Ejercicios con Grafos

EEDD - GRADO EN ING. INFORMATICA - UCO

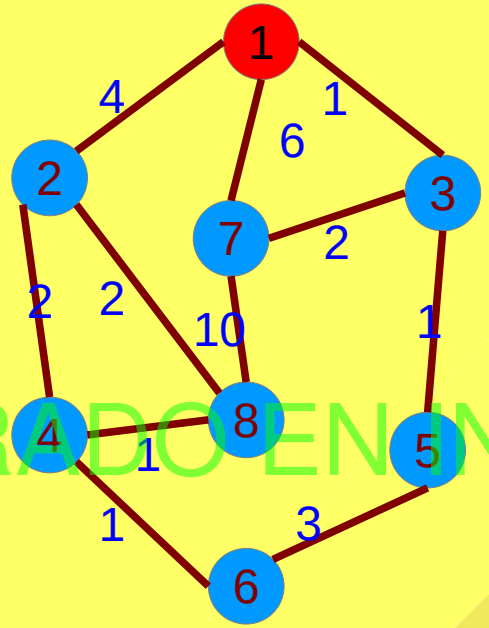
Recorridos y AAM

ChangeLog

16/5/2025:
- Corregida errata en ejercicio recorrdo en anchura al no expandir vértice 4 llegando desde 2 en la iteración 5.

EEDD - GRADO EN ING. INFORMATICA - UCO

Recorrido en profundidad



¿El lado (7,8) es del árbol?

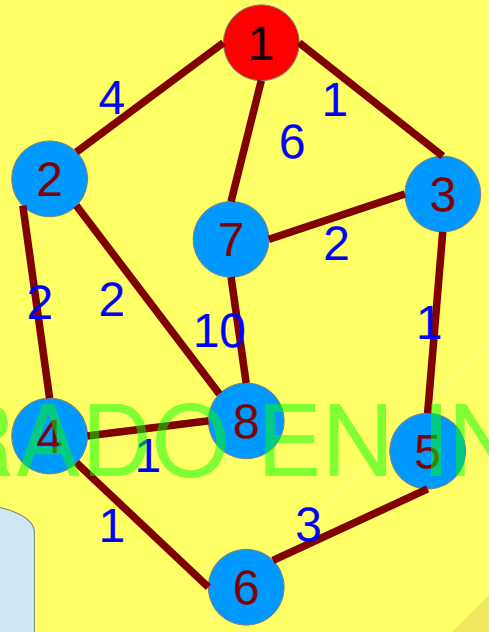
EEDD - GRADO ENING. INFORMATICA - UCO

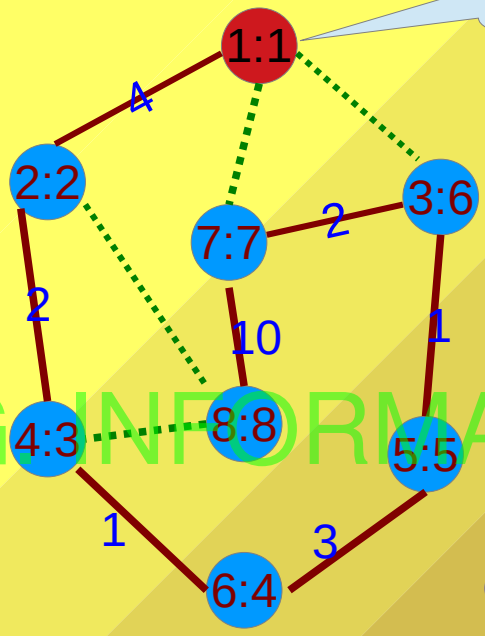
Coste del árbol: ¿?

Pila

En caso de varias alternativas coger la etiqueta menor

Recorrido en profundidad





"1" se procesa el primero

del árbol

av./ret.

EEDD - GRADO EN INGENIERIA INFORMATICA - UCO

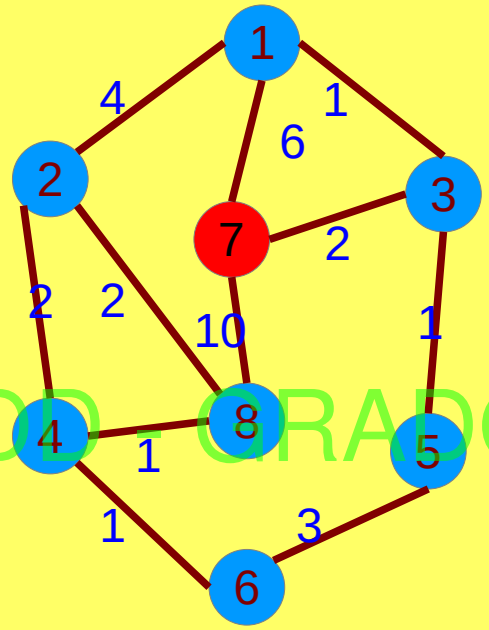
lado "2" llegando desde "1"

Coste del árbol: 4+2+1+3+1+2+10=23

1	2	3	4	5	6	7	8	-	-	-	-
1:1	**2:1** 3:1 7:1	**4:2** 8:2 3:1 7:1	**6:4** 8:4 8:2 3:1 7:1	**5:6** 8:4 8:2 3:1 7:1	**3:5** 8:4 8:2 3:1 7:1	**7:3** 8:4 8:2 3:1 7:1	**8:7** 8:4 8:2 3:1 7:1	8:4 8:2 3:1 7:1	8:2 3:1 7:1	3:1 7:1	7:1

En caso de varias alternativas coger la etiqueta menor

Recorrido en anchura



cola

7

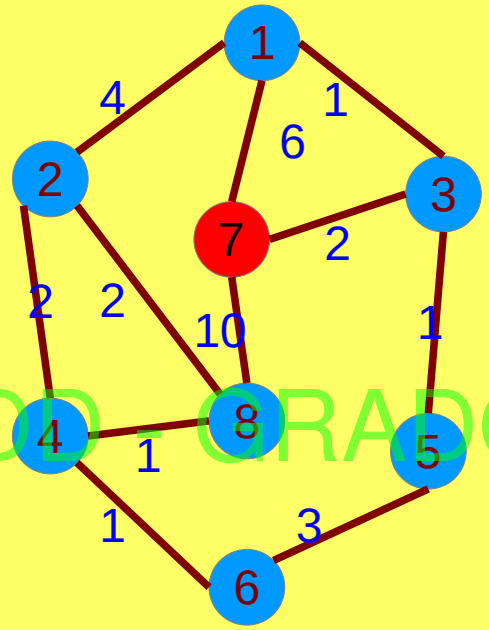
¿El lado (2,8) es cruzado?

EEDD GRADO EN ING. INFORMATICA - UCO

Coste total del árbol: ¿?

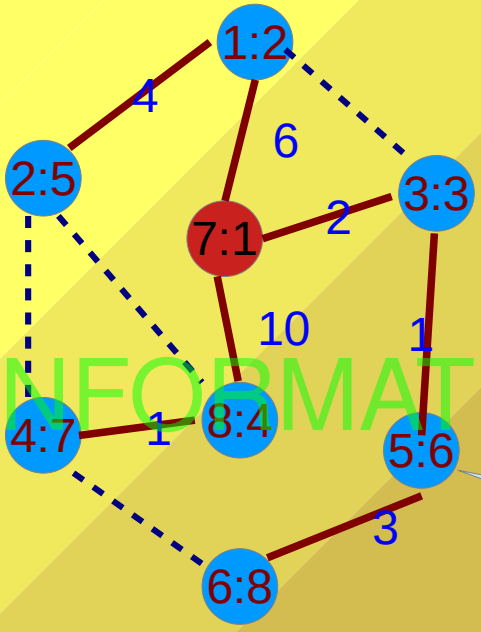
En caso de varias alternativas coger la etiqueta menor

Recorrido en anchura



Cola

1	**7:7**
2	**1:7**,3:7,8:7
3	**3:7**,8:7,2:1,3:1
4	**8:7**,2:1,3:1,5:3
5	**2:1**,3:1,5:3,2:8,4:8
-	3:1,5:3,2:8,4:8,4:2
6	**5:3**,2:8,4:8,4:2
-	2:8,4:8,4:2,6:5
7	**4:8**,4:2,6:5
-	4:2,6:5
8	**6:5**
-	6:4



del árbol
cruzado

"6" llegando desde "4"

"5" se procesa el sexto

Coste total del árbol:
4+6+2+10+1+1+3=27

En caso de varias alternativas coger la etiqueta menor

EEDD GRADO EN ING. INFORMÁTICA - UCO

Ordenacion Topológica

Dado el siguente Makefile,

a) Generar el correspondiente grafo dirigido donde un lado x1 → x2 significa x2 depende de que x1 ya esté realizado con antelación.

b) Obtener la secuencia de tareas a realizar que asegurá que se cumplan todas las dependencias (ordenación topológica).

Makefile:

```
# no ponemos los comandos para simplificar.

all: main.x libFuncs.a
main.x : main.o aux.o libFuncs.a
main.o : main.c aux.h funcs.h
aux.o : aux.h aux.c funcs.h
libFuncs.a: funcs1.o funcs2.o
funcs1.o : funcs1.c funcs.h
funcs2.o : funcs2.c funcs.h
```

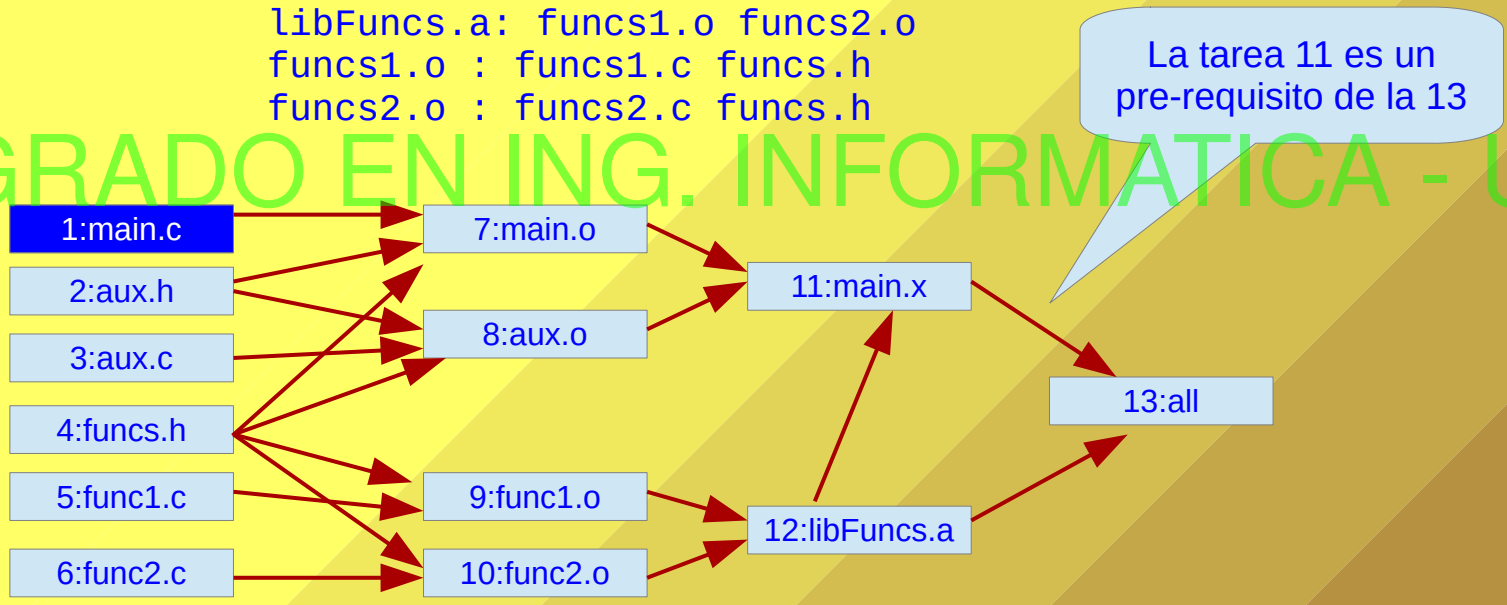
EEDD - GRADO EN ING. INFORMATICA - UCO

Ordenación Topológica

Primero: construir el grafo

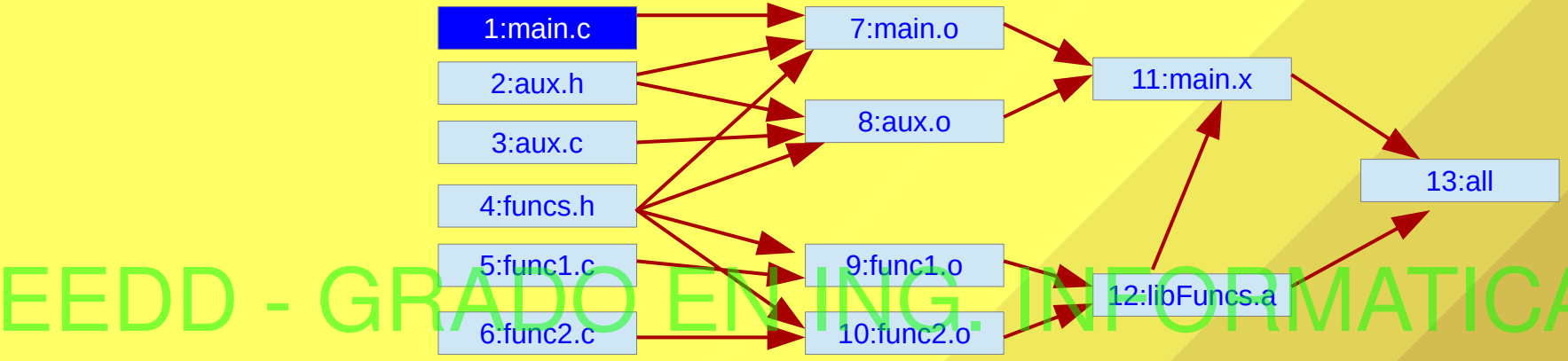
```
all: main.x libFuncs.a
main.x : main.o aux.o libFuncs.a
main.o : main.c aux.h funcs.h
aux.o : aux.h aux.c funcs.h
libFuncs.a: funcs1.o funcs2.o
funcs1.o : funcs1.c funcs.h
funcs2.o : funcs2.c funcs.h
```

EEDD - GRADO EN ING. INFORMATICA - UCO

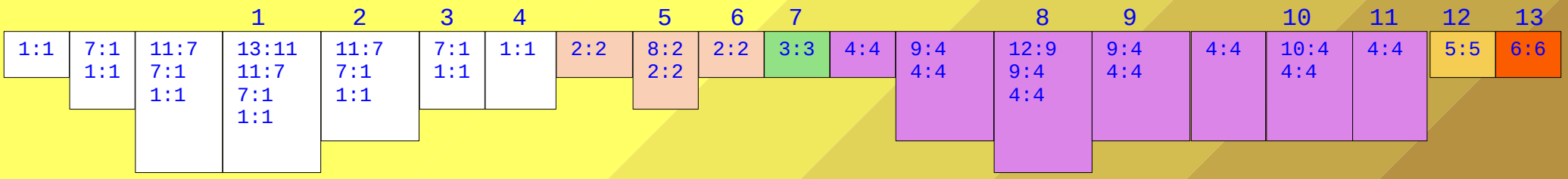


Ordenación Topológica

Segundo: recorrido en profundidad post-fijo.



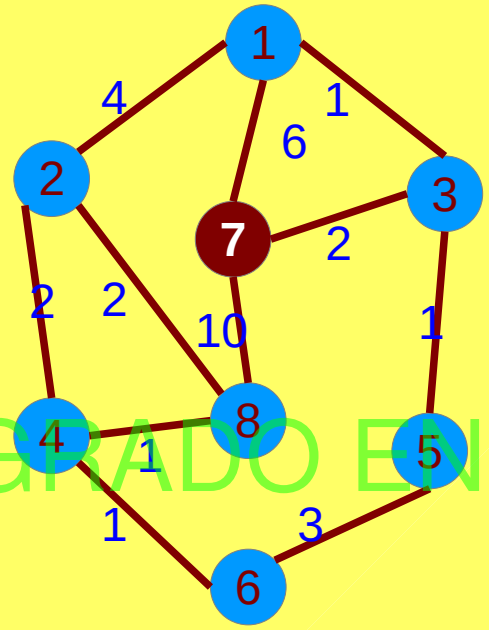
EEDD - GRADO EN ING. INFORMATICA - UCO




```
01:{13}
02:{11,13}
03:{7,11,13}
04:{1,7,11,13}
05:{8,1,7,11,13}
06:{2,8,1,7,11,13}
```

```
07:{3,2,8,1,7,11,13}
08:{12,3,2,8,1,7,11,13}
09:{9,12,3,2,8,1,7,11,13}
10:{10,9,12,3,2,8,1,7,11,13}
11:{4,10,9,12,3,2,8,1,7,11,13}
12:{5,4,10,9,12,3,2,8,1,7,11,13}
13:{6,5,4,10,9,12,3,2,8,1,7,11,13}
```

AAM: Algoritmo de Prim

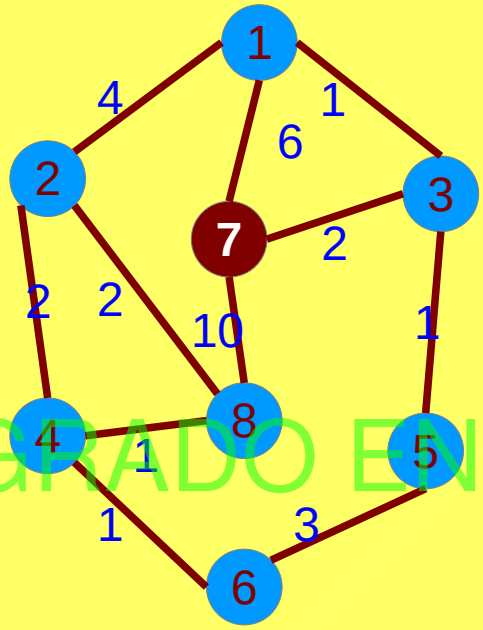


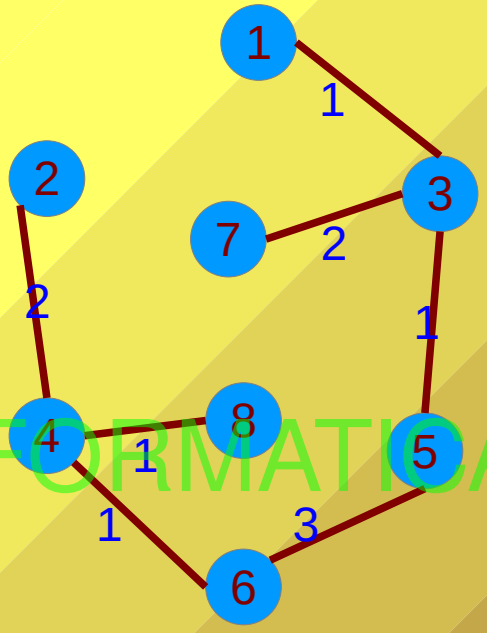
EEDD - GRADO EN ING. INFORMÁTICA - UCO

U	A	V	C
1	0	7	6
2	0	0	-
3	**0**	**7**	**2**
4	0	0	-
5	0	0	-
6	0	0	-
7	1	0	-
8	0	7	10

Coste total=

AAM: Algoritmo de Prim



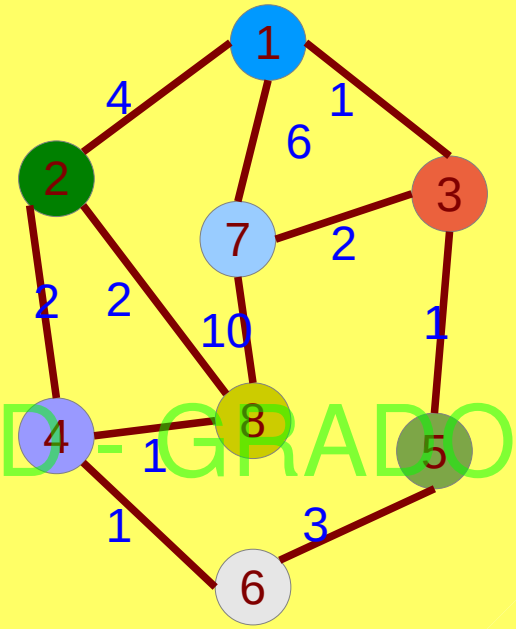


EEDD - GRADO EN ING. INFORMÁTICA - UCO

U	A	V	C	A	V	C	A	V	C	A	V	C	A	V	C	A	V	C	A	V	C	A	V	C
1	0	7	6	**0**	**3**	**1**	1	3	1	1	3	1	1	3	1	1	3	1	1	3	1	1	3	1
2	0	0	-	0	0	-	0	1	4	0	1	4	0	1	4	0	4	2	**0**	**4**	**2**	1	4	2
3	**0**	**7**	**2**	1	7	2	1	7	2	1	7	2	1	7	2	1	7	2	1	7	2	1	7	2
4	0	0	-	0	0	-	0	0	-	0	0	-	**0**	**6**	**1**	1	6	1	1	6	1	1	6	1
5	0	0	-	0	3	1	**0**	**3**	**1**	1	3	1	1	3	1	1	3	1	1	3	1	1	3	1
6	0	0	-	0	0	-	0	0	-	**0**	**5**	**3**	1	5	3	1	5	3	1	5	3	1	5	3
7	1	0	-	1	0	-	1	0	-	1	0	-	1	0	-	1	0	-	1	0	-	1	0	-
8	0	7	10	0	7	10	0	7	10	0	7	10	0	7	10	**0**	**4**	**1**	1	4	1	1	4	1

Coste total= 11

AAM: Algoritmo de Kruskal

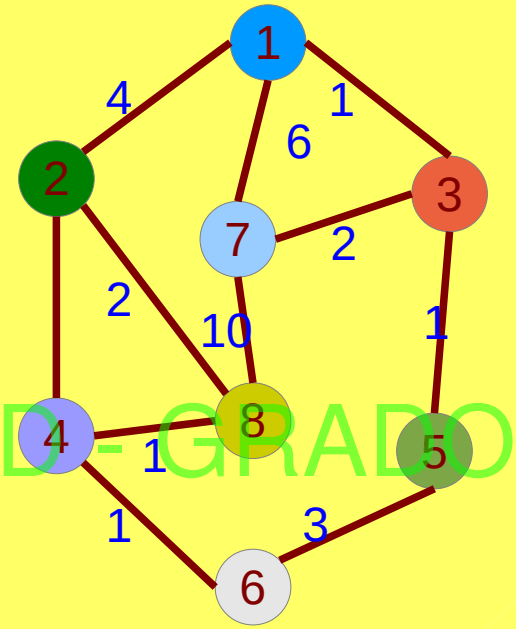


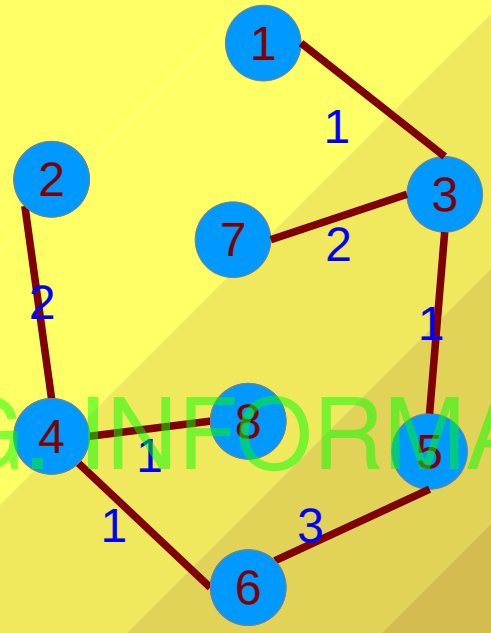
EEDD - GRADO EN ING. INFORMATICA - UCO

Coste total =

Orden lexicográfico: peso, nodo1, nodo2

Algoritmo de Kruskal





EEDD - GRADO EN INGENIERIA INFORMATICA - UCO

Coste total = 11

1:(1,3) 1:(3,5) 1:(4,6) 1:(4,8) 2:(2,4) 2:(2,8) 2:(3,7) 3:(5,6) 4:(1,2) 6:(1,7) 10:(7,8)
1:(1,3) 1:(3,5) 1:(4,6) 1:(4,8) 2:(2,4) 2:(2,8) 2:(3,7) 3:(5,6) 4:(1,2) 6:(1,7) 10:(7,8)
1:(1,3) **1:(3,5)** 1:(4,6) 1:(4,8) 2:(2,4) 2:(2,8) 2:(3,7) 3:(5,6) 4:(1,2) 6:(1,7) 10:(7,8)
1:(1,3) 1:(3,5) **1:(4,6)** 1:(4,8) 2:(2,4) 2:(2,8) 2:(3,7) 3:(5,6) 4:(1,2) 6:(1,7) 10:(7,8)
1:(1,3) 1:(3,5) 1:(4,6) **1:(4,8)** 2:(2,4) 2:(2,8) 2:(3,7) 3:(5,6) 4:(1,2) 6:(1,7) 10:(7,8)
1:(1,3) 1:(3,5) 1:(4,6) 1:(4,8) **2:(2,4)** 2:(2,8) 2:(3,7) 3:(5,6) 4:(1,2) 6:(1,7) 10:(7,8)
1:(1,3) 1:(3,5) 1:(4,6) 1:(4,8) 2:(2,4) 2:(2,8) **2:(3,7)** 3:(5,6) 4:(1,2) 6:(1,7) 10:(7,8)
1:(1,3) 1:(3,5) 1:(4,6) 1:(4,8) 2:(2,4) 2:(2,8) 2:(3,7) **3:(5,6)** 4:(1,2) 6:(1,7) 10:(7,8)

Orden lexicográfico: peso < nodo1 < nodo2