

Grafs

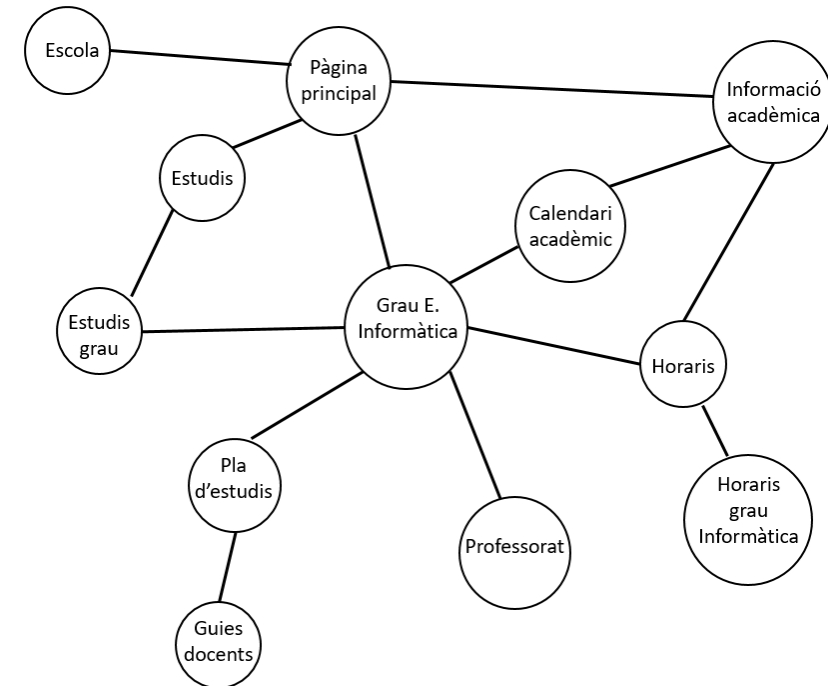
Sessió 13

Representació dels grafs

	Pg pral.	Escola	Estudis	info. acadèmica	e.graus	grau e. inf.	calendari	pla estudis	guies docents	professorat	horaris	horaris inf.
Pg pral.	0	1	1	1	0	1	0	0	0	0	0	0
Escola	1	0	0	0	0	0	0	0	0	0	0	0
Estudis	1	0	0	0	1	0	0	0	0	0	0	0
info. acadèmica	1	0	0	0	0	0	1	0	0	0	1	0
e.graus	0	0	1	0	0	1	0	0	0	0	0	0
grau e. inf.	1	0	0	0	1	0	1	1	0	1	1	0
calendari	0	0	0	1	0	1	0	0	0	0	0	0
pla estudis	0	0	0	0	0	1	0	0	1	0	0	0
guies docents	0	0	0	0	0	0	0	1	0	1	0	0
professorat	0	0	0	0	0	1	0	0	1	0	0	0
horaris	0	0	0	1	0	1	0	0	0	0	0	1
horaris inf.	0	0	0	0	0	0	0	0	0	0	1	0

• Forma 1: Matriu d'adjacència (MA)

- Cada fila i cada columna representa la llista de nodes
- A cada casella posem si hi ha una aresta que uneix els dos nodes (1) o no hi ha aresta (0)



```
class Graf
{
public:
    Graf();
    Graf(const vector<string>& nodes, const vector<vector<int>>& matriu_adj);
    ~Graf();

    int getNumNodes() { return m_numNodes; }
    void inserirAresta(int posNode1, int posNode2);
    void eliminarAresta(int posNode1, int posNode2);
    void afegirNode(string& node);
    void eliminarNode(string& node);
    void mostra();

private:
    vector<string> m_nodes;
    vector<vector<int>> m_matriuAdj;
    size_t m_numNodes;
    size_t m_numArestes;
};
```

size_t pot guardar la mida màxima de qualsevol tipus d'objecte. Si en comptes de size_t utilitzes un unsigned int i la mida excedeix UINT_MAX el programa fallaria.

1. L'empresa de TMB ens ha fet uns encàrrecs sobre la xarxa de metro de Barcelona. Per poder-los portar a terme, representarem aquesta xarxa com un graf. La informació està desada en forma de parelles de parades que estan unides per un tram de metro. Ara ens cal crear la matriu d'adjacència corresponent.

Implementa un mètode a la classe Graf que serveixi **per crear i omplir la matriu d'adjacència**, aquest mètode el cridarem des del constructor que rep com a paràmetre el vector de nodes.

Podem fer servir aquest constructor... ?

```
Graf(const vector<string>& nodes, const vector<vector<int>>& matriu_adj);
```

Quina diferència hi ha?

Exercici 1

Podem fer servir aquest constructor... ?

```
Graf(const vector<string>& nodes, const vector<vector<int>>& matriu_adj);
```

Quina diferència hi ha?

```
vector<vector<int>> matAdj{
{ 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0 },
{ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 },
{ 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0 },
{ 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0 },
{ 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0 },
{ 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 },
{ 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0 },
};
```

```
parells_parades = {
{0,1},{0,2},{0,3},{0,5},{2,4},{3,6},{
3,10},{4,5},{5,6},{5,7},{5,10},{7,8},
{8,9},{10,11}};
```

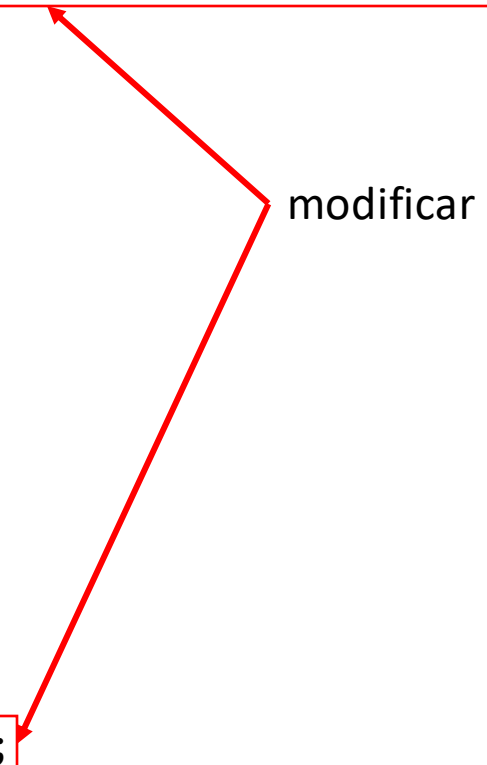
Implementació

```
class Graf
{
public:
    Graf();
    Graf(const vector<string>& nodes, const vector<vector<int>>& parelles_nodes );
    ~Graf();

    size_t getNumNodes() { return m_numNodes; }
    void inserirAresta(int posNode1, int posNode2);
    void eliminarAresta(int posNode1, int posNode2);
    void afegirNode(const string& node);
    void eliminarNode(const string& node);
    void mostra();

private:
    vector<string> m_nodes;
    vector<vector<int>> m_matriuAdj;
    size_t m_numNodes;
    size_t m_numArestes;
    void crearMatriu(const vector<vector<int>>& parelles);

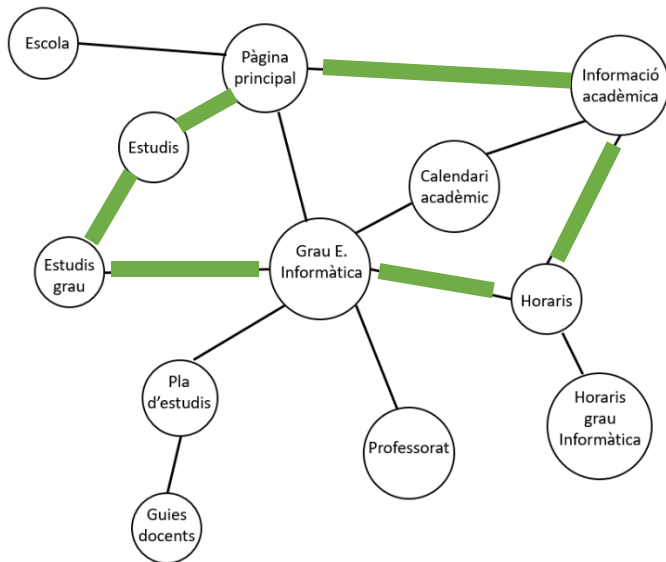
};
```



modificar

Exercici 2 – Exemple de Cicle

Vèrtex inici i final: Pàgina Principal

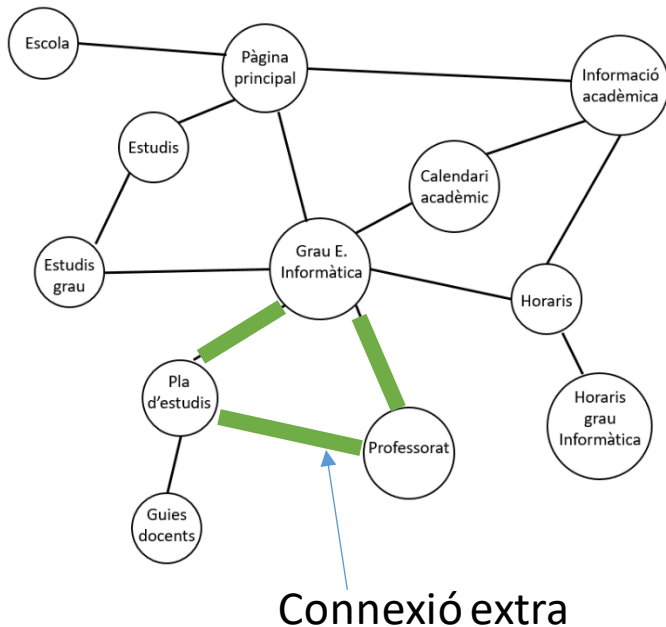
[illegible]

2. La primera cosa que volen saber per poder veure el grau de solapament entre línies és saber quins conjunts de 3 parades formen un cicle.

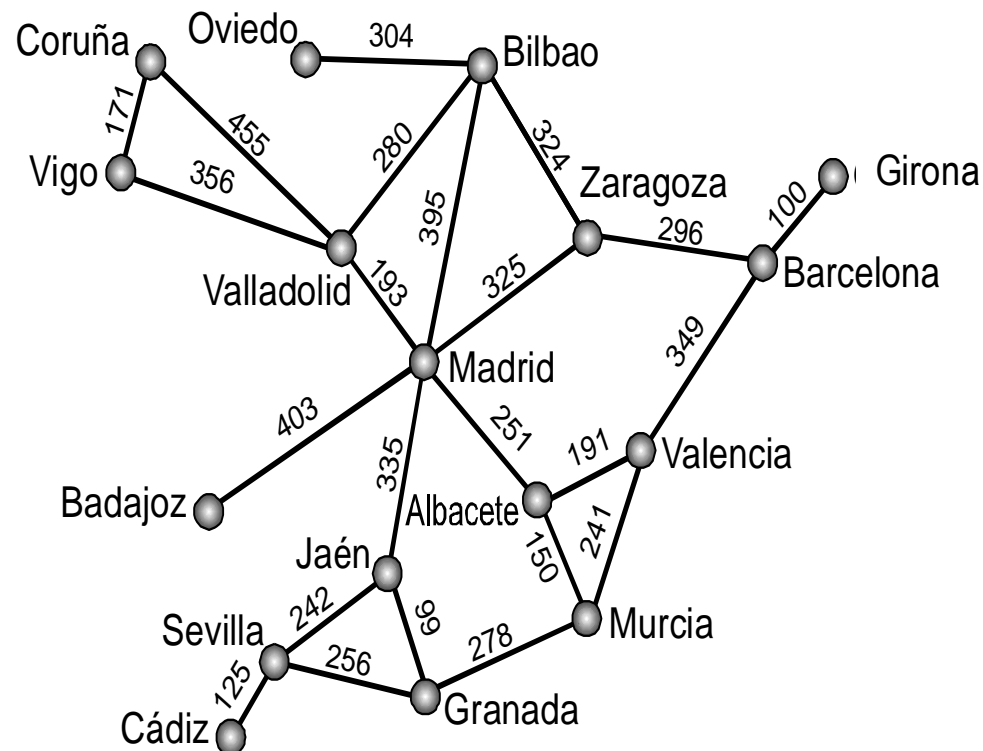
Implementa un mètode per poder respondre aquesta pregunta. Aquest mètode haurà de retornar un vector amb els conjunts de tres parades que formen un cicle.

Definició: Un **cicle** és un camí on el vèrtex inicial i el final és el mateix, però no hi ha cap altre vèrtex que es repeteixi.

Exercici 2 – Exemple de Cicle 3 Parades

[illegible]

Definició: En el cas que les arestes portin associada una informació numèrica que representi el cost necessari per recórrer aquella aresta, direm que el graf és **ponderat**.



3. Suposem ara que també ens han passat les distàncies entre parades.

```
vector<int> pesos =  
{511,995,425,2271,1782,787,375,1603,493,960,1212,499,1181,647,494,124,1794,1607,  
536,1781,311,708,186,882,1796,1132,1090,1145,2393,1730,620,442,1654,706,292,305,  
1204,617,20,1242,184,655,2034,1170,281,979,1569,990,461,1702,1024,986,647,113,  
1585,520,962,1339,662,851,1695,1337,622,1290};
```

- Com hauríem de representar aquesta informació a la classe graf?
- Implementa la classe GrafPonderat i modifica-hi el mètode de l'exercici 1 per introduir aquesta informació. Afegeix els altres mètodes i constructor que consideris convenient.

Implementació

```
class GrafPonderat : Graf {  
public:
```

```
    GrafPonderat();
```

```
    GrafPonderat(const vector<string>& nodes,  
                 const vector<vector<int>>& parelles_nodes, ...);
```

```
    ~GrafPonderat();
```

```
    // Quin mètode més podem afegir?
```

```
private:
```

```
    void crearMatriu(const vector<vector<int>>& parelles, ...);
```

```
};
```

modificar
respecte
equivalent de
Graf

