

Tarea 1, 2 Unidad3. Grafos

Estructuras de Datos y Algoritmos II.

Prof. Gerardo Tovar Tapia.
Alumna. Monroy Velázquez Alejandra Sarahí
Grupo 6

Sección 1. Programar.

- 1) Del libro “python para informáticos”. Leer el capítulo 8, hacer los ejercicios que se proponen en de dichas sección.

Ejercicio 8.1 Escribe una función llamada *recorta*, que tome una lista, la modifique, eliminando los elementos primero y ultimo, y devuelva None.

Después escribe una función llamada *centro*, que tome una lista y devuelva otra que contenga todos los elementos de la original, menos el primero y el último.

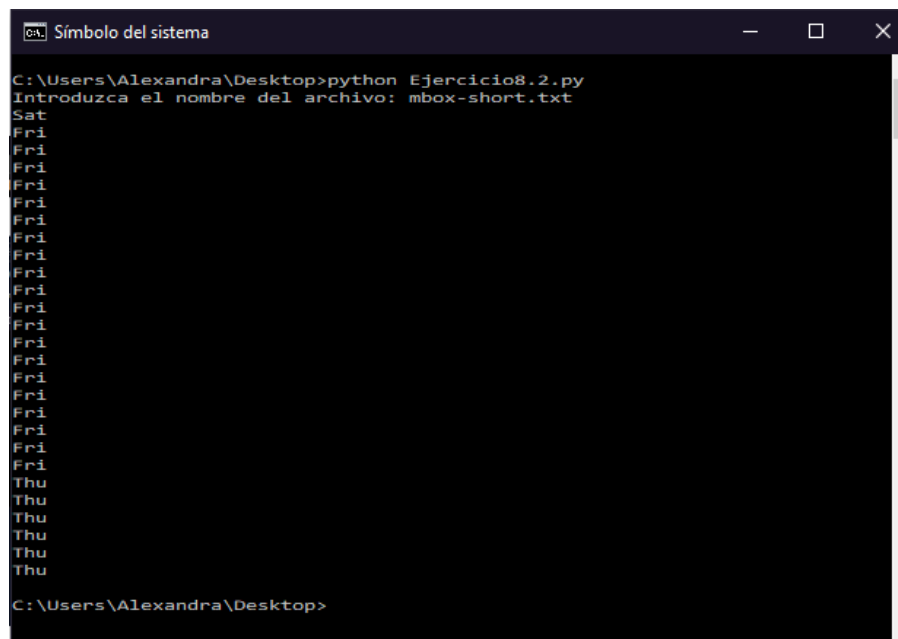
```
1 def recorta(A):
2     A.pop(0)
3     A.pop(len(A)-1)
4     return None
5
6 def centro(A):
7     del A[1:len(A)-1]
8     return A
9
10 lista = [1,2,3,4,5,6,7,8,9]
11 print("--Ejercicio 8.1--\n")
12 print("Lista original: ",lista)
13 recorta(lista)
14 print("Borrando primer y último elemento de la lista: ", lista)
15 otlista = centro(lista)
16 print("Borrando centro de la lista ya modificada: ",otlista)
17
```

--Ejercicio 8.1--

```
Lista original: [1, 2, 3, 4, 5, 6, 7, 8, 9]
Borrando primer y último elemento de la lista: [2, 3, 4, 5, 6, 7, 8]
Borrando centro de la lista ya modificada: [2, 8]
```

Ejercicio 8.2 Averigua qué línea del programa anterior aún no está suficientemente protegida. Intenta construir un archivo de texto que provoque que el programa falle, luego modifica el programa para que esa línea quede protegida adecuadamente, y pruébalo para asegurarte de que es capaz de manejar tu nuevo archivo de texto.

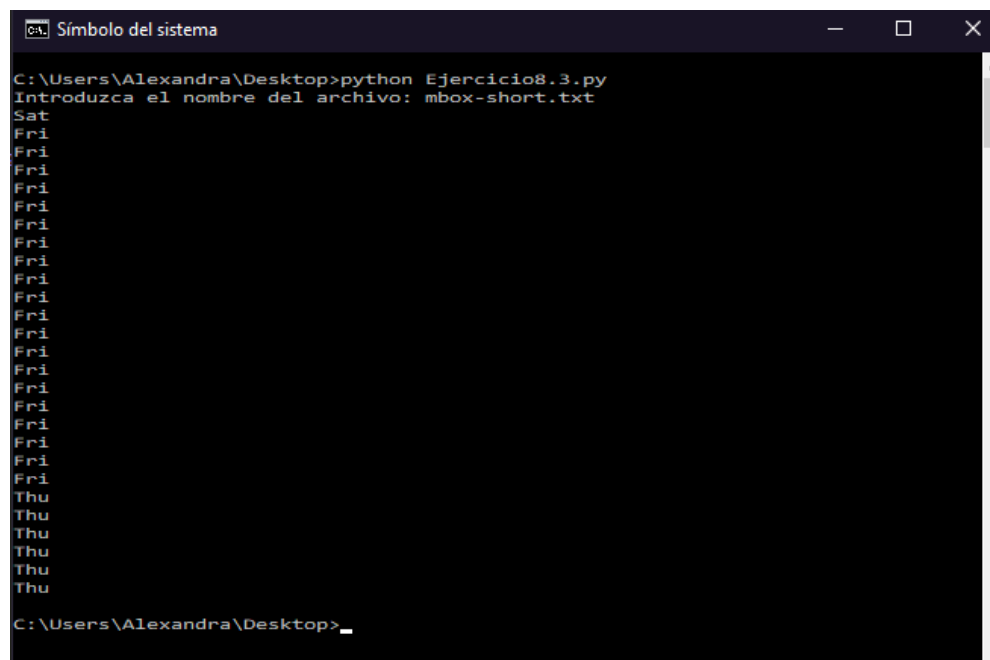
```
1 name=input("Introduzca el nombre del archivo: ")
2 try:
3     manf=open(name)
4 except:
5     print("No se pudo abrir el fichero: ",name)
6     exit()
7
8 for linea in manf:
9     palabras = linea.split()
10    if len(palabras) == 0: continue
11    if palabras[0] != 'From' :continue
12    print (palabras[2])
13
```



```
Símbolo del sistema
C:\Users\Alexandra\Desktop>python Ejercicio8.2.py
Introduzca el nombre del archivo: mbox-short.txt
Sat
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Thu
Thu
Thu
Thu
Thu
C:\Users\Alexandra\Desktop>
```

Ejercicio 8.3 Reescribe el código guardián en el ejemplo de arriba para que no use dos sentencias `if`. En su lugar, usa una expresión lógica compuesta, utilizando el operador lógico `and` en una única sentencia `if`.

```
1 name=input("Introduzca el nombre del archivo: ")
2 try:
3     manf=open(name)
4 except:
5     print("No se pudo abrir el fichero: ",name)
6     exit()
7
8 for linea in manf:
9     palabras = linea.split()
10    if len(palabras) != 0 and palabras[0] == 'From':
11        print (palabras[2])
12
```



```
Símbolo del sistema
C:\Users\Alexandra\Desktop>python Ejercicio8.3.py
Introduzca el nombre del archivo: mbox-short.txt
Sat
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Fri
Thu
Thu
Thu
Thu
Thu
C:\Users\Alexandra\Desktop>
```

Ejercicio 8.4 Descarga una copia del fichero, desde www.py4inf.com/code/romeo.txt

Escribe un programa que abra el archivo `romeo.txt` y lo lea línea a línea. Para cada línea, divídela en una lista de palabras usando la función `split`.

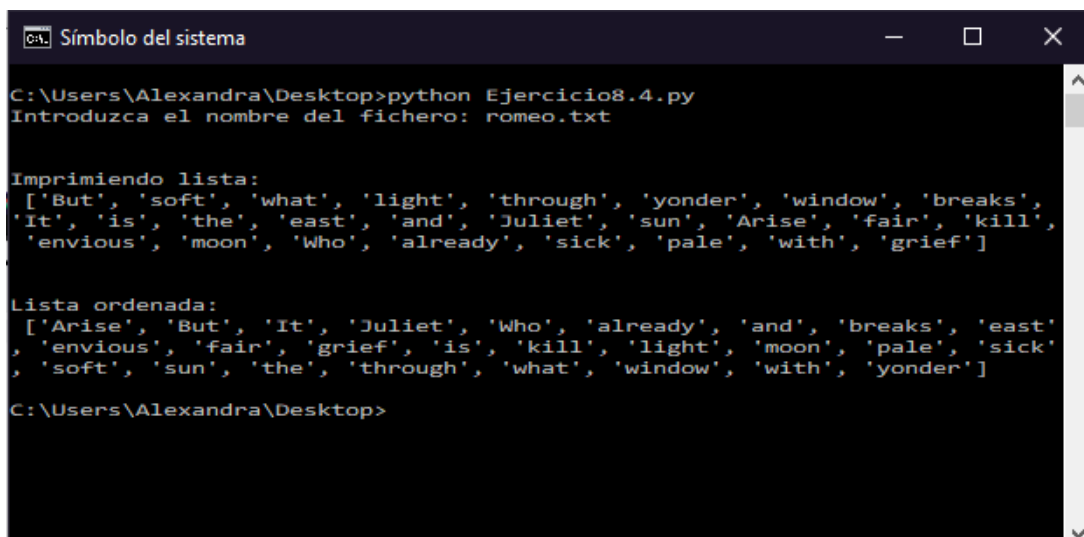
Para cada palabra, mira a ver si esa palabra ya existe en la lista. Si no es así, añádela.

Cuando el programa finalice, ordena y muestra en pantalla las palabras resultantes, en orden alfabético.

Introduzca fichero: `romeo.txt`

`['Arise', 'But', 'It', 'Juliet', 'Who', 'already', 'and', 'breaks', 'east', 'envious', 'fair', 'grief', 'is', 'kill', 'light', 'moon', 'pale', 'sick', 'soft', 'sun', 'the', 'through', 'what', 'window', 'with', 'yonder']`

```
1 name = input("Introduzca el nombre del fichero: ")
2 try:
3     manf=open(name)
4 except:
5     print("No se pudo abrir el fichero: ",name)
6     exit()
7
8 lista = []
9 for linea in manf:
10     palabras = linea.split()
11     for i in palabras:
12         if i not in lista:
13             lista.append(i)
14
15 print("\n\nImprimiendo lista: \n",lista)
16 lista.sort()
17 print("\n\nLista ordenada: \n",lista)
18
```



```
C:\Users\Alexandra\Desktop>python Ejercicio8.4.py
Introduzca el nombre del fichero: romeo.txt

Imprimiendo lista:
['But', 'soft', 'what', 'light', 'through', 'yonder', 'window', 'breaks',
'It', 'is', 'the', 'east', 'and', 'Juliet', 'sun', 'Arise', 'fair', 'kill',
'envious', 'moon', 'Who', 'already', 'sick', 'pale', 'with', 'grief']

Lista ordenada:
['Arise', 'But', 'It', 'Juliet', 'Who', 'already', 'and', 'breaks', 'east',
'envious', 'fair', 'grief', 'is', 'kill', 'light', 'moon', 'pale', 'sick',
'soft', 'sun', 'the', 'through', 'what', 'window', 'with', 'yonder']

C:\Users\Alexandra\Desktop>
```

Ejercicio 8.5 Escribe un programa que lea a través de los datos de un buzón de correo, y cuando encuentre una línea que empiece por "From", la divida en palabras usando la función `split`. Estamos interesados en quien nos envían el mensaje, que es la segunda palabra de la línea From.

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

Debes analizar la línea From y mostrar en pantalla la segunda palabra de cada una de esas líneas, luego ir contabilizando también el número de líneas From (no From:), y mostrar el total al final.

Este es un buen ejemplo de salida con algunas líneas eliminadas:

`python fromcount.py`

Introduzca un nombre de fichero: `mbox-short.txt`

`stephen.marquard@uct.ac.za`

`louis@media.berkeley.edu`

`zqian@umich.edu`

[...parte de la salida eliminada...]

`ray@media.berkeley.edu`

`cwen@iupui.edu`

`cwen@iupui.edu`

`cwen@iupui.edu`

Hay 27 líneas en el archivo con From como primera palabra.

```
1 name=input("Introduzca el nombre del archivo: ")
2 try:
3     manf=open(name)
4 except:
5     print("No se pudo abrir el fichero: ",name)
6     exit()
7
8 cont = 0
9 for linea in manf:
10     if linea.startswith('From: '):
11         linea = linea.split()
12         print(linea[1])
13         cont+=1
14
15 print("Hay {} líneas con la palabra From al inicio.".format(cont))
16
```

```
Símbolo del sistema

C:\Users\Alexandra\Desktop>python Ejercicio8.5.py
Introduzca el nombre del archivo: mbox-short.txt
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
zqian@umich.edu
rjlowe@iupui.edu
zqian@umich.edu
rjlowe@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
gsilver@umich.edu
gsilver@umich.edu
zqian@umich.edu
gsilver@umich.edu
wagnermr@iupui.edu
zqian@umich.edu
antranig@caret.cam.ac.uk
gopal.ramasammycook@gmail.com
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
louis@media.berkeley.edu
ray@media.berkeley.edu
cwen@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
Hay 27 líneas con la palabra From al inicio.

C:\Users\Alexandra\Desktop>
```

Ejercicio 8.6 Reescribe el programa que pide al usuario una lista de números e imprime en pantalla el máximo y mínimo de los números introducidos al final, cuando el usuario introduce "fin". Escribe ahora el programa de modo que almacene los números que el usuario introduzca en una lista y usa las funciones `max()` y `min()` para calcular los números máximo y mínimo después de que el bucle termine.

Introduzca un número: 6
Introduzca un número: 2
Introduzca un número: 9
Introduzca un número: 3
Introduzca un número: 5
Introduzca un número: fin
Máximo: 9.0
Mínimo: 2.0

```
1 try:
2     nums= []
3     num = int(input("Introduzca un numero: "))
4     while True:
5         num = int(input("Introduzca un numero: "))
6         nums.append(num)
7     except:
8         print("El Maximo es: ",max(nums))
9         print("El Minimo es: ",min(nums))
10
11
```

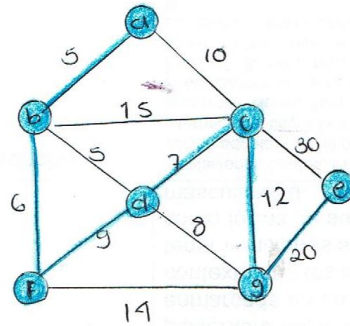
```
[GCC 4.8.2] on linux
>
Introduzca un numero: 6
Introduzca un numero: 2
Introduzca un numero: 9
Introduzca un numero: 3
Introduzca un numero: 5
Introduzca un numero: fin
El Maximo es: 9
El Minimo es: 2
>
```

Sección 2. Investigar, analizar, resolver.

Nota: Las actividades fueron hechas a mano y escaneadas.

Ejercicio 2.

En la siguiente gráfica los vertices representan ciudades y los números sobre las aristas son los costos de construir carreteras indicadas. Encuentre el sistema de carreteras menos costoso que conecte todas las ciudades.

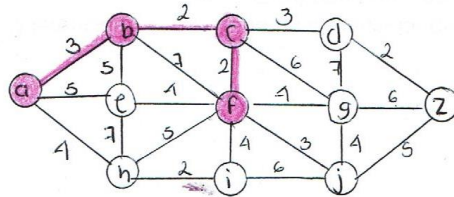


$a \rightarrow b \rightarrow f \rightarrow d \rightarrow c \rightarrow g \rightarrow e$

$$5 + 6 + 9 + 7 + 12 + 20 = 59 //$$

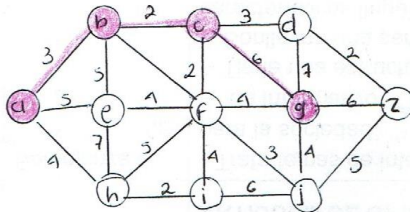
Ejercicio 3

En los ejercicios 1 al 5, encuentre la longitud de una ruta más corta entre cada par de vértices en la gráfica ponderada.

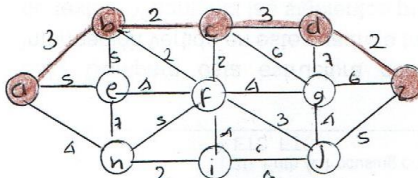


- A (a, a)
- B (3, a) (3, a)
- C (5, b) (5, b)
- D (5, a) (8, b)
- E (5, a) (8, b)
- F (10, b) (7, c)
- G (a, a)
- H (a, a)

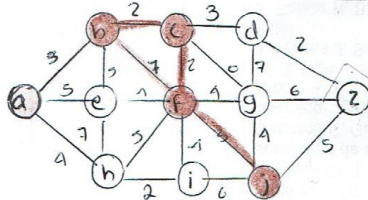
1. a, f $a \rightarrow b \rightarrow c \rightarrow f$
 $3 + 2 + 2 = 7$



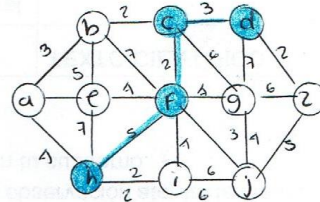
2. a, g $a \rightarrow b \rightarrow c \rightarrow g$
 $3 + 2 + 6 = 11$



3. a, z $a \rightarrow b \rightarrow c \rightarrow d \rightarrow z$
 $3 + 2 + 3 + 2 = 10$



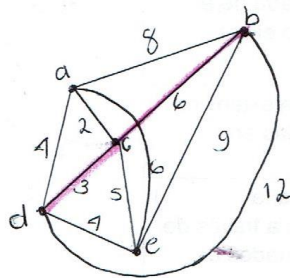
4. b, j $b \rightarrow c \rightarrow f \rightarrow j$
 $2 + 2 + 3 = 7$



5. h, d $h \rightarrow f \rightarrow c \rightarrow d$
 $5 + 2 + 3 = 10$

Ejercicio A.

Encuentra el camino más corto (d, b)



$$d \rightarrow a = 4$$

$$d \rightarrow e = 4$$

$$d \rightarrow b = 12$$

$$d \rightarrow c = 3$$

$$\Rightarrow c \rightarrow a = 2 + 3 = 5$$

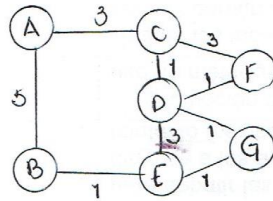
$$c \rightarrow e = 5 + 3 = 8$$

$$c \rightarrow b = 6 + 3 = 9$$

La ruta es de "d" a "c" y luego de "c" a "b" = 9

Ejercicio 5

Para la red de figura determine los caminos más cortos con origen en el nodo A hacia todos los demás aplicando el algoritmo de Dijkstra.

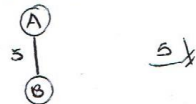


- Algoritmo -

Vertice

A	(0, A)
B	(5, A)
C	(3, A)
D	(3, A) (4, C)
E	(6, B) (7, D)
F	(6, C) (5, D)
G	(5, D)

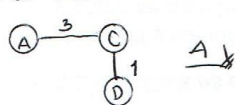
• Del nodo $A \rightarrow B$



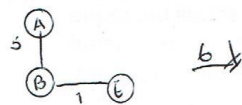
• Del nodo $A \rightarrow C$



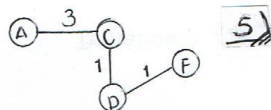
• Del nodo $A \rightarrow D$



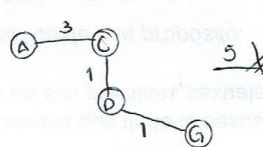
• Del nodo $A \rightarrow E$



• Del nodo $A \rightarrow F$



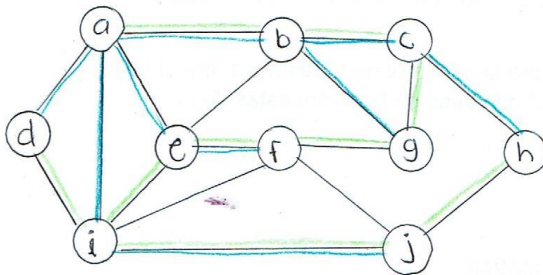
• Del nodo $A \rightarrow G$



Nota: Los recorridos en anchura corresponden al color azul y en profundidad al color verde, dibujados en los grafos.

Ejercicio 6

a)



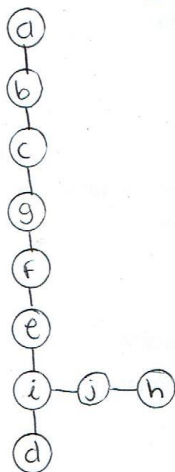
- Lista de adyacencia

a	b	c	d	e	f	g	h	i	j
b	a	b	a	a	e	b	c	a	f
e	c	g	i	b	g	c	g	d	h
d	e	h		f	i	f	h	e	i
i	g			i	j	h		f	
									j

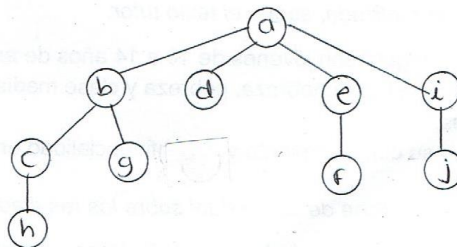
- Matriz

a	b	c	d	e	f	g	h	i	j
0	1	0	1	1	0	0	0	1	0
1	0	1	0	1	0	1	0	0	0
0	1	0	0	0	0	1	1	0	0
1	0	0	0	0	0	0	0	1	0
1	1	0	0	0	1	0	0	1	0
0	0	0	0	1	0	1	0	1	1
0	1	1	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0	0	1
1	0	0	1	1	1	0	0	0	1
0	0	0	0	0	1	0	1	1	0

- Recorrido en profundidad:



- Recorrido en anchura:



```

graph TD
    a((a)) --- b((b))
    a --- h((h))
    a --- g((g))
    b --- c((c))
    b --- i((i))
    b --- j((j))
    b --- k((k))
    c --- d((d))
    i --- o((o))
    j --- m((m))
    k --- l((l))
    g --- f((f))
    g --- n((n))
    f --- e((e))
    n --- p((p))
  
```

- Lista de adyacencia.

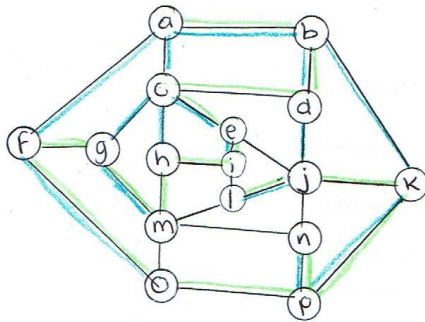


a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
b	a	b	c	d	e	a	a	b	b	b	d	e	f	i	m
g	c	d	e	f	g	f	g	h	i	c	k	j	g	j	n
h	i	k	l	m	n	h	i	j	k	j	m	l	h	p	o
	j					n	n	n	m	l		n	i		
								o	o			p	m		p

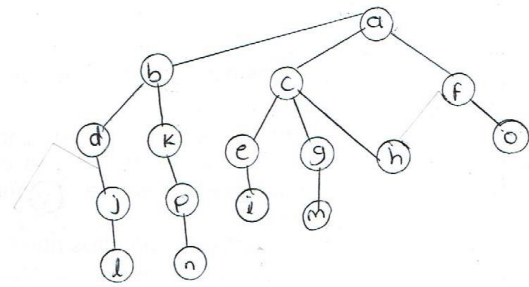
- Matrix

[illegible]

c)



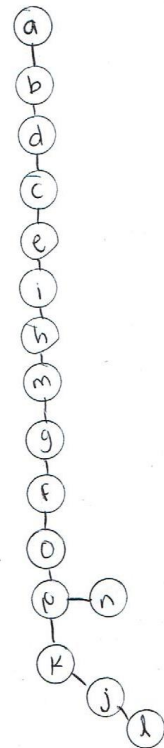
- Recorrido en anchura:



- Lista de adyacencia.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
b	a	a	b	c	a	c	c	e	d	b	i	g	j	f	k
c	d	d	c	i	o	f	i	h	e	j	j	h	m	m	n
f	k	e	j	j	g	m	m	l	k	p	m	l	p	p	o
		g							h			o			

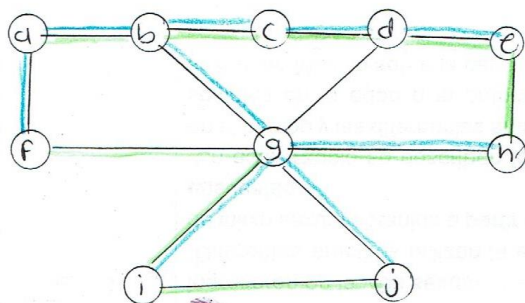
- Recorrido en Profundidad:



- Matriz

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
a	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
b	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
c	1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0
d	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0
e	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0
f	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
g	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0
h	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
i	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0
j	0	0	0	1	1	0	0	0	0	1	1	0	1	0	0	0
k	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1
l	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
m	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	0
n	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1
o	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1
p	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

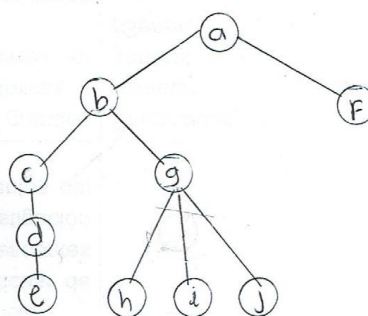
d)



- Lista de adyacencia

a	b	c	d	e	f	g	h	i	j
b	a	b	c	d	a	b	c	g	g
f	c	d	e	h	g	d	g	j	i
	g		g			f	h	i	j
						h	i	j	

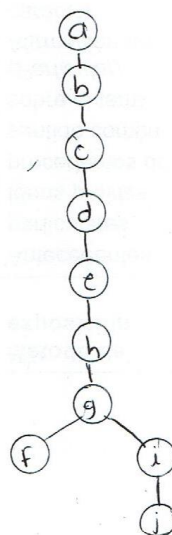
- Recorrido en anchura:

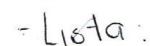


- Matriz

	a	b	c	d	e	f	g	h	i	j
a	0	1	0	0	0	1	0	0	0	0
b	1	0	1	0	0	0	1	0	0	0
c	0	1	0	1	0	0	0	0	0	0
d	0	0	1	0	1	0	1	0	0	0
e	0	0	0	1	0	0	0	1	0	0
f	1	0	0	0	0	0	1	0	0	0
g	0	1	0	1	0	1	0	1	1	1
h	0	0	0	0	1	0	1	0	0	0
i	0	0	0	0	0	0	1	0	0	1
j	0	0	0	0	0	0	1	0	1	0

- Recorrido en profundidad

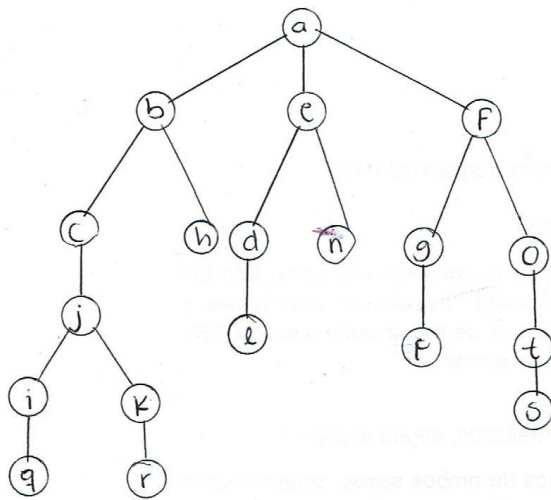




- Matrix

[illegible]

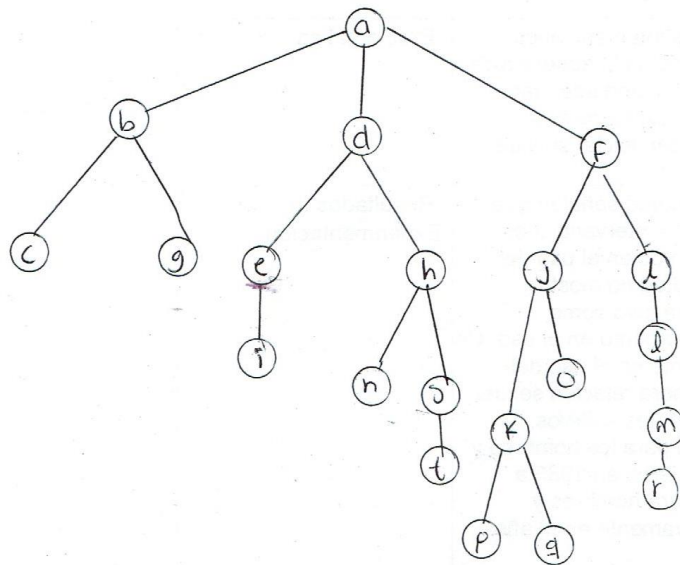
- Recorrido en anchura:



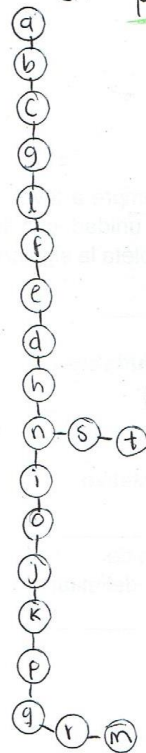
- Recorrido en profundidad



- Recorrido en anchura



- Recorrido en profundidad.

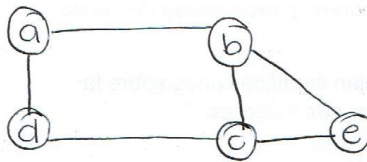


Ejercicio 7.

Hacer el recorrido de profundidad y anchura del siguiente grafo, además dibujar árboles resultantes.

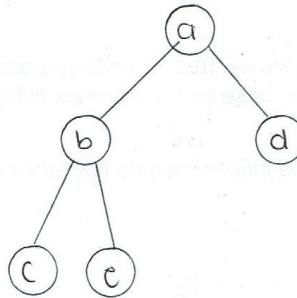
$$A = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{matrix}$$

Grafo :

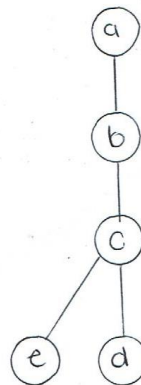


a	b	c	d	e
b	a	b	a	b
d	c	e	c	c
	e	d		

Recorrido en Anchura:



Profundidad :



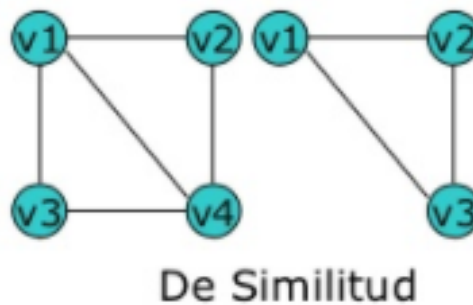
8) Explica que es un grafo.

Un grafo en el ámbito de las ciencias de la computación es un tipo abstracto de datos (TAD), que consiste en un conjunto de nodos (también llamados vértices) y un conjunto de arcos (aristas) que establecen relaciones entre los nodos. El concepto de grafo TAD descende directamente del concepto matemático de grafo.

Formalmente, Un grafo G es un par ordenado $G = (V, A)$, donde V es un conjunto de vértices o nodos y A es un conjunto de arcos o aristas, que relacionan estos nodos.

9) Qué es una gráfica de similitud.

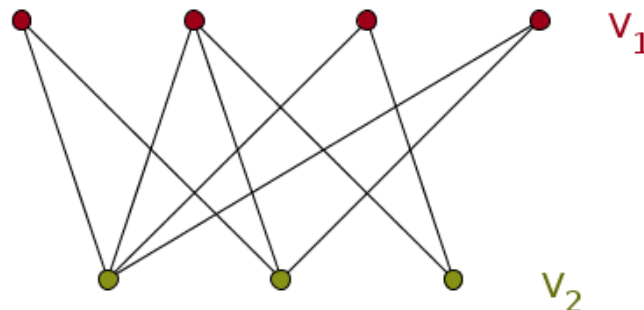
En la teoría de grafos, también llamada teoría de gráficas dice que una gráfica de similitud es un grafo de similitud, son aquellos donde se derivan sub-grafos.



10) Investiga y explicar qué es una gráfica bipartita

Una gráfica GG es bipartita si existe una partición de los vértices de GG en dos conjuntos V_1 y V_2 (no vacíos) de forma que cada arista de GG tenga un extremo en V_1 y el otro en V_2 .

Por ejemplo, la gráfica de la siguiente figura es bipartita:



Los vértices rojos forman el conjunto V_1 y los vértices verdes el conjunto de vértices V_2 .

11) Investigar qué es un grafo conexo

En teoría de grafos, un grafo se dice **conexo** si, para cualquier par de vértices u y v en G , existe al menos una trayectoria (una sucesión de vértices adyacentes que no repita vértices) de u a v .

