JULIHO CASTILLO COLMENARES

OPTIMIZACIÓN COMBINATORIA

WWW.ASIMOVIAN.ACADEMY

Índice general

1 Optimización de modelos 5

1.1 Motivación 5

2 Programación lineal 13

2.1 Introducción a la programación lineal 13

3 Teoría de Gráficas 15

3.1 Matrices 15

3.2 Teoría general de grafos 19

3.3 Digrafos 27

1 Optimización de modelos

1.1 Ejemplos de programación lineal

Problema de proyectos de inversión

Imaginemos que ocupamos el puesto de coordinador de proyectos dentro de una empresa. El gerente general de dicha empresa ha destinado 100,000 pesos para invertir en los proyectos que generen beneficios económicos a esta. Existen tres proyectos en los que se puede invertir. ¿En cuál(es) proyecto(s) debería invertir la empresa para obtener los máximos beneficios económicos?

Se tiene la siguiente información sobre los proyectos:

Nombre	Costo de Inversión	Benefició económico
Proyecto A	$50,000	$80,000
Proyecto B	$70,000	$90,000
Proyecto C	$25,000	$30,000

```
#!/usr/bin/env python
# coding: utf-8

# ## Problema de proyectos de óinversin
#
# Imaginemos que ocupamos el puesto de coordinador de
    proyectos dentro de una empresa. El gerente
# general de dicha empresa ha destinado 100,000 pesos para
    invertir en los proyectos que generen beneficios
# óeconmicos a esta. Existen tres proyectos en los que se
    puede invertir. ¿En ácul(es) proyecto(s)
# ídebera invertir la empresa para obtener los ámximos
    beneficios óeconmicos?
#
# Se tiene la siguiente óinformacin sobre los proyectos:
#
# |Nombre|Costo de óInversin|óBenefici óeconmico|
# |:----:|:----------------:|:-----------------|
# |Proyecto A| \$50,000 | \$80,000 |
# |Proyecto B| \$70,000 | \$90,000 |
# |Proyecto C| \$25,000 | \$30,000 |

from ortools.linear_solver import pywraplp
```

```
20
21 # SCIP(Solving Constraint Integer Programs)
22 solver = pywraplp.Solver.CreateSolver('SCIP')
23 x = {}
24 tags = ["A", "B", "C"]
25
26 for tag in tags:
27     x[tag] = solver.IntVar(0, 1, tag)
28
29 cost = {'A': 50_000, 'B': 70_000, 'C': 25_000}
30 utility = {'A': 80_000, 'B': 90_000, 'C': 30_000}
31 solver.Add(sum(cost[tag] * x[tag] for tag in tags) <= 100000)
32 solver.Maximize(sum(utility[tag] * x[tag] for tag in tags))
33
34 # Solve the system.
35 status = solver.Solve()
36
37 if status == pywraplp.Solver.OPTIMAL:
38     print('óSolucin:')
39     print('Valor objetivo =', solver.Objective().Value())
40     for k, v in x.items():
41         print("Variable: {}, valor:{} ".format(k, v.
    solution_value()))
42
43 else:
44     print('EL problema no tiene ósolucin óptima.')
45
46 print('\nUso avanzado:')
47 print('Problema resuelto en %f milisegundos' % solver.
    wall_time())
48 print('Problema resuelto en %d iteraciones' % solver.
    iterations())
49
50 """ó
51 Solucin:
52 Valor objetivo = 120000.0
53 Varible: A, valor:0.0
54 Varible: B, valor:1.0
55 Varible: C, valor:1.0
56
57 Uso avanzado:
58 Problema resuelto en 111.000000 milisegundos
59 Problema resuelto en 0 iteraciones
60 """
```

Listing 1.1: Proyectos de inversión

Fabricación de ropa

Una costurera fabrica y vende faldas y pantalones de mezclilla, para lo cual cada semana compra un rollo de 50 metros de mezclilla. Para hacer un pantalón requiere 2 metros de tela, mientras que para una falda, 1.5 metros.

Por lo general, ella trabaja ocho horas diarias, de lunes a viernes. Para hacer un pantalón requiere tres horas, mientras que hacer una falda le toma una. Un pantalón le genera 80 pesos de ganancia, mientras que al vender una falda gana 50 pesos.

Construir un modelo matemático que permita maximizar la ganancia semanal de la costurera, considerando que todo producto que fabrique puede venderlo.

```
#!/usr/bin/env python
# coding: utf-8

# ## óFabricacin de ropa
#
# Una costurera fabrica y vende faldas y pantalones de mezclilla, para lo cual cada semana compra un
# rollo de 50 metros de mezclilla. Para hacer un ópantaln requiere 2 metros de tela, mientras que para
# una falda, 1.5 metros.
#
# Por lo general, ella trabaja ocho horas diarias, de lunes a viernes. Para hacer un ópantaln requiere
# tres horas, mientras que hacer una falda le toma una. Un ópantaln le genera 80 pesos de ganancia,
# mientras que al vender una falda gana 50 pesos.
#
# Construir un modelo ámatemtico que permita maximizar la ganancia semanal de la costurera, considerando
# que todo producto que fabrique puede venderlo.

from ortools.linear_solver import pywraplp
solver = pywraplp.Solver.CreateSolver('SCIP')
infinity = solver.infinity()
n={}
tags = ["pantalon", "falda"]
for tag in tags:
    n[tag] = solver.IntVar(0, infinity, tag)

material = {'pantalon': 2, 'falda': 1.5}
tiempo = {'pantalon': 3, 'falda': 1}
ganancia = {'pantalon': 80, 'falda': 50}

solver.Add(
    material['pantalon']*n['pantalon']+ material['falda']*n['falda']<=50
)

solver.Add(
    tiempo['pantalon']*n['pantalon']+tiempo['falda']*n['falda']<=40
)

solver.Maximize(
    ganancia['pantalon']*n['pantalon']+ganancia['falda']*n['falda']
)

# Resolvemos el sistema.
status = solver.Solve()

if status == pywraplp.Solver.OPTIMAL:
    print('óSolucin:')
    print('Valor objetivo =', solver.Objective().Value())
    for k, v in n.items():
        print("Variable: {}, valor:{} ".format(k, v.solution_value()))
```

```
49
50 else:
51     print('EL problema no tiene ósolucin óptima.')
52
53 print('\nUso avanzado:')
54 print('Problema resuelto en %f milisegundos' % solver.
       wall_time())
55 print('Problema resuelto en %d iteraciones' % solver.
       iterations())
```

Listing 1.2: Proyectos de inversión

Problemas de mezcla de productos

Una compañía fabrica tres productos: crema corporal, crema facial y crema para bebés. Los tres productos comparten ingredientes en su elaboración: mezcla base, aceite de almendras, vitamina E y manteca de karité. En la siguiente tabla se presenta información acerca de los porcentajes de composición de cada uno de los tres productos:

.	Mezcla base	Aceite de Almendras	Vitamina E	Manteca de karité
Crema Corporal	90 %	4 %	1 %	5 %
Crema facial	85 %	8 %	2.5 %	4.5 %
Crema para bebé	80 %	10 %	-	10 %

Cada día, la compañía cuenta con 500 litros de la mezcla base, 50 litros de aceite de almendras, 5 litros de vitamina E y 30 litros de manteca de karité. Adicionalmente, se tiene la siguiente información sobre costos y precios de venta.

Ingrediente	Costo por litro
Mezcla base	$20
Aceite de almedras	$500
Vitamina E	$1500
Manteca de karité	$200

Producto	Precio de venta ($/L)
Crema corporal	$80
Crema facial	$120
Crema para bebé	$100

La demanda diaria de la crema corporal es de 200 litros; de la crema facial, 150 litros; y de la crema para bebé, de 250 litros. Por políticas de la empresa, se deben fabricar al menos 50 litros de crema

facial. ¿Cuánto de cada producto deberá producir la compañía para maximizar su utilidad?

```
#!/usr/bin/env python
# coding: utf-8

# # Problema Resuelto 3
# ## Problemas de mezcla de productos
#
# Una ñícompaa fabrica tres productos: crema corporal, crema facial y crema para ébebs. Los tres productos
# comparten ingredientes en su óelaboracin: mezcla base, aceite de almendras, vitamina E y manteca
# de ékarit. En la siguiente tabla se presenta óinformacin acerca de los porcentajes de ócomposicin de cada
# uno de los tres productos:
#
# | . | Mezcla base | Aceite de Almendras | Vitamina E | Manteca de ékarit |
#
|---|-------------|---------------------|------------|-------------------|

# |Crema Corporal| 90% | 4% | 1% | 5% |
# |Crema facial  | 85% | 8% | 2.5% | 4.5% |
# |Crema para ébeb | 80% | 10% | - | 10% |

# Cada ída, la ñícompaa cuenta con 500 litros de la mezcla base, 50 litros de aceite de almendras, 5 litros
# de vitamina E y 30 litros de manteca de ékarit. Adicionalmente, se tiene la siguiente óinformacin sobre
# costos y precios de venta.

# | Ingrediente | Costo por litro |
# |-------------|-----------------|
# |Mezcla base  | \$20 |
# |Aceite de almedras | \$500 |
# |Vitamina E | \$1500 |
# |Manteca de ékarit | \$200 |

# | Producto | Precio de venta (\$/L) |
# |----------|------------------------|
# |Crema corporal | \$80 |
# |Crema facial | \$120 |
# |Crema para ébeb | \$100 |

# La demanda diaria de la crema corporal es de 200 litros; de la crema facial, 150 litros; y de la crema para
# ébeb, de 250 litros. Por ípolticas de la empresa, se deben fabricar al menos 50 litros de crema facial.
# ¿áCunto de cada producto ádeber producir la ñícompaa para maximizar su utilidad?

"""Í
ndice: producto
0: crema corporal
1: crema facial
2: crema para ébeb
"""

from ortools.linear_solver import pywraplp
from sympy import Matrix
```

```
48
49 solver = pywraplp.Solver.CreateSolver("GLOP")
50
51 insumos = ["mezcla", "aceite", "vitamina", "manteca"]
52 productos = ["corporal", "facial", "bebe"]
53
54 """
55 Restricciones para productos°
56  óRestriccin para la demanda de crema corporal: 0<= x0 <=
       200.°
57  óRestriccin para la demanda de crema facial: 50<= x1 <= 150.
       °
58  óRestriccin para la demanda de crema para ébeb: 0 <= x2 <=
       250.
59 """
60
61 restricciones_productos = dict(
62     zip(productos, [[0,200], [50,150], [0,250]])
63 )
64
65 inf = solver.infinity()
66 x = dict()
67 for producto, restriccion in restricciones_productos.items():
68     a, b = restriccion
69     x[producto] = solver.NumVar(a,b,producto)
70
71 """
72 Beneficio = Ingreso - Costo
73 """
74 pct_values = [
75     [0.90, 0.04, 0.01, 0.05],
76     [0.85, 0.08, 0.025, 0.045],
77     [0.80, 0.10, 0.00, 0.10]
78 ]
79
80 pct = dict(
81     [(producto,
82         dict(
83             zip(insumos, renglon)
84             )
85       )
86      for producto, renglon in zip(productos, pct_values)]
87 )
88
89 print(pct)
90
91
92
93 """
94 Cm: costo marginal por litro
95 """
96 Cm = Matrix([
97     20, 500, 1500, 200
98 ])
99
100 # """
101 # costo productos = T(costo ingredientes)
102 # """
103 C = Matrix(pct_values)*Cm
104 print("C <-",C)
105
```

```
"""
I : ingresos
"""
I = Matrix([
    80, 120, 100
])
print("I <-",I)

"""
R: Ingresos
"""
R = dict(
    zip(productos, (I-C))
)
print("R <- ", R)

solver.Maximize(
    sum(R[tag]*x[tag] for tag in productos)
)


"""
Restricciones para insumos°
 óRestriccin para la mezcla base: 0.9x0 + 0.85x1 + 0.8x2 <=
    500.°
 óRestriccin para el aceite de almendras: 0.04x0 + 0.08x1 +
    0.1x2 <= 50.°
 óRestriccin para la vitamina E: 0.01x0 + 0.025x1 <= 5.°
 óRestriccin para la manteca de ékarit: 0.05x0 + 0.045x1 +
    0.1x2 <= 30.
"""
restricciones_insumos = dict(zip(insumos, [500, 50, 5, 30]))

for insumo in insumos:
    #insumo_total = 0
    #for producto in productos:
        #insumo_total = insumo_total + pct[producto][insumo]*
    x[producto]
    solver.Add(
        sum(
            pct[producto][insumo]*x[producto] for producto in
     productos
        )<=restricciones_insumos[insumo]
    )



# Resolvemos el sistema.
status = solver.Solve()

if status == pywraplp.Solver.OPTIMAL:
    print('óSolucin:')
    print('Valor objetivo =', solver.Objective().Value())
    for k, v in x.items():
        print("Variable: {}, valor:{} ".format(k, v.
    solution_value()))

else:
    print('EL problema no tiene ósolucin óptima.')

print('\nUso avanzado:')
```

```
161 print('Problema resuelto en %f milisegundos' % solver.
        wall_time())
162 print('Problema resuelto en %d iteraciones' % solver.
        iterations())
```

Listing 1.3: Proyectos de inversión

2 Programación lineal

2.1 Introducción a la programación lineal

```
from ortools.linear_solver import pywraplp

"""Linear programming sample."""
# Instantiate a Glop solver, naming it LinearExample.
solver = pywraplp.Solver.CreateSolver('GLOP')

# Create the two variables and let them take on any non-
    negative value.
x = solver.NumVar(0, solver.infinity(), 'x')
y = solver.NumVar(0, solver.infinity(), 'y')
z = solver.NumVar(0, solver.infinity(), 'z')

print('Number of variables =', solver.NumVariables())

# Constraint 0:
solver.Add(3*x+5*y+4*z<=540)

# Constraint 1:
solver.Add(6*x+y+3*z<=480)

print('Number of constraints =', solver.NumConstraints())

# Objective function: 3x + 4y.
solver.Maximize(5*x+3.5*y+4.5*z)

# Solve the system.
status = solver.Solve()

if status == pywraplp.Solver.OPTIMAL:
    print('Solution:')
    print('Objective value =', solver.Objective().Value())
    print('x =', x.solution_value())
    print('y =', y.solution_value())
    print('z =', z.solution_value())
else:
    print('The problem does not have an optimal solution.')

print('\nAdvanced usage:')
print('Problem solved in %f milliseconds' % solver.wall_time
    ())
print('Problem solved in %d iterations' % solver.iterations()
    )
```

3 Teoría de Gráficas

3.1 Matrices

Las matrices son arreglos rectangulares de número que nos ayudan a codificar información. Por ejemplo:

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

puede ser útil para codificar los coeficientes del sistema de ecuaciones:

$$\begin{cases} a_{1,1}x + a_{1,2}y = b_1 \\ a_{2,1}x + a_{2,2}y = b_2 \end{cases}$$

En general, una matriz tiene la forma

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & & & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \tag{A}$$

Los subíndices de cada elemento $a_{i,j}$ denotan la posición del mismo: i es el número del *renglón* (contando de arriba a abajo), mientras que j es el número de la columna (contanto de izquierda a derecha).

Podemos extraer renglones y columnas de la matrix (A): El $i-$esímo renglón es

$$R_i = \begin{pmatrix} a_{i,1} & \cdots & a_{i,n} \end{pmatrix}$$

mientras que la $j-$ésima columna será

$$C_j = \begin{pmatrix} a_{j,1} \\ \vdots \\ a_{j,m} \end{pmatrix}$$

Diremos que la matriz (A) tiene dimensión $m \times n$.

Si existe un conjunto de números F, tal que todos los elementos $a_{i,j}$ de la matriz pertenecen a dicho conjunto, Diremos que la matriz tiene coeficientes en F.

Observación. Para que las operaciones entre matrices estén bien definidas, es necesario que la suma, resta y multiplicación entre entre elementos de F también este bien definida. Por esto generalmente F se elige como $\mathbb{R}$ o $\mathbb{Z}$.

La colección de todas las matrices de dimensión $m \times n$ con coeficientes en F se denotará por

$$M_{m,n}(F).$$

Definición. Las matrices de dimensión $m \times 1$ se conocen como *vectores columna,* mientras que las de dimensión $1 \times n$ se conocen como *vectores renglón.*

La colección $M_{m,1}(F)$ de todos los vectores columna con coeficientes comunmente se denota por F^m. Mientras que la colección $M_{1,n}(F)$ de todos los vectores columna con coeficientes comunmente se denota por F^{n*}.

Operaciones elementales

Por brevedad, la matriz (A) se denota por $A = [a_{i,j}]$.

En el caso de los vectores renglones y columnas, podemos omitir el subíndice fijo

$$R = [R_{1,j}] = [R_j],\ C = [C_{i,1}] = [C_i].$$

Si $B = [b_{i,j}]$ es otra matriz de dimensión $m \times n$, la suma se define como

$$A + B = [a_{i.j} + b_{i,j}].$$

De manera similar, la resta se define como

$$A - B = [a_{i,j} - b_{i,j}].$$

Problema 3.1.

$$\begin{pmatrix} 1 & -1 & 0 \\ 2 & 3 & -4 \end{pmatrix} + \begin{pmatrix} 7 & 0 & -1 \\ 2 & -1 & 5 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & -1 & 0 \\ 2 & 3 & -4 \end{pmatrix} - \begin{pmatrix} 7 & 0 & -1 \\ 2 & -1 & 5 \end{pmatrix} =$$

Observe que para que la *suma y resta* tenga sentido, ambas matrices deben tener exactamente las *mismas dimensiones.*

Después de ver la facilidad para definir la suma y resta, uno se ve tentado a definir la multiplicación de la misma forma. Pero tal definición es poco útil en las aplicaciones.

Por esta razón, desarrollaremos el concepto de multiplicación, a fin de poder aplicar esta operación en la resolución de Ejemplos.

Multiplicación

Definición. Sean $R = [R_j]$ un vector renglón y $C = [C_i]$ un vector columna, ambos de longitud n.

El *producto renglón-columna* se define como

$$RC = \begin{pmatrix} R_1 & \cdots & R_n \end{pmatrix} \begin{pmatrix} C_1 \\ \vdots \\ C_n \end{pmatrix} = \sum_{i=1}^{n} R_j C_i. \qquad \text{(RC)}$$

Problema 3.2. Considere

$$R = \begin{pmatrix} 1 & 0 & -1 \end{pmatrix}, \; C = \begin{pmatrix} 2 \\ 1 \\ -2 \end{pmatrix}.$$

Calcule RC.

Problema 3.3. Reescriba la siguiente ecuación, utilizando el *producto renglón-columna*:

$$2x - 3y + z = 0.$$

Definición. Sea $A = [a_{i,j}] \in M_{m\times n}$ y $B = [b_{j,k}] \in M_{n\times l}$. Definimos su producto como

$$AB = \begin{pmatrix} R_i C_k \end{pmatrix} \qquad \text{(AB)}$$

donde R_i es el $i-$ésimo renglón de A y C_k es la $k-$ésima columna de B.

Observación.

- Para que esta multiplicación tenga sentido, los renglones de A y las columnas de B deberán tener la misma longitud n.
- La matriz resultante tendrá dimensión $m \times l$.
- A menos que $m = l$, el producto BA podría no estar definido.
- Aun cuando BA estuviera bien definido, el producto de matrices no es *conmutativo,* es decir, generalmente tendremos que

$$AB \neq BA.$$

Problema 3.4. Encuentre el producto AB de las siguientes matrices

$$A = \begin{pmatrix} 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & -1 \end{pmatrix}$$

Solución:

$$AB = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

Problema 3.5. Encuentre el producto AB de las siguientes matrices

$$A = \begin{pmatrix} 0 & -1 \\ -1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Solución:

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

Problema 3.6. Encuentre el producto AB de las siguientes matrices

$$A = \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix}$$

$$B = \begin{pmatrix} -1 & 0 & 0 \end{pmatrix}$$

Solución:

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

Problema 3.7. Encuentre el producto AB de las siguientes matrices

$$A = \begin{pmatrix} 6 \\ -9 \\ -10 \end{pmatrix}$$

$$B = \begin{pmatrix} -5 \end{pmatrix}$$

Solución:

$$AB = \begin{pmatrix} -30 \\ 45 \\ 50 \end{pmatrix}$$

Problema 3.8. Encuentre el producto AB de las siguientes matrices

$$A = \begin{pmatrix} 2 \end{pmatrix}$$

$$B = \begin{pmatrix} -1 & 1 & -3 \end{pmatrix}$$

Solución:

$$AB = \begin{pmatrix} -2 & 2 & -6 \end{pmatrix}$$

Problema 3.9. Encuentre el producto AB de las siguientes matrices

$$A = \begin{pmatrix} -1 & -3 \\ -7 & -1 \end{pmatrix}$$

$$B = \begin{pmatrix} -7 \\ -4 \end{pmatrix}$$

Solución:

$$AB = \begin{pmatrix} 19 \\ 53 \end{pmatrix}$$

Problema 3.10. Rescriba el siguiente sistema de ecuación en forma matricial y encuentre su solución:

$$\begin{cases} -x - 3y = 19 \\ -7x - y = 53 \end{cases}$$

3.2 *Teoría general de grafos*

En matemáticas, la *teoría de grafos* estudia estructuras matemáticas usadas para modelar relaciones por pares entre objetos.

Definición de grafo

Concepto de gráfo Un *grafo* G consiste de:

(a) Un conjunto V cuyos elementos son llamados *vértices,* puntos o nodos de G.

(b) Un conjunto E de pares (no ordenados) de distintos vertices, a los que llamaremos *aristas* de G.

Denotaremos un grafo por $G(V, E)$ cuando querramos enfatizar los componentes del mismo.

Observación. Debido a una ambigüedad en la traducción del inglés al español, en ocasiones, a un grafo también se le conoce como *gráfica,* que se puede confundir con el concepto de teoría de conjuntos. En este material, a veces utilizaremos *gráfica,* pero debe entenderse como un grafo.

Multigrafos Consideremos la figura 3.1 (b). Las aristas e_4y e_5 son llamadas *aristas multiples* ya que conectan los mismos extremos, mientras que la arista e_6 es llamada *bucle* ya que conecta un vértice consigo mismo.

Tales diagramas son llamados *multigrafos;* la definición formal de grafo no admite aristas multiples ni bucles.

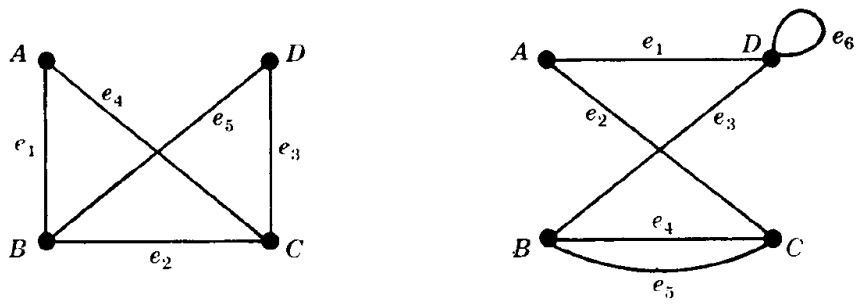


Figura 3.1: Grafos y multigrafos

Observación. Sin embargo, algunos textos utilizan "grafos" para referirse a lo que nosotros llamaremos multigrafos, mientras que ocupan "grafo simple" para lo que nosotros llamaremos grafos.

Grado de un vértice El *grado* de un vértice v es un grafo G, denotado por $\deg(v)$, es igual al número de aristas in G que contienen a v, es decir, que *inciden* en v.

Dado que cada arista incide en dos vértices diferentes, tenemos el siguiente resultado simple pero importante:

Teorema 3.1. *La suma de los grados de los vértices en un grafo G es el doble del número de aristas.*

Problema 3.11. En el grafo de la figura 3.1(a), tenemos que

$$\deg(A) = 2,\ \deg(B) = 3,\ \deg(C) = 3,\ \deg(D) = 2.$$

La suma de los grados es igual a 10, que es dos veces el número de aristas.

Definición. Diremos que un vértice es *par* o *impar* de acuerdo a la paridad de su grado.

En el ejemplo anterior, tanto A com D son vértices pares, mientras que B y C son impares.

Observación. Diremos que un vertice de grado cero está *aislado*.

Gráfos finitos y triviales Diremos que un grafo es *finito* si tiene un número finito de vértices y un número finito de aristas.

Observe que un número finito de vértices implica un número finito de aristas; pero no lo contrario no es necesariamente cierto.

Diremos que un grafo con un único vértice, sin aristas, es *trivial.*

Observación. A menos que se indique de otra manera, sólo trataremos con grafos finitos.

Subgrafos y grafos homeomorfos e isomorfos

Ahora, discutiremos relaciones de equivalencia entre grafos.

Subgrafos Consideremos un grafo $G(V, E)$. Diremos que otro grafo $H(V', E')$ es un *subgrafo* de G si los vértices y aristas de H están contenidos en los vértices y aristas de G, es decir,

$$V' \subset V,\ E' \subset E.$$

En particular:

(a) Un subgrafo $H(V', E')$ de $G(V, E)$ es llamado subgrafo *inducido* por sus vértices V' si el conjunto de aristas E' contiene todas las aristas en G cuyo extremos pertenecen a los vértices en H.

(b) Si v es un vértice en G, entonces $G - v$ es el subgrafo de G ontenido al borrar v de G y todas las aristas en G que inciden en v.

(c) Si e es una arista en G, entonces $G - e$ es el subgrafo de G obtenido borrando la arista e en G.

Grafos isomorfos Dos grafos $G(V, E)$ y $G^*(V^*, E^*)$ son llamados *isomorfos* si existe una función biyectiva $f : V \to V^*$ tal que: $\{u, v\}$ es una arista de G si y solo si $\{f(u), f(v)\}$ es una arista de G^*.

La idea es que estos grafos son equivalentes, aún cuando sus representaciones pueden lucir muy diferentes.

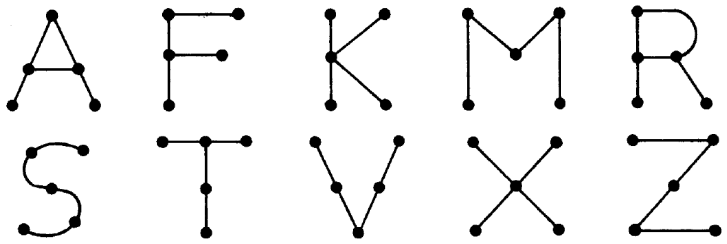


Figura 3.2: Grafos isomorfos.

Grafos homeomorfos Dado un grafo G, podemos obtener un nuevo grafo dividiendo una arista de G con vértices adicionales.

Dos grafos G y G^* son llamados *homeomorfos* si pueden obtenerse de gráficas isomorfas a través de este método.

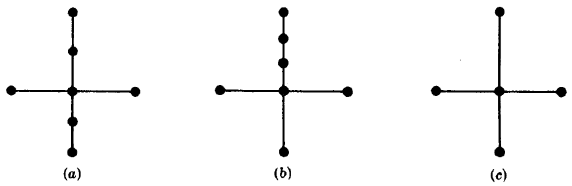


Figura 3.3: Grafos homomorfos

Los grafos (a) y (b) son homeomorfos, ya que se pueden obtener añadiendo vértices al grafo (c).

Caminos y conexidad

Un *camino* en un (multi)grafo G consiste en una sucesión alternante de vértices y arista de la forma

$$v_0, e_1, v_1, ..., e_{n-1}, v_{n-1}, e_n, v_n$$

donde cada arista e_i contiene los vértices v_{i-1} y v_i.

Observación. Observe que en grafo, podemos simplificar la notación para un camino, indicando sólo los vértices que recorre:

$$v_0, v_1, ..., v_n.$$

Diremos que el camino es *cerrado* si $v_n = v_0$. En otro caso, Diremos que el camino conecta v_0 con v_n.

Un *camino simple* es aquel en el cual todos los vértices son distintos. Mientras que un camino en el que todas las aristas son distintas se llama *paseo.*

La *longitud* de un camino es igual a número de aristas en la sucesión que lo define.

Un *ciclo* es un camino cerrado de *longitud* al menos 3, en el que todos los vértices son distintos, excepto el inicial v_0 y el final v_n.

Un ciclo de longitud k es llamado $k-$*ciclo.*

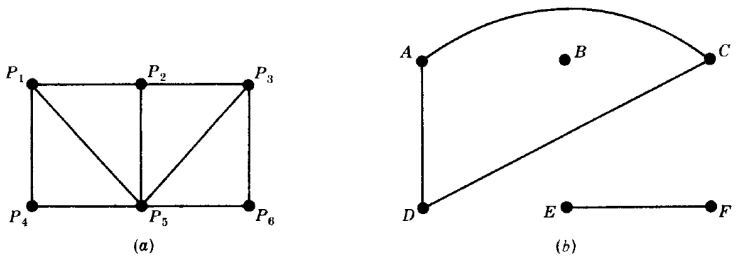


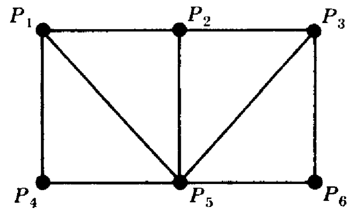
Figura 3.4: Conexidad en grafos

Problema 3.12. Consideremos el grafo 3.4(a). Considere las siguientes sucesiones

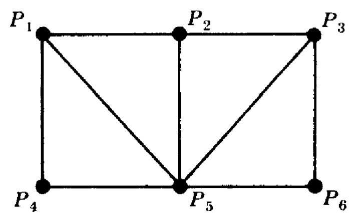
$$\begin{aligned}
\alpha &= (P_4, P_1, P_2, P_5, P_1, P_2, P_3, P_6)\,, \\
\beta &= (P_4, P_1, P_5, P_2, P_6) \\
\gamma &= (P_4, P_1, P_5, P_2, P_3, P_5, P_6) \\
\delta &= (P_4, P_1, P_5, P_3, P_6)\,.
\end{aligned}$$

α es un camino de P_4 a P_6, pero no es un paseo.

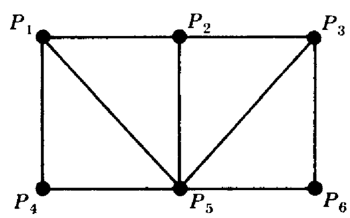
β no es un camino, ya que no existe alguna arista $\{P_2, P_6\}$.



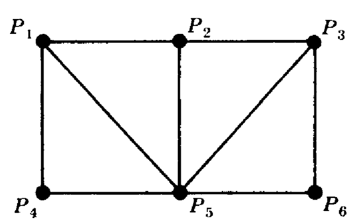
(a)



(a)



(a)



(a)

γ es un paseo, pero no es un camino simple.

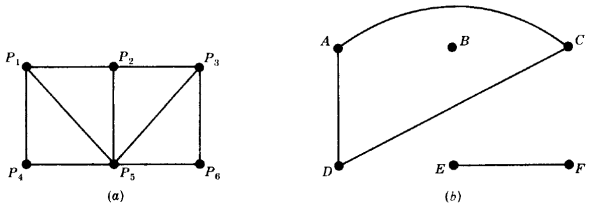
δ es un camino simple de P_4 a P_6, pero no es el camino más corto, es decir, con el meno número de aristas. ¿Cuál es el camino más corto?

Eliminando aristas innecesarias, no es difícil ver que cualquier camino de u a v puede ser reemplazado por un camino simple.

Formalmente:

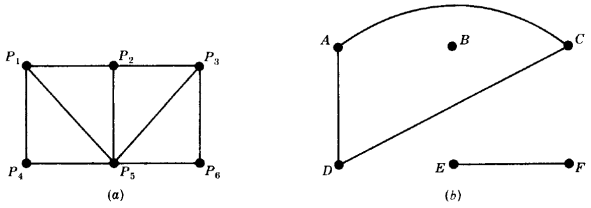
Teorema 3.2. *Existe un camino del vértice u a v si y solo si existe un camino simple de u a v.*

Conexidad y componentes conexas Un grafo G es conexo si existe un camino entre cualesquiera dos vértices. Por ejemplo, el grafo 3.4(a) es conexo, pero no así el grafo 3.4(b).



Consideremos un grafo G. Un subgrafo conexo H de G es llamado *componente conexa* de G si H no está contenido de manera propia en cualquier otro grafo conexo de G.

Por ejemplo, el grafo 3.4(b) tiene tres componentes conexas.



Observación. Formalmente, permitiendo que un vértice u esté conectado consigo mismo, la relación

$$u \texttt{ está conectado con } v$$

es una relación de equivalencia en el conjunto de vértices del grafo G, y las clases de equivalencia de esta relación son las componentes conexas de G.

Distancia y diametro Consideremos un grafo conexo G. La distancia entre dos vértices u y v en G, denotada por $d(u, v)$, es la longitud del camino más corto entre u y v. Eñ diametro de G, escrito $diam(G)$, es la distancia máxima entre cualesquiera dos puntos en G.

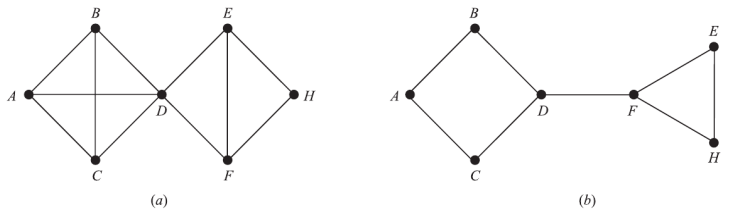


Figura 3.5: Distancia y diametro

Por ejemplo, en el grafo 3.5(a), el diamtero es 3, mientras que en el (b), el diametro es 4.

Puntos de corte y puentes Sea G un grafo conexo. Un vértice v en G es llamado *punto de corte* si $G - v$ es disconexo. Una arista e en G es llamada *puente* si $G - e$ es disconexo.

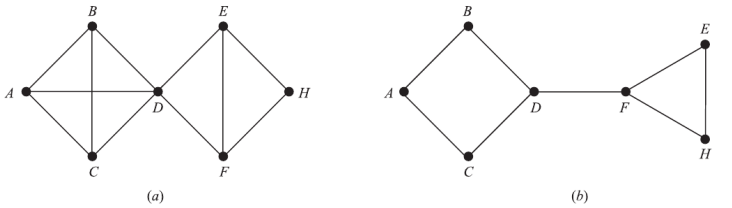


Figura 3.6: Puntos de corte y puentes

Grafos transitables y eulerianos

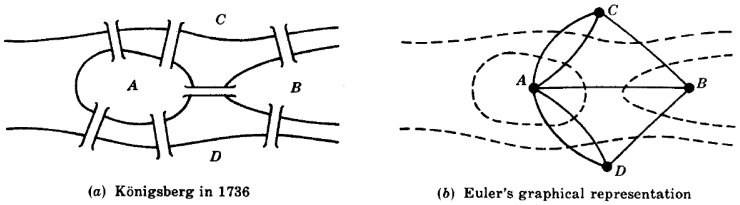


Figura 3.7: Puentes de Königsberg y su representación

Un multigrafo es llamado *transitable* si existe un *paseo* (un camino dónde todos las aristas son diferentes), que incluye *todos los vértices y todas las aristas.*

Tal paseo será llamado *paseo transitable.*

Observación. De manera equivalente, un paseo transitable es un camino en el que todos los vértices se transitan *al menos* una vez, pero las aristas *exactamente* una vez.

Proposición 3.1. *Cualquier grafo conexo y finito con exactamente dos vértices impares es transitable. Un paseo transitable puede comenzar en alguno de los vértices impares y terminar en el otro vértice impar.*

Un grafo G es llamado *grafo Euleriano* si existe un *paseo transitable cerrado.*

A tal paseo le llamaremos *paseo Euleriano.*

Teorema 3.3 (Euler)**.** *Un grafo conexo y finito es Euleriano si y solo si cada vértice tiene grado par.*

Grafos hamiltonianos En la definición de grafos Eulerianos se enfatizó pasar por todas las aristas.

Ahora, nos enfocaremos en visitar todos los vértices.

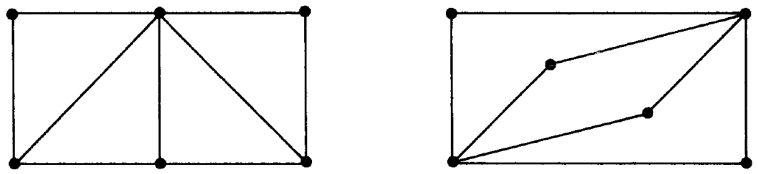
Un *circuito Hamiltoniano* es un grafo G es un camino cerrado que visita cada vértice en G *exactamente* una vez.

Si G admite un circuito Hamiltoniano, entonces G es llamado un *grafo Hamiltoniano.*

Observación. En la definición de circuito Hamiltoniano, cuando decimos que el camino *visita* cada vértice exactamente una vez significa que, aunque el vértice inicial tiene que ser el mismo que el final, todos los demás vértices intermedios deben ser distintos.

Observación. Un paseo Euleriano atraviesa cada una de las aristas exactamente una vez, pero los vértices se pueden repetir, mientras que un circuito Hamiltoniano visita cada uno de los vértices exactamente una vez, pero las aristas pueden repetirse.

Teorema 3.4. *Sea G un grafo conexo con n vértices. Entonces G es Hamiltoniano si $n \geq 3$ y $n \leq \deg(v)$ para cada vértice v en G.*



(a) Hamiltonian and non-Eulerian

(b) Eulerian and non-Hamiltonian

Figura 3.8: Circuitos Eulerianos y Hamiltonianos

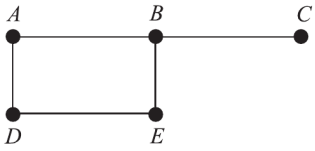
Matriz de adyacencia

Supongamos que G es un gráfo con m vértices y que estos han sido ordenados:

$$v_1, v_2, ..., v_m.$$

Entonces, la *matriz de adyacencia* $A = (a_{i,j})$ del grafo G es la matriz de dimensión $m \times m$ definida por:

$$a_{i,j} = \begin{cases} 1 & v_i \texttt{ es adyacente a } v_j \\ 0 & \texttt{en otro caso} \end{cases}$$



(a)

	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	0	1
C	0	1	0	0	0
D	1	0	0	0	1
E	0	1	0	1	0

(b)

Figura 3.9: Matriz de adyacencia

3.3 *Digrafos*

Los *grafos dirigidos* o *digrafos* son grafos en los que las aristas tienen una dirección.

Grafos dirigidos

Un grafo dirigido $G = G(V, E)$ consiste de:

1. Un conjunto $V = V(G)$ cuyos elementos son llamados *vértices*;
2. un conjunto $E = E(G)$ de *pares ordenados* ordenados de vértices llamados *arcos* o *aristas dirigidas.*

Supongamos que $e = (u, v)$ es un arco en el digrafo G. Entonces, la siguiente terminología es usada:

- e comienza en v y termina en v;
- u es el origen o punto inicial de e, mientras que v es el destino o punto final de e.
- v es un sucesor de u;
- u es adyacente a v y v es adyacente desde u.

Si $u = v$, e es llamado un *bucle.*

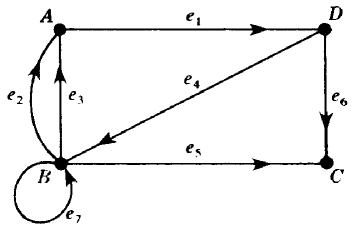
Si las aristas o los vértices de un digrafo están etiquetas con algún tipo de dato, Diremos que es un *digrafo etiquetado.*

De manera similar a un grafo, un digrafo será finito si el conjunto de vértices y el de aristas es finito.

Problema 3.13. Consideremos el siguiente digrafo. Las aristas e_2 y e_3 son llamados *paralelos,* ya que ambos comienzan en B y terminan en A. La arista e_7 es un *bucle.*

Matriz de adyacencia

Ahora, sólo consideraremos *digrafos simples* $G(V, E)$, es decir, sin aristas paralelas. Entonces E es simplemente una relación en V.



Problema 3.14.

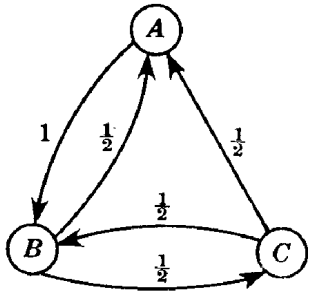


Figura 3.10: Proceso estocástico

De manera inversa, si R es una relación en V, entonces $G(V, R)$ es un digrafo simple.

En unidades anteriores, ya hemos construido digrafos asociados a relaciones de orden parcial, llamados diagramas de Hasse.

Supongamos que G es un digrafo simple con m vértices, y supongamos que los vértices de G han sido ordenados y son llamados $v_1, v_2, ..., v_m$.

Entonces la *matrix de adyacencia* $A = (a_{i,j})$ de G es la una matriz de dimensión $m \times m$ definida de la siguiente manera

$$a_{i,j} = \begin{cases} 1 & \exists e \in E : e = (v_i, v_j) \\ 0 & \texttt{en otro caso} \end{cases}$$

Observación. Las matrices de adyacencia de un mismo grafo dependen del orden en que se enumeren los vértices. Sin embargo, dos matrices de adyacencia de un mismo grafo están relacionadas por operaciones elementales: cambiar el orden de columnas y renglones.

Problema 3.15. Sea G el siguiente digrafo

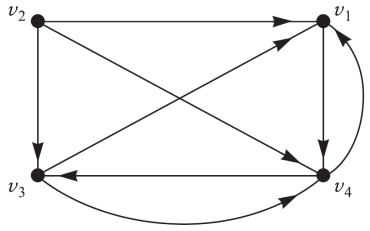


Figura 3.11: Construya su matriz de adyacencia del digrado anterior.

La matriz identidad $I_m = (I_{i,j})$ de dimensión $m \times m$ se define como

$$I_{i,j} \begin{cases} 1 & i = j \\ 0 & i \neq j, \end{cases}$$

es decir, es matriz cuadrangular con $1's$ en la *diagonal principal*, y ceros en cualquier otra entrada.

Problema 3.16.

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

La propiedad principal de una matriz identidad I_m es que es nuestra respecto a la multiplicación de matrices, es decir, para cualquier otra matriz $A \in M_n$:

$$AI_n = I_n A = A.$$

La potencia $n-$ésima de una matriz $A \in M_n$ se define de manera recursiva como

$$A^n = \begin{cases} I_n & n = 0 \\ AA^{n-1} & n \in \mathbb{N}. \end{cases}$$

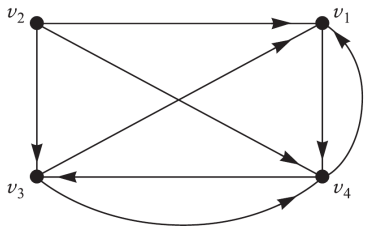
Es decir,

$$A^0 = I, A^1 = A, A^2 = AA, ...$$

Definamos $a_k(i, j)$ como la entrada en la posición i, j de A^k.

Proposición 3.2. *Sea A la matriz de adyacencia de un grafo G. Entonces $a_k(i, j)$ es igual al número de caminos de longitud k que van de v_i a v_j.*

Ejemplo Consideremos nuevamente el grafo



Recordemos que su matriz de adyacencia es

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad \text{(AD)}$$

Entonces

$$A^2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 2 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 2 \end{pmatrix} \quad A^3 = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 3 & 0 & 2 & 3 \\ 2 & 0 & 1 & 2 \\ 2 & 0 & 2 & 1 \end{pmatrix}$$

$$A^4 = \begin{pmatrix} 2 & 0 & 2 & 1 \\ 5 & 0 & 3 & 5 \\ 3 & 0 & 2 & 3 \\ 3 & 0 & 1 & 4 \end{pmatrix}$$

Observe que $a_2(4,1) = 1$, de manera que existe un solo camino de longitud 2 de v_4 a v_1. De manera similar, como $a_3(2,3) = 2$, entonces existen dos caminos de longitud 3 de v_2 a v_3.

Observación. Si definimos

$$B_r = \sum_{i=1}^{r} A^i,$$

entonces la entrada i, j de esta matriz nos indicará el número de caminos de longitud a lo más r de v_i a v_j.

En nuestro ejemplo, considerando A dado por (AD), tenemos que

$$B_4 = \begin{pmatrix} 4 & 0 & 3 & 4 \\ 11 & 0 & 7 & 11 \\ 7 & 0 & 4 & 7 \\ 7 & 0 & 4 & 7 \end{pmatrix} \tag{3.1}$$

¿Existe alguna manera de llegar al vertice v_2 desde el vértice v_1, sin importar la longitud del camino?

Matriz de accesibilidad

Sea $G = G(V, E)$ un grafo simple dirigido con m vértices $v_1, ..., v_m$. La *matriz de accesibilidad* de G es la matriz $m-$cuadrangular $P = (p_{ij})$ definida de la siguiente manera:

$$p_{ij} = \begin{cases} 1 & \texttt{existe un camino de } v_i \texttt{ a } v_j \\ 0 & \texttt{en otro caso} \end{cases}$$

Proposición 3.3. *Sea A la matriz de adyacencia de un grafo G con m vértices. Entonces la matriz de accesibilidad y*

$$B_m = \sum_{i=1}^{m} A^i \tag{3.2}$$

tienen exactamente las mismas entradas no nulas.

Definición. Un digrafo es *fuertemente conexo* si para cualquier par de vértices u, v existe al menos un camino de u a v y otro de v a u.

Proposición 3.4. *Sea $A \in M_m$ la matriz de adyacencia de un grafo G. Entonces, las siguientes proposiciones son equivalentes:*

1. *G es fuertemente conexo;*
2. *la matriz de accesibilidad P no tiene entradas nulas;*
3. *la matriz B_m, dada por (3.2), no tiene entradas nulas.*

Problema 3.17. Para encontrar la matriz de accesibilidad asociada a la matriz de adyacencia A, dada por (AD), basta sustitur las entradas no nulas en la matriz B_4, dada por (3.1), por $1's$:

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