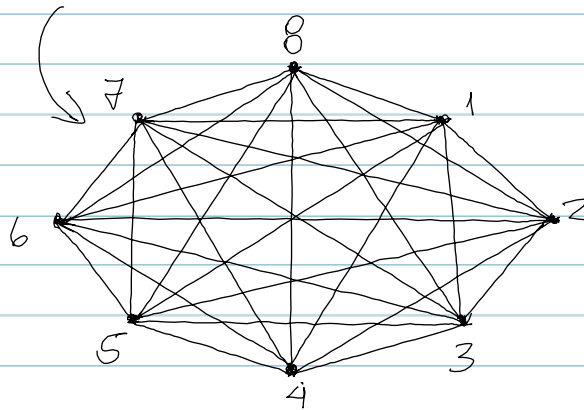
Tipos de Grafos

Mathematica posee múltiples comandos que construyen tipos especiales de grafos.

- Complete Graph: genera un grafo completo. (K_n)

Por ejemplo:

Complete Graph [$\{8\}$ VertexLabels \rightarrow "Name", ImagePadding $\rightarrow 10$]



$$m = \frac{n(n-1)}{2}$$

Complete Q [$\{5\}$]

Por ejemplo:

(K_1)

<< Combinatorica

$n = \{1, 2, 3, 4\}$, $\{1, 2, 3\}$, $\{1, 3, 4\}$, $\{2, 3, 4\}$, $\{1, 2, 4\}$, $\{1, 3, 4\}$, $\{2, 3, 4\}$;

ShowGraph [$\{5\}$ = setGraphOptions [$\{From Unordered Pairs [\{n\}$,

VertexColor \rightarrow Black, EdgeColor \rightarrow Thick],

VertexLabel \rightarrow True, PlotRange \rightarrow 0.1]

CompleteQ [$\{5\}$]

Quit [$\{5\}$]

True.

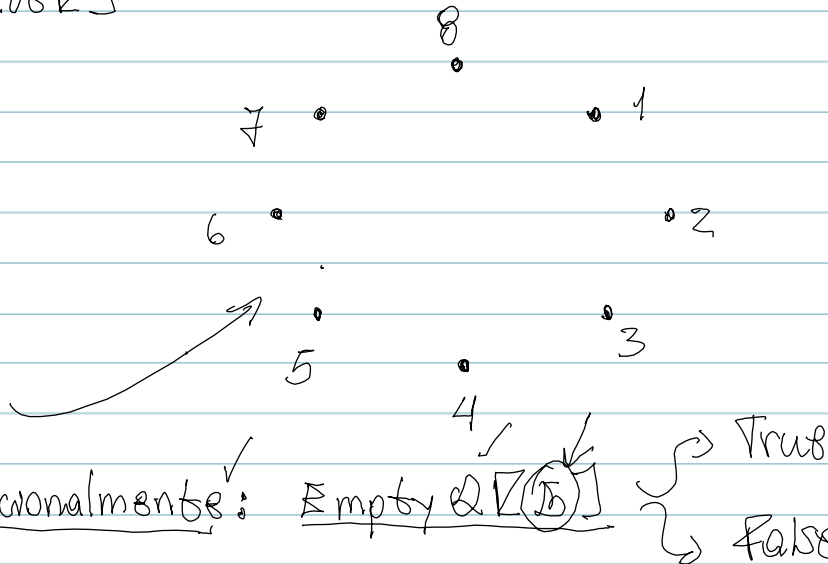
- Empty Graph: se llama grafo discreto. D_n

for exampl:

Empty Graph [8]

<<Combinatorica

Show Graph [Empty Graph [8], Vertex Number -> True,
 PlotRange -> 0.1]
 Quit[]

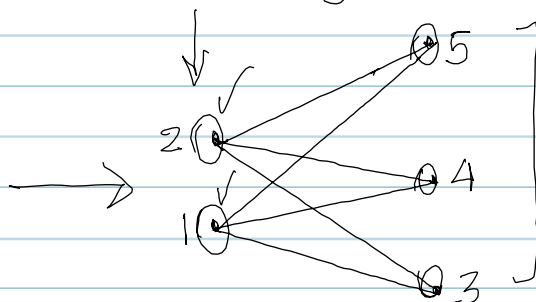


- Complete K Partite Graph: construye un grafo bipartito completo. usa el paquete "combinatorica".

$K_{m,m} \rightarrow K_{2,3}$

<<Combinatorica

Show Graph [Complete K Partite Graph [2,3],
 Vertex Number -> True, PlotRange -> 0.1]
 Quit[]



Cantidad (h) de aristas en un grafo bipartito completo:

→ << Combinatoria >> $[K_n, n+1]$
 Table $[d_i, i+1]$, ShowGraph $[5, \text{CompleteKPartiteGraph}[i, i+1]$
 VertexNumber \rightarrow True, PlotRange \rightarrow 0.1, Length $[Edges[5]]]$,
 $d_i, (2, 40)]$
 Quit $[]$

$$h = n \cdot m \quad \text{en } K_{n,m}$$

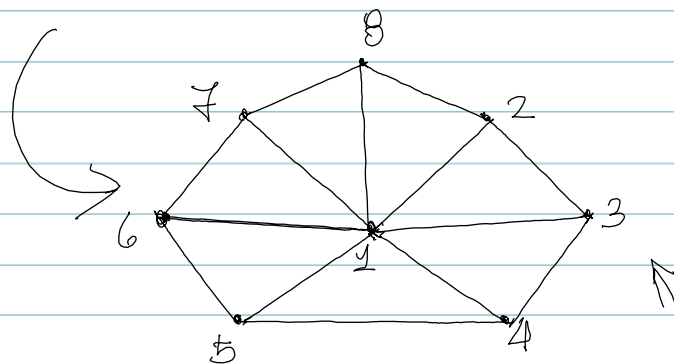
Mathematica: BipartiteGraph $[5]$

- Butterfly Graph: se denomina grafo mariposa.

→ ButterflyGraph $[3, \text{VertexLabels} \rightarrow \text{"Name"}, \text{ImagePadding} \rightarrow 10]$

- Wheel Graph: llamado grafo rueda.

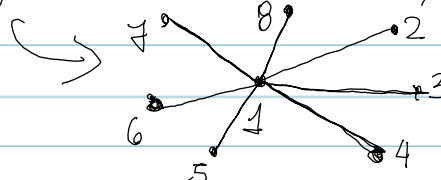
WheelGraph $[8, \text{VertexLabels} \rightarrow \text{"Name"}, \text{ImagePadding} \rightarrow 10]$



Grafo rueda

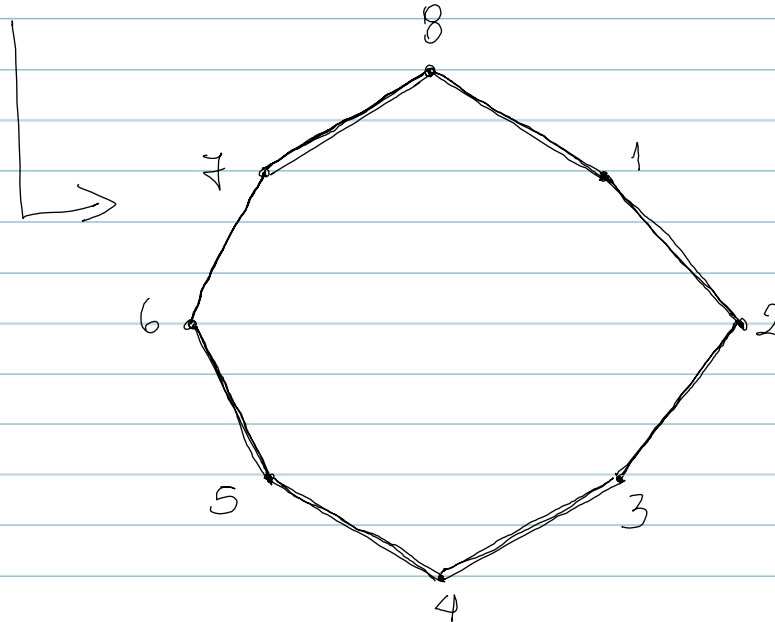
- Star Graph: genera la figura de una estrella.

StarGraph $[8, \text{VertexLabels} \rightarrow \text{"Name"}, \text{ImagePadding} \rightarrow 10]$



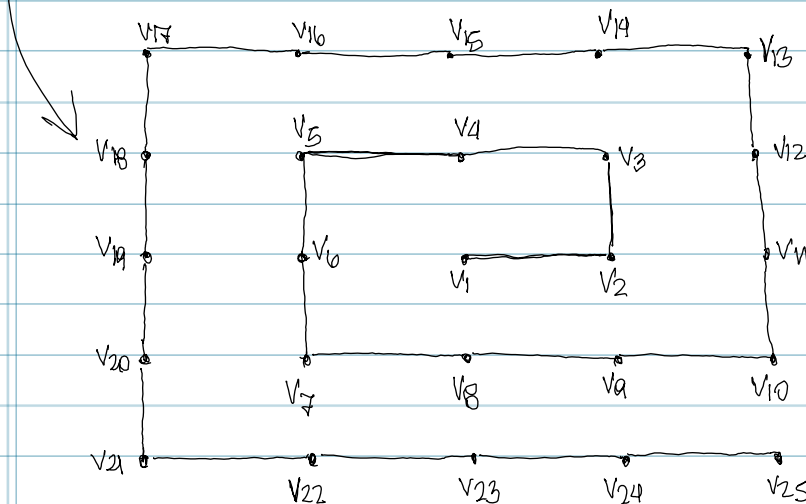
- Cycle Graph: se denomina grafo círculo.

Cycle Graph [8, Vertex Labels \rightarrow "Name", Image Padding \rightarrow 10]



- Path Graph: llamado grafo camino.

Path Graph [Table [i, d[i, 1, 25]],
 Vertex Labels \rightarrow Table [i \rightarrow (V_i), d[i, 1, 25]], Image Padding \rightarrow 10]

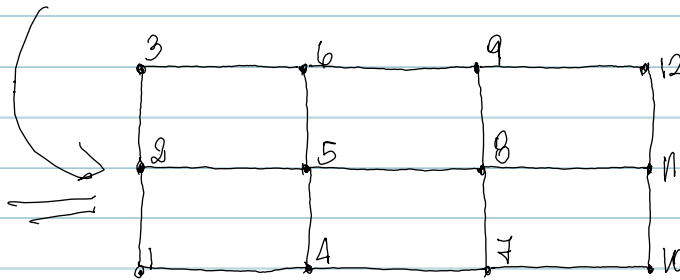


- Grid Graph: denominado grafo cuadrícula

(n) y (m)

Por ejemplo:

Grid Graph [2,3,4], Vertex Labels → "Name", ImagePadding → 10]

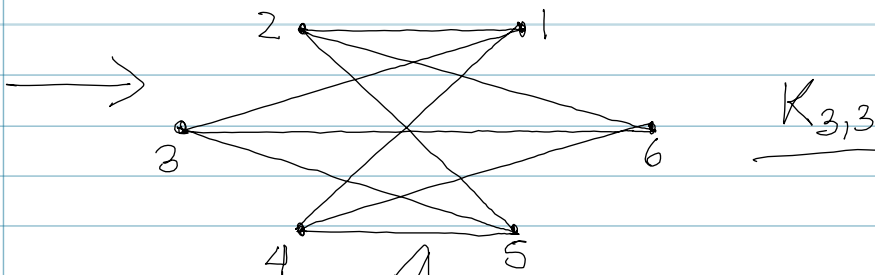


- Regular Graph: construye de forma pseudoaleatoria si es posible un grafo regular.

(K), n

<< Combinatorica

ShowGraph [RegularGraph [3,6], Vertex Number → True,
 PlotRange → 0.1]
 Quit[]



Regular[2]

<< Combinatorica
 ShowGraph [CompleteKPartiteGraph [2,4],
 Vertex Number → True, PlotRange → 0.1]
 Regular [5] false

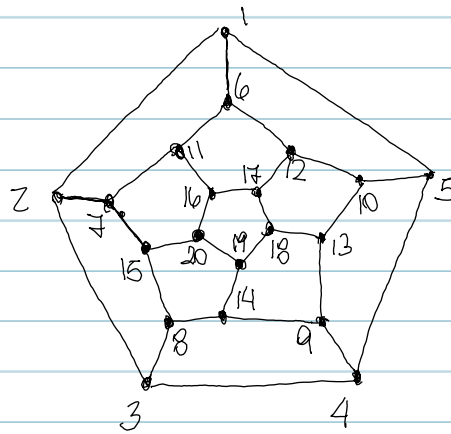
- DodecahedralGraph; llamado grafo dodecaedro.

<< Combinatorica ✓

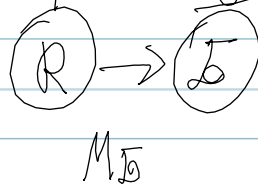
ShowGraph [DodecahedralGraph, VertexNumber → True,

PlotRange → 0.1]

Quit []]



Representaciones para un grafo



Mathematica:

Adjacency Matrix
 To Adjacency Matrix ✓

Ejemplo

Halle la matriz de adyacencia del grafo dodecaedro.

Al utilizar el software:

⟨⟨ Combinatoria ⟩⟩

ShowGraph [J = Dodecahedral Graph, Vertex Number → True,
 PlotRange → 0.1];

MatrixForm [ToAdjacencyMatrix [J]]

Quit [J]

→ Ejemplo Determine la matriz de adyacencia de 10 grafos
 simples no dirigidos pseudoaleatorios con 5 vértices y 4 lados.
 ¿Cuáles características tienen en común las matrices?

En Mathematica tenemos:

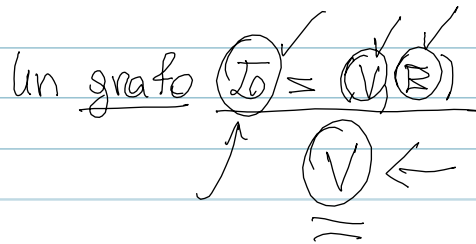
LJ = RandomGraph [UniformGraphDistribution [5, 4], {10},
 VertexLabels → "Name", ImagePadding → 10];

For [i = 1, i ≤ Length [LJ], LJ = LJ[[i]]];

I = MatrixForm [AdjacencyMatrix [LJ]]; Print [LJ];

Print ["La matriz de adyacencia del grafo anterior es: ", LJ, "\n"]

1. son simétricas $(M_{ij})^t = M_{ji}$ (R) 'a' ↔ 'b'
2. Diagonal principal (i, i) ○



Ejemplo. Considere el mismo enunciado del ejemplo anterior tomando los grafos pseudoaleatorios, dirigidos.

RandomGraph : DirectedEdges \rightarrow True.

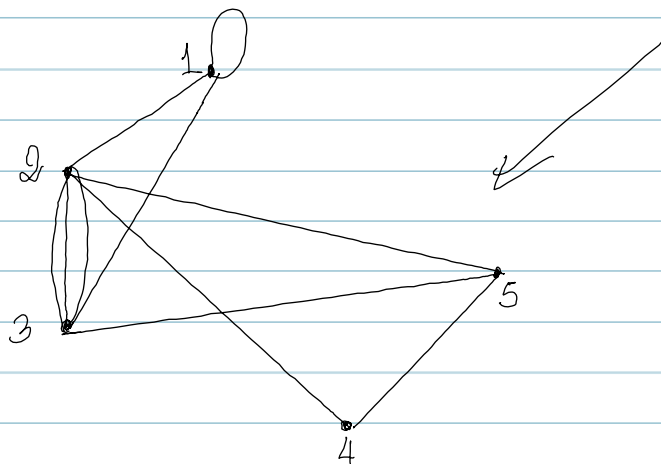
M_G

Si un grafo G incluye lados múltiples, la matriz de adyacencia M_G deja de ser una matriz booleana.

$$\sum_{i,j} v_{ij} > K \rightarrow (i,j), M_G$$



Ejemplo Encuentre la matriz de adyacencia del grafo



In Mathematica:

<<Combinatorica

$n = \{1,1,1\}, \{1,2,2\}, \{1,3,3\}, \{2,2,3\}, \{2,2,3\}, \{2,2,3\}, \{2,2,4\}, \{4,5\}, \{5,3\}, \{5,2\};$

ShowGraph[

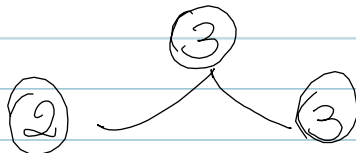
GraphOptions[FromUnorderedPairs[n], VertexColor->Black,
 EdgeColor->Thick, VertexLabel->True, PlotRange->0.1]

MatrixForm[ToAdjacencyMatrix[n]]

Quit[]

Out[]

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 3 & 1 & 1 \\ 1 & 3 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$



$\mathbb{Z}_5 \rightarrow M_5$
 $M_5 \rightarrow \mathbb{Z}_5$

Adjacency Graph
 From Adjacency Matrix ✓

\mathbb{Z}_5

Ejemplo Dados los siguientes matrices de adyacencia represente con Mathematica cada grafo.

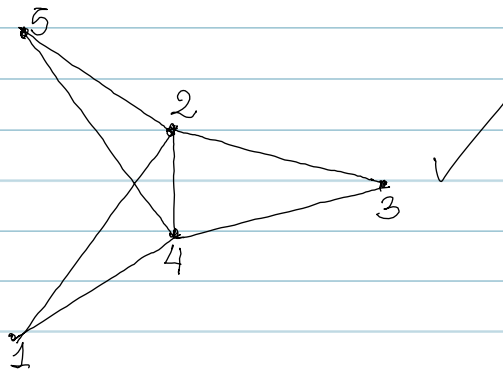
$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \checkmark$$

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \checkmark$$

- Por medio del comando Adjacency Graph:

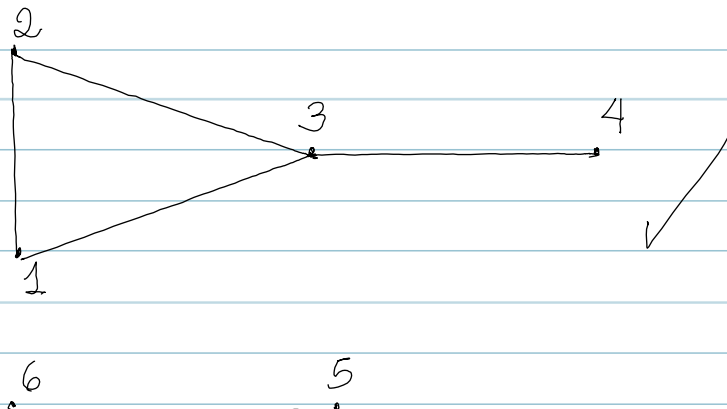
$$M_{10} = \{ \{ 0, 1, 0, 1, 0 \}, \{ 1, 0, 1, 1, 1 \}, \{ 0, 1, 0, 1, 0 \}, \\ \{ 1, 1, 1, 0, 1 \}, \{ 0, 1, 0, 1, 0 \} \};$$

Adjacency Graph [M_{10} , VertexLabels → "Name", ImagePadding → 10]



$$M_{10} = \{ \{ 0, 1, 1, 0, 0, 0 \}, \{ 1, 0, 1, 0, 0, 0 \}, \{ 1, 1, 0, 1, 0, 0 \}, \\ \{ 0, 0, 1, 0, 0, 0 \}, \{ 0, 0, 0, 0, 0, 1 \}, \{ 0, 0, 0, 0, 1, 0 \} \};$$

Adjacency Graph [M_{10} , VertexLabels → "Name", ImagePadding → 10]



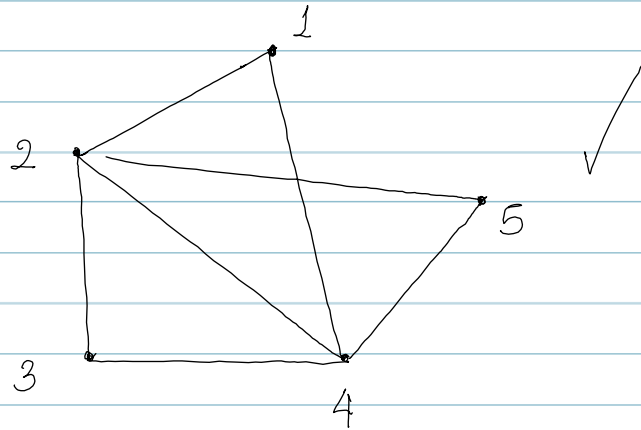
- con From Adjacency Matrix:

◀ Combinatorica

$$M_{10} = \{ \{ 0, 1, 0, 1, 0 \}, \{ 1, 0, 1, 1, 1 \}, \{ 0, 1, 0, 1, 0 \}, \{ 1, 1, 1, 0, 1 \}, \\ \{ 0, 1, 0, 1, 0 \} \};$$

Show Graph [From Adjacency Matrix [M_{10}], VertexNumber → True,
Plot Range → 0.1]

Quit []

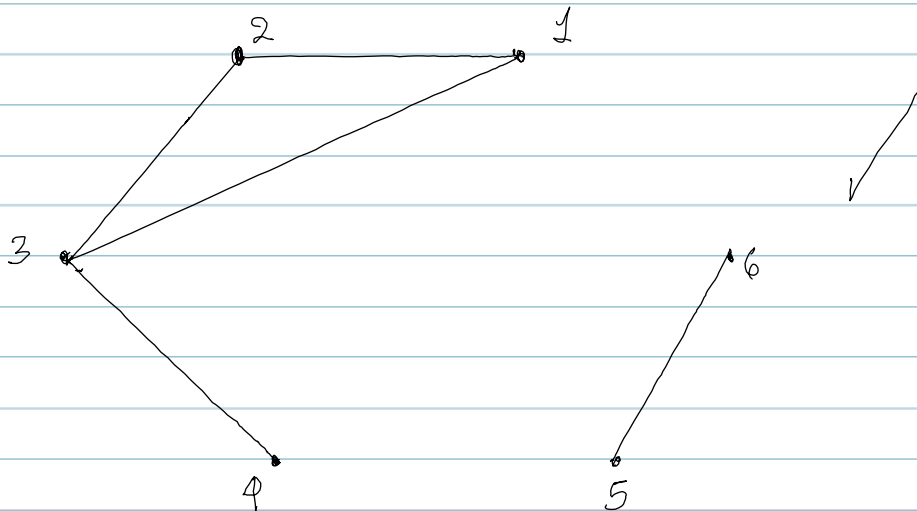


Combinatorica

$M_5 = \{ \{0, 1, 1, 0, 0, 0\}, \{1, 0, 1, 0, 0, 0\}, \{1, 1, 1, 0, 1, 0, 0\}, \{0, 0, 1, 0, 0, 0\}, \{0, 0, 0, 0, 1, 0\}, \{0, 0, 0, 0, 1, 0\} \}$

Show Graph [From Adjacency Matrix $[M_5]$, Vertex Number \rightarrow True,
 PlotRange $\rightarrow 0.1]$

Quot 2.]



Teorema Sea $G=(V,E)$ un grafo simple, $V=\{v_1, v_2, \dots, v_n\}$ y M_G la matriz de adyacencia de G en el orden del conjunto V . $(M_G)^K = (a_{ij})$, con $K \in \mathbb{N}$ es una matriz donde la entrada a_{ij} representa el número de rutas (con repetición de aristas) de longitud K , del nodo v_i al nodo v_j en G .

$$(M_G)^K \rightarrow \text{Matrix Power } [M_G, K]$$

Ejemplo Desarrolle con ayuda de software, una función que determine el número de rutas de longitud K entre dos vértices a y b , sobre un grafo simple G . Suponga que se permite la repetición de aristas en las trayectorias.

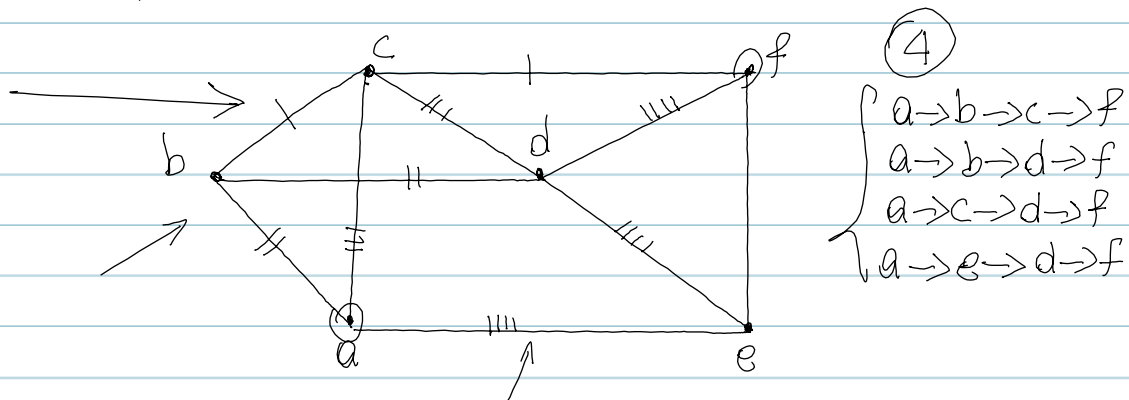
Se asume como parámetros de la función: G , K , " a " y " b ". Luego:

```

CantRutasK [ $G$  - Graph,  $K$  - ,  $a$  - ,  $b$  - ] :=
Module [  $\{ \}$ ,  $L = \text{VertexList}[G]$ ;
→ IF [MemberQ [ $L$ ,  $a$ ] == True && MemberQ [ $L$ ,  $b$ ] == True,
→  $T = \text{AdjacencyMatrix}[G]$ ;  $P = \text{MatrixPower}[T, K]$ ;
→ For [ $i = 1$ ,  $i \leq \text{VertexCount}[G]$ , IF [ $a == L[[i]]$ ,  $posa = i$ ];
IF [ $b == L[[i]]$ ,  $posb = i$ ];  $i++$ ];
Print ["La cantidad de rutas con repeticiones en las aristas del
nodo ",  $a$ , " al nodo ",  $b$ , " es: ",  $P[[posa, posb]]$ ];
Print ["Alguno de los nodos no forma parte del grafo "]]]

```

Por ejemplo, si se corre esta función en el grafo: $K=3$



Definición, Sea $G = (V, E)$ un grafo no dirigido con $V = \{v_1, v_2, \dots, v_n\}$ y $E = \{e_1, e_2, \dots, e_m\}$. Se llama matriz de incidencia de G en el orden de los elementos de V y E , a la matriz denotada M_G , $M_G = (m_{ij})$ de tamaño n por m , tal que:

$$m_{ij} = \begin{cases} 1 & \text{si la arista } e_j \text{ es incidente sobre el vértice } v_i \\ 0 & \text{en caso contrario} \end{cases}$$

Fila i -ésima $\rightarrow v_i$
 G

En el software Mathematica:

G \rightarrow IncidenceMatrix $\left\{ \begin{array}{l} \text{Graph } G \\ \text{Show Graph} \end{array} \right. (i, j) \quad v_i \rightarrow e_j$

Ejemplo Halle la matriz de incidencia del grafo propuesto en la página anterior.

En Mathematica:

$\rightarrow G = \text{Graph}[\text{Table}[a \leftrightarrow b, a \leftrightarrow c, b \leftrightarrow c, c \leftrightarrow d, d \leftrightarrow b, e \leftrightarrow a, e \leftrightarrow f, f \leftrightarrow d, f \leftrightarrow c, e \leftrightarrow d], \text{VertexLabels} \rightarrow \text{"Name"}, \text{ImagePadding} \rightarrow 10];$
Matrix Form \rightarrow IncidenceMatrix $[G]$

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Otra forma de visualización de la matriz de incidencia, reside en construir una tabla.

En el software:

```

g = Graph[{a -> b, a -> c, b -> c, c -> d, d -> b, e -> a,
e -> f, f -> d, f -> c, e -> d}, VertexLabels -> "Name",
ImagePadding -> 10];

```

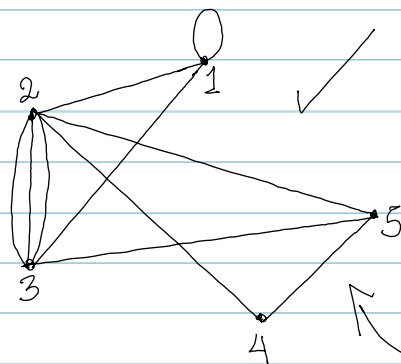
```

TableForm[Normal@IncidenceMatrix[g], TableHeadings ->
{Style[#, Black] &/@ VertexList[g], Style[#, Black] &/@
EdgeList[g]}]

```

	<u>a -> b</u>	<u>a -> c</u>	<u>b -> c</u>	<u>c -> d</u>	<u>d -> b</u>	<u>e -> a</u>	<u>e -> f</u>
→ a	1	1	0	0	0	1	0
→ b	1	0	1	0	1	0	0
→ c	0	1	1	1	0	0	0
→ d	0	0	0	1	1	0	0
→ e	0	0	0	0	0	1	1
→ f	0	0	0	0	0	0	1

Ejemplo Determine la matriz de incidencia del grafo:



<< Combinatoria

$h = \{d\{1,1\}, d\{1,2\}, d\{1,3\}, d\{2,3\}, d\{2,3\},$
 $\{2,3\}, \{2,4\}, \{4,5\}, \{5,3\}, \{5,2\}\};$

ShowGraph[

$g = \text{SetGraphOptions}[$

FromUnorderedPairs[h], VertexColor

→ Black, EdgeColor → Thick,

VertexLabel → True, PlotRange → {0,1};

MatrixForm[IncidenceMatrix[g]]

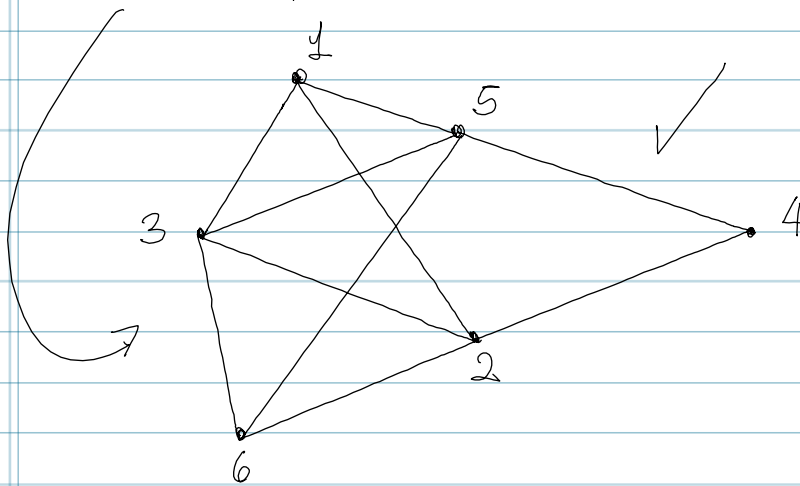
$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad (3)$$

Ejemplo Represente con Mathematica el grafo dado por la matriz de incidencia

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\underline{W} = \{ \{ 1, 1, 0, 0, 0, 0, 0, 1, 0, 0 \}, \{ 1, 0, 1, 1, 0, 1, 0, 0, 0, 0 \}, \\ \{ 0, 1, 1, 0, 1, 0, 0, 0, 0, 1 \}, \{ 0, 0, 0, 1, 0, 0, 0, 0, 1, 0 \}, \\ \{ 0, 0, 0, 0, 1, 0, 1, 1, 1, 0 \}, \{ 0, 0, 0, 0, 0, 1, 1, 0, 0, 1 \} \};$$

Incidence Graph [W], VertexLabels \rightarrow "Name", ImagePadding \rightarrow 10]



Ta se comento la funcionalidad:

Weighted Adjacency Matrix
Weighted Adjacency Graph

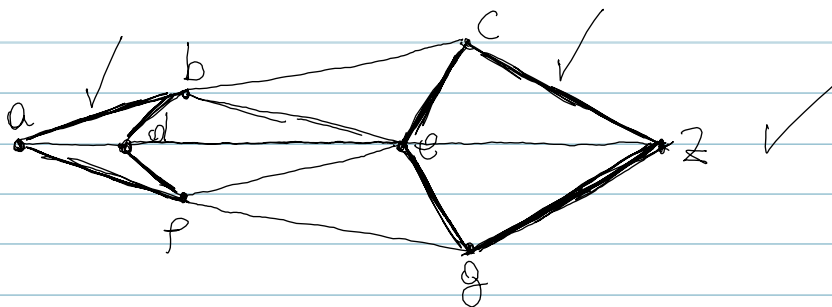
Circuitos en un grafo

Definición Sea $G = (V, E)$ un grafo. A cualquier ruta sobre G que inicie y finalice en el mismo vértice se le llama circuito. Si un circuito no repite nodos se dice que es simple. Si un circuito contiene todas las aristas de G se denomina circuito de Euler. Si un circuito contiene todos los vértices de G sin repetición de nodos, se llama circuito de Hamilton. Una ruta que incluye todos los lados de G sin ser un circuito, se define como un camino de Euler o trayectoria euleriana. Cualquier trayectoria que no sea un circuito y que contenga una única vez todos los vértices de G , se llama ruta de Hamilton o camino hamiltoniano.



Ejemplo

Graph [d a--b, a--d, a--f, b--c, b--d, b--e, c--e, c--z, d--e, d--f, e--g, e--f, f--g, g--z, z--e],
 VertexLabels -> "Name", ImagePadding -> 10,
 GraphHighlight -> { d a--b, b--d, d--f, f--a, c--z, z--g, g--e, b--c }, GraphHighlightStyle -> "Thick"]



En el software Mathematica:

FindCycle [G]
 ↳ 2 f

FindH [G]
 ↳ ∞

for ejemplo:

<< Combinatorica'

$n = \{ \{1,2\}, \{1,3\}, \{2,4\}, \{2,6\}, \{3,4\}, \{3,5\}, \{4,5\}, \{4,6\} \};$

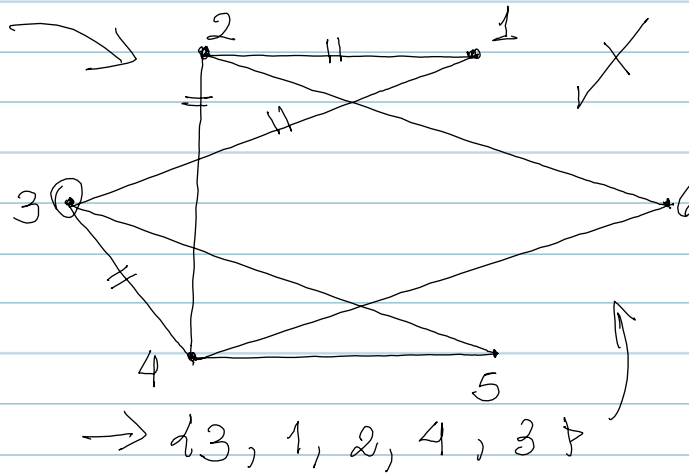
ShowGraph [5] SetGraphOptions [FromUndirectedPairs [n],
 VertexColor -> Black, EdgeColor -> Thick, VertexLabel -> True,

PlotRange -> 0.1] (j)

→ FindCycle [5]

Print [5]

Quit []



(3)

El resultado de FindCycle se puede desplegar gráficamente en Mathematica:

↓
 { v = FindCycle [5];
 ShowGraph [Highlight [5, {Partition [v, 2, 1]}],
 VertexLabel -> True, PlotRange -> 0.1]

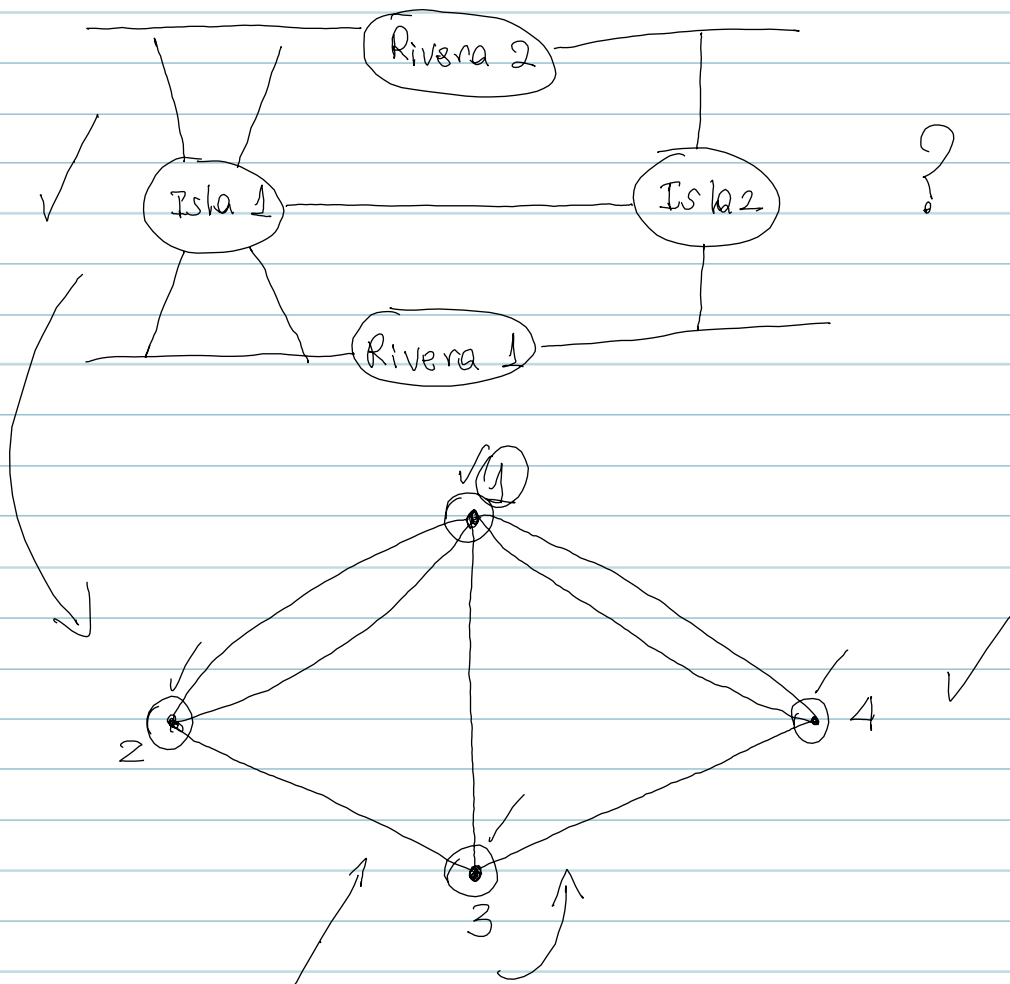
Otro comando para circuitos:

Acyclic Graph Q [5] (True)
↓
False

Circuitos de Euler ✓

Leonhard Euler ✓ → (1736)

7 puentes de Königsberg



Teorema Sea $G = (V, E)$ un grafo conexo. Si todas las valencias de los vértices de G son pares, G tiene un circuito de Euler. Además, si hay al menos un grado impar en los nodos de G , el grafo no posee un circuito de Euler.

Ejemplo Elabore un programa en Mathematica que determine si un grafo sin ciclos posee un circuito de Euler.

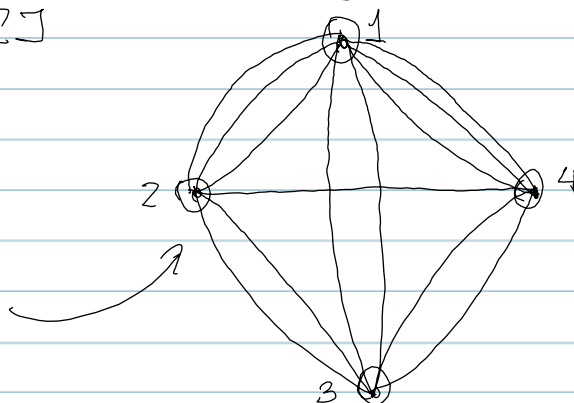
Utilizando el software:

```
<<Combinatorica'
BooleanCircuitoEuler[G_Graph]:=
Module[{v=Degrees[G], i=0},
IF[EdgeConnectivity[G]≠0,
G, For[i=1, i≤Length[v],
IF[OddQ[v[[i]]]==True, Print["False"]; i=i+1;
Break[]]; IF[j==0, Print["True"]];
Print["False"]]]]
```

Por ejemplo

$h = \{ \{1,2\}, \{1,2\}, \{1,2\}, \{1,4\}, \{1,4\}, \{1,4\}, \{1,3\}, \{1,3\}, \{2,3\}, \{2,3\}, \{2,4\}, \{3,4\}, \{3,4\} \}$.

```
ShowGraph[G=SetGraphOptions[FromUnorderedPairs[h],
VertexColor→Black, EdgeColor→Thick,
VertexLabel→True, PlotRange→0.1]
BooleanCircuitoEuler[G]
Quit[]]
```



True

Más directamente:

\checkmark Eulerian Q [5] \checkmark
 \checkmark Eulerian Graph Q [5] }

Ejemplo Conjetura con ayuda de software, ¿qué condiciones debe cumplir un grafo completo K_n para tener circuitos de Euler? ¿cuáles condiciones se deben satisfacer en un grafo bipartito completo $K_{n,m}$?

Para K_n :

\rightarrow For $[i=3, i \leq 12, \text{Go} = \text{Complete Graph}[i];$
 Print ["Para el orden ", i , " el valor lógico sobre la existencia de un circuito de Euler es: ", EulerianGraphQ[5]]; $i++$]

n es impar

Para $K_{n,m}$:

\hookleftarrow Combinatoria
 For $[i=1, i \leq 6,$
 For $[j=1, j \leq 6, \text{Go} = \text{Complete Bipartite Graph}[i, j];$
 Print ["Para el orden ", i , " x ", j , " el valor lógico sobre la existencia de un circuito de Euler es: ", EulerianQ[5]]; $j++$]; $i++$]
 Quit []

(26) $K_{i,j}$

Teorema (Algoritmo de Fleury) Sea $G = (V, E)$ un grafo conexo con grados pares en todos sus vértices. Un circuito de Euler sobre G se forma así:

1. Se parte de un vértice v cualquiera de G y se agrega a L una arista inicial e con extremo v , que no sea un puente (un puente es un lado del grafo, que al ser eliminado, lo transforma en disconexo), sea el conjunto de aristas seleccionables (CAS), $CAS = E - \{e\}$.

2. Si $CAS = \emptyset$, se ha finalizado, L contiene el circuito de Euler.

3. Se escoge un nuevo lado e_N para el circuito a construir, siempre y cuando e_N no sea un puente y sea adyacente a la última arista de L . Si la única arista elegible e_N es un puente, se selecciona.

4. Sea $CAS = CAS - \{e_N\}$ y $L = L \cup \{e_N\}$, quedando e_N como último elemento de L . Vaya al paso dos.

(L) ✓

Ejemplo Desarrolle una implementación en Mathematica del algoritmo de Fleury sobre un grafo simple.

◻ Combinatoria

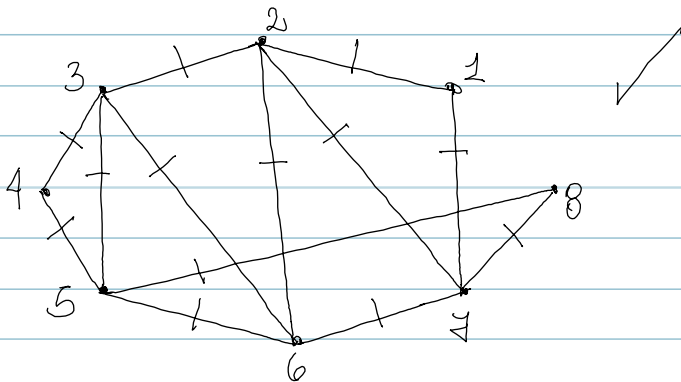
```
CircuitoEuleriano[G_Graph]:=Module[{GL=G},
  If[EulerianQ[G]==True, CAS=Edges[GL],
  L=Flatten[CAS[[1]]]; GL=DeleteEdge[GL, CAS[[1]]];
  CAS=Delete[CAS, Position[CAS, CAS[[1]]]];
  While[Length[Edges[GL]]!=0, CAI={};
  For[i=1, i<Length[CAS], w=Last[L],
  -> If[w==CAS[[i,1]] || w==CAS[[i,2]],
  CAI=Append[CAI, CAS[[i]]]];
  L=CAI;
```

```

CAS = Delete Duplicates [CAS]; i++;
For [i=1, i ≤ Length [CAS],
→ IF [MemberQ [Bridges [EL], CAS[[i]]] == False,
    EL = Delete Edge [EL, CAS[[i]]];
    IF [CAS[[i, 2]] ≠ w], (L) = Append [L, CAS[[i, 2]]],
    L = Append [L, CAS[[i, 1]]];
    CAS = Delete [CAS, Position [CAS, CAS[[i]]]]; Break[];
IF [i = Length [CAS], (EL) = Delete Edge [EL, CAS[[i]]];
    IF [CAS[[i, 2]] ≠ w, L = Append [L, CAS[[i, 2]]],
    (L) = Append [L, CAS[[i, 1]]];
    CAS = Delete [CAS, Position [CAS, CAS[[i]]]]; i++;
Print ["Un circuito de Euler mediante una representación
de nodos es: ", L];
Print ["El mismo circuito representado a través de aristas es: ",
Partition[L, 2, 1]]; Print ["El grafo no tiene circuitos de Euler"]

```

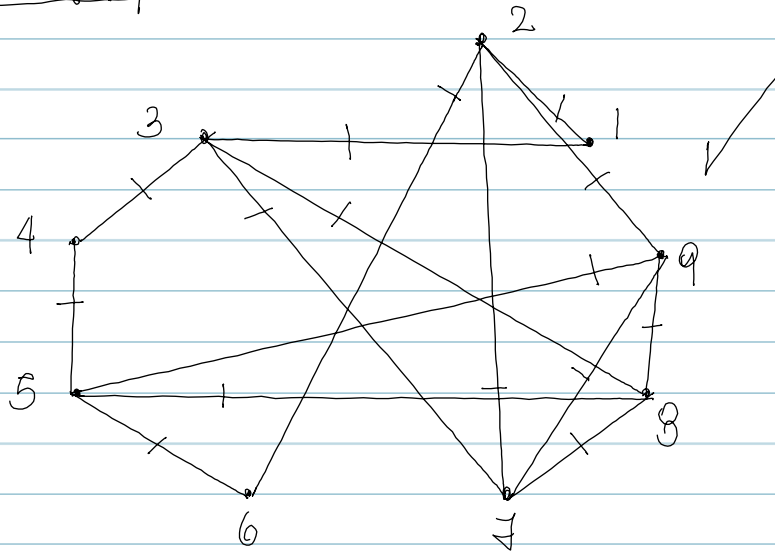
Ejemplo:



Se obtiene

→ {1, 2, 3, 4, 5, 3, 6, 2, 7, 6, 5, 8, 7, 1}

Otro ejemplo:



✓ 1,2, 2,6, 6,5, 5,4, 4,3, 3,7, 7,2, 2,9, 9,5, 5,8, 8,7, 7,9, 9,8, 8,3, 3,1, 1,2

En Mathematica:

→ EulerianCycle ✓

→ FindEulerianCycle ✓

↳ FindEulerianCycle [K, K]

(K)

Trayectorias de Euler

Teorema Sea $G = (V, E)$ un grafo conexo. Si G tiene exactamente dos nodos a y b de valencia impar, G posee una ruta de Euler que une el vértice a con el nodo b . Si en G hay más o menos de dos vértices con grado impar entonces G no contiene ningún camino euleriano.

Ejemplo Elabore con Mathematica una función booleana que determine si dado un grafo sin ciclos ¿este contiene una trayectoria de Euler?

En el software:

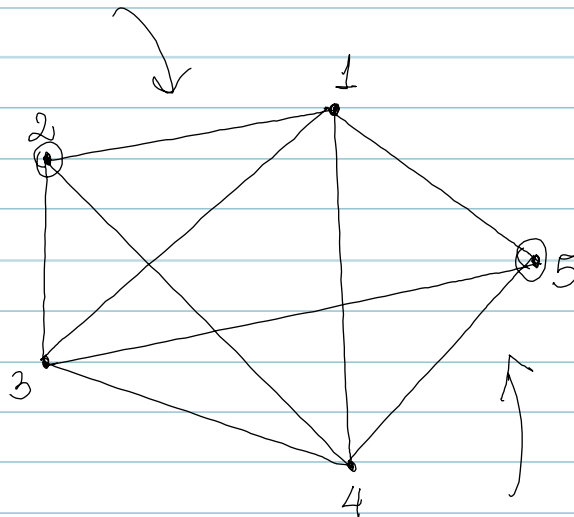
<< Combinatoria

BooleanRutaEuler[G_Graph] := Module[{dV = Degrees[G], j = 0},

If[EdgeConnectivity[G] > 0,

For[i = 1, i <= Length[dV], If[OddQ[dV[[i]]] == True, j++, i++]; If[j == 2, Print["True"], Print["False"]], Print["False"]]]

Por ejemplo:



BooleanRutaEuler[G] → "True"

Circuitos de Hamilton

Teorema. Sea $G = (V, E)$ un grafo simple y conexo, con n nodos y m lados, $n \geq 2$, luego:

- (1) (Propiedad de Ore) Si para todo par de vértices no adyacentes a y b de V , se cumple: $\deg(a) + \deg(b) \geq n$ entonces G tiene un circuito de Hamilton.
- (2) (Propiedad de Dirac) Si para todo vértice a de V , $\deg(a) \geq \frac{n}{2}$ entonces G posee un circuito hamiltoniano.
- (3) Si $m \geq \frac{n^2 - 3n + 6}{2}$ entonces G contiene un circuito de Hamilton.

Ejemplo Elabore una rutina en Mathematica por medio del teorema anterior, donde al recibir un grafo indique si éste tiene un circuito de Hamilton, o bien, no hay criterio.

En el software:

ExisteCircuitoHamilton $[G_Graph] :=$

```

→ IF [SimpleGraphQ[G]] == True && ConnectedGraphQ[G] == True &&
VertexCount[G] > 2,
→ Module[{vertices = VertexList[G], v = VertexDegree[G],
n = VertexCount[G], m = EdgeCount[G], prop1 = 1, h = 1, prop2 = 1,
prop3 = 1}, For[i = 1, i <= VertexCount[G], For[j = i, j <= n,
→ IF [i != j && EdgeQ[G, vertices[[i]], vertices[[j]]] == False,
h = 0; IF [VertexDegree[G, vertices[[i]]] +
VertexDegree[G, vertices[[j]]] < n, prop1 = 0; Break[];
Break[]]; j++]; i++];
→ For[i = 1, i <= Length[v], IF [v[[i]] < n/2, prop2 = 0; i++];
→ IF [m < (n^2 - 3n + 6)/2, prop3 = 0];

```

✓
 IF $[\text{prop1} == 1 \text{ AND } h == 1,$

Print $["\text{No se puede verificar la propiedad de Ore pues todos los vértices son adyacentes entre sí"}];$

IF $[\text{prop1} == 1 \text{ AND } h == 0,$ Print $["\text{Se satisface la propiedad de Ore}"];];$

IF $[\text{prop1} == 0,$ Print $["\text{No se satisface la propiedad de Ore}"];];$

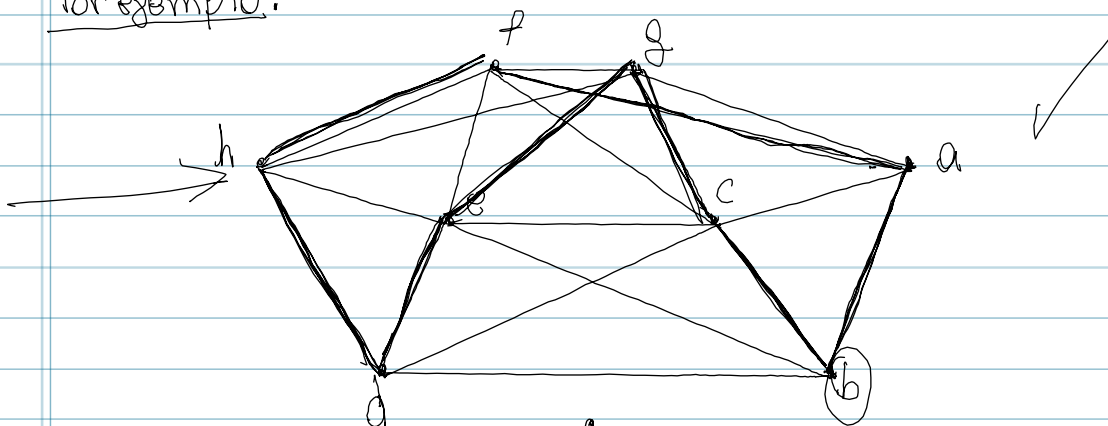
→ IF $[\text{prop2} == 0,$ Print $["\text{No se cumple la propiedad de Dirac}"];]$
 Print $["\text{Se cumple la propiedad de Dirac}"];];$

→ IF $[\text{prop3} == 0,$ Print $["\text{No se satisface la propiedad 3}"];]$
 Print $["\text{Se satisface la propiedad 3}"];];$

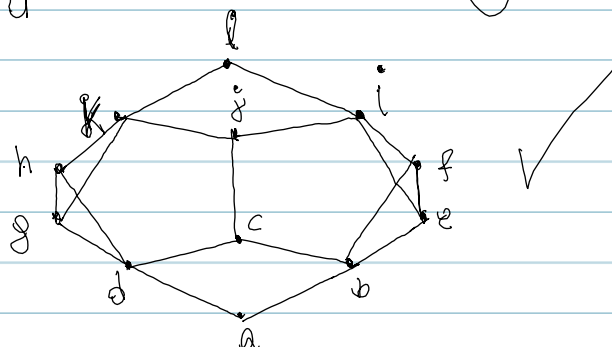
→ IF $[\text{prop1} == 1 \text{ AND } \text{prop2} == 1 \text{ AND } \text{prop3} == 1,$ Print $["\text{Existe un circuito de Hamilton}"];]$ Print $["\text{No hay criterio}"];];]$

Print $["\text{El grafo no es simple, no es conexo o hay una cantidad insuficiente de vértices}"];];$

Por ejemplo:



Otro ejemplo:



De forma complementaria:

HamiltonianQ
HamiltonianGraphQ } True - False

Por ejemplo:

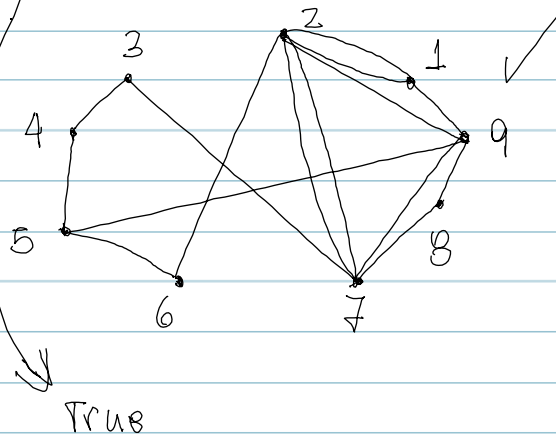
<<Combinatorica

$n = \{ \{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{1, 6\}, \{1, 7\}, \{1, 8\}, \{1, 9\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{2, 6\}, \{2, 7\}, \{2, 8\}, \{2, 9\}, \{3, 4\}, \{3, 5\}, \{3, 6\}, \{3, 7\}, \{3, 8\}, \{3, 9\}, \{4, 5\}, \{4, 6\}, \{4, 7\}, \{4, 8\}, \{4, 9\}, \{5, 6\}, \{5, 7\}, \{5, 8\}, \{5, 9\}, \{6, 7\}, \{6, 8\}, \{6, 9\}, \{7, 8\}, \{7, 9\}, \{8, 9\} \}$

ShowGraph[\mathcal{G} = SetGraphOptions[FromUnorderedPairs[n], VertexColor -> Black, EdgeColor -> Thick], VertexLabel -> True, PlotRange -> 0.1]

HamiltonianQ[\mathcal{G}]

Quit[]



Ejemplo Conjetura cuál condición debe cumplir $K_{n,m}$ para contener un circuito de Hamilton.

<<Combinatorica

→ For [i = 1, i ≤ 5], For [j = 1, j ≤ 5], \mathcal{G} = CompleteKPartiteGraph[i, j];
 Print["Para el orden ", i, "x", j, " el valor lógico sobre la existencia de un circuito de Hamilton es: ", HamiltonianQ[\mathcal{G}]]; j++; i++;
 Quit[]

$$n = m, \underline{n \geq 2}$$

Para retornar circuitos de Hamilton con Mathematica:

HamiltonianCycle → All ✓
FindHamiltonianCycle ✓

Por ejemplo:

FindHamiltonianCycle[5]

→ $d \rightarrow b \rightarrow a, a \rightarrow f, f \rightarrow h, h \rightarrow d, d \rightarrow e, e \rightarrow g, g \rightarrow c,$
 $c \rightarrow b$ ✓

Rutas de Hamilton

HamiltonianPath [5] → All ✓

Problema del agente viajero ✓

Traveling Salesman [5] ✓

Ejemplo Resuelva el problema del agente viajero sobre el grafo dodecaedro.

En Mathematica:

◀ Combinatoria

⑤ = DodecahedronGraph;

vs TravelingSalesman[5]

ShowGraph [Highlight [5], Partition [v, 2, 1] ✓]

VertexLabel → True, PlotRange → 0.1]

Quit[]

→ 1, 2, 3, 4, 5, 10, 12, 17, 16, 20, 19, 18, 13, 9, 14,
8, 15, 7, 11, 6 ✓

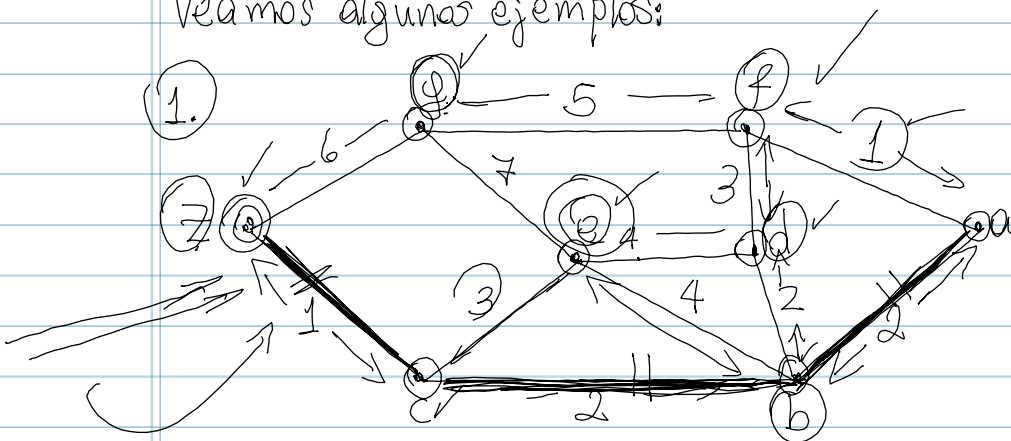
Algoritmo del camino más corto o algoritmo de Dijkstra

Teorema (Algoritmo de Dijkstra) Sea $G = (V, E)$ un grafo conexo y ponderado y $MP = (c_{ij})$ la matriz de adyacencia de pesos de G en el orden de los elementos de V . Si se desea obtener la longitud de un camino más corto o de peso mínimo entre dos nodos $"a"$ y $"z"$ de G , se realizan los siguientes pasos:

1. [Inicialización] Sea $\text{Marca}(a) = 0$ y $\text{Marca}(v) = \infty, \forall v, v \in V$. Sea el conjunto $LPN = V$.
2. Si $z \notin LPN$, se ha terminado y $\text{Marca}(z)$ es la longitud de un camino más corto de $"a"$ a $"z"$.
3. Se elige un vértice w de LPN , donde $\text{Marca}(w)$ tiene el valor mínimo y se actualiza LPN como: $LPN = LPN - \{w\}$.
4. La marca de cada nodo r adyacente a w , se renueva así:
$$\text{Marca}(r) = \min(\text{Marca}(r), \text{Marca}(w) + (c_{ij}))$$

siendo c_{ij} la entrada de la matriz MP que une el vértice w con el nodo r . Vuelva al paso 2.

Veamos algunos ejemplos:



Inicialización

$$\text{Marca}(a) = 0$$

$$\text{Marca}(v) = \infty$$

$$\text{LPN} = \{a, b, c, d, e, f, g, z\}$$

$$\forall v, v \in \{b, c, d, e, f, g, z\}$$

- Se selecciona al nodo a , $\text{LPN} = \text{LPN} - \{a\}$

$$\text{Marca}(b) = 2, \text{ y } \text{Marca}(f) = 1$$

- Se selecciona al nodo f , $\text{LPN} = \text{LPN} - \{f\}$

$$\text{Marca}(g) = 6, \text{ Marca}(d) = 4, \text{ Marca}(b) = 2$$

- Se selecciona al nodo b , $\text{LPN} = \text{LPN} - \{b\}$

$$\text{Marca}(c) = 4, \text{ Marca}(d) = 4, \text{ Marca}(e) = 6, \text{ Marca}(g) = 6$$

- Se selecciona al nodo c , $\text{LPN} = \text{LPN} - \{c\}$

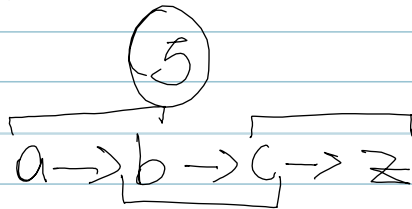
$$\text{Marca}(e) = 6, \text{ Marca}(z) = 5, \text{ Marca}(d) = 4, \text{ Marca}(g) = 6$$

- Se elige al vértice d , $\text{LPN} = \text{LPN} - \{d\}$

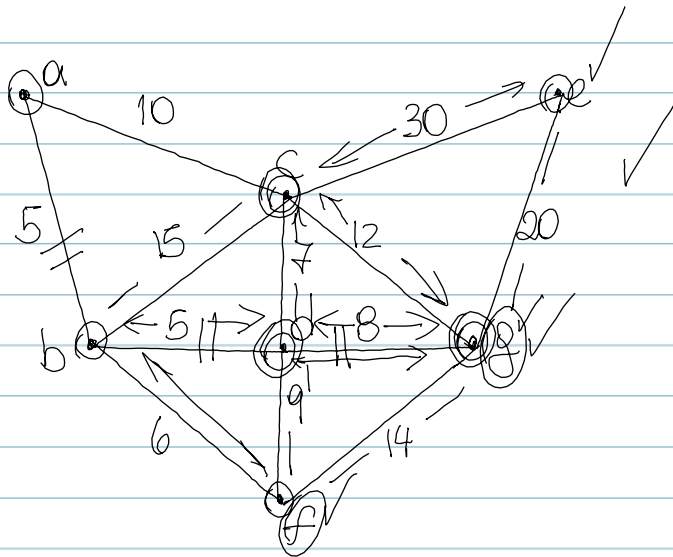
$$\text{Marca}(e) = 6, \text{ Marca}(z) = 5, \text{ Marca}(g) = 6$$

- Se selecciona al vértice z , $\text{LPN} = \text{LPN} - \{z\}$

$$\text{Marca}(g) = 6, \text{ Marca}(e) = 6$$



(2.)



Inicialización

$\text{Marca}(a) = 0$

$\text{Marca}(v) = \infty, \forall v, v \in \{b, c, d, e, f, g\}$

$\text{LPN} = \infty$

- Se selecciona al nodo a, $\text{LPN} = \text{LPN} - \{a\}$
 $\text{Marca}(b) = 5, \text{Marca}(c) = 10$
- Se elige el vértice b, $\text{LPN} = \text{LPN} - \{b\}$
 $\text{Marca}(c) = 10, \text{Marca}(d) = 10, \text{Marca}(f) = 11$
- Se selecciona al nodo c, $\text{LPN} = \text{LPN} - \{c\}$
 $\text{Marca}(e) = 40, \text{Marca}(g) = 22, \text{Marca}(d) = 10, \text{Marca}(f) = 11$
- Se selecciona al vértice d, $\text{LPN} = \text{LPN} - \{d\}$
 $\text{Marca}(g) = 18, \text{Marca}(f) = 11, \text{Marca}(e) = 40$
- Se selecciona al nodo f, $\text{LPN} = \text{LPN} - \{f\}$
 $\text{Marca}(g) = 18, \text{Marca}(e) = 40$
- Se elige al nodo g, $\text{LPN} = \text{LPN} - \{g\}$
 $\text{Marca}(e) = 38$

$a \rightarrow b \rightarrow d \rightarrow g$

Se aclara al estudiante

En el software Mathematica:

FindShortestPath ✓
ShortestPath ✓

Por ejemplo:

FindShortestPath [G, a, z]

Out[]
{d, a, b, c, z} ✓✓

Gráficamente:

$G = \text{Graph} [d \leftrightarrow b, a \leftrightarrow f, b \leftrightarrow c, b \leftrightarrow d, b \leftrightarrow e, c \leftrightarrow e, c \leftrightarrow z, d \leftrightarrow e, d \leftrightarrow f, e \leftrightarrow g, f \leftrightarrow g, g \leftrightarrow z],$
EdgeWeights $\rightarrow \{2, 1, 2, 2, 4, 3, 1, 4, 3, 7, 5, 6\},$
VertexLabels $\rightarrow \text{"Name"}, \text{ImagePadding} \rightarrow 10];$
→ RutaNodos = FindShortestPath [G, a, z];
→ RutaAristas = Partition [RutaNodos, 2, 1];
→ ConstruirRuta [RutaAristas_]:= Module [d[1] = { }], For [i = 1, i ≤ Length [RutaAristas], L = Append [L, RutaAristas [[i, 1]]] → RutaAristas [[i, 2]]]; L = Append [L, RutaAristas [[i, 2]]]; i++]; L = Append [L, RutaAristas [[i, 1]]]; L = DeleteDuplicates [L]; Return [L]]
G = HighlightGraph [G, ConstruirRuta [RutaAristas],
GraphHighlightStyle → "Thick"]

También: AllPairsShortestPath [G] $n \times n$ (i, j) ✓✓
GraphDistance [G, a, z] ✓✓
Dijkstra [G, a] ✓

For example:

<< Combinatorial

$h = \{1, 2\}, \{2, 3\}, \{3, 5\}, \{4, 3\}, \{4, 1\}, \{5, 4\}, \{5, 2\}$;

Show Graph [$G = \text{SetGraphOptions}$ [From Unordered Pairs [h],

Vertex Color \rightarrow Black, Edge Color \rightarrow Thick]

Vertex Label \rightarrow true, PlotRange \rightarrow 0.1]

$G = \text{SetEdgeWeights}$ [$G, \{5, 2, 3, 5, 8, 3, 2, 1\}$]; ✓

Dijkstra [G , 1] [2] ←

Quit []

1, 0, 1, 4, 3, 2