1 Que es un Grafo?

Dicho formalmente un Grafo es un conjunto de Vértices y Arcos.

$$G = \{V, A\}$$

Pero un Vértice puede representar cualquier cosa (ciudades, personas, objetos, etc) y los Arcos son las relaciones que tienen esos Vértices, desde este punto de vista un Grafo puede representar cualquier cosa que queramos (para los entusiastas la vida es un Grafo).

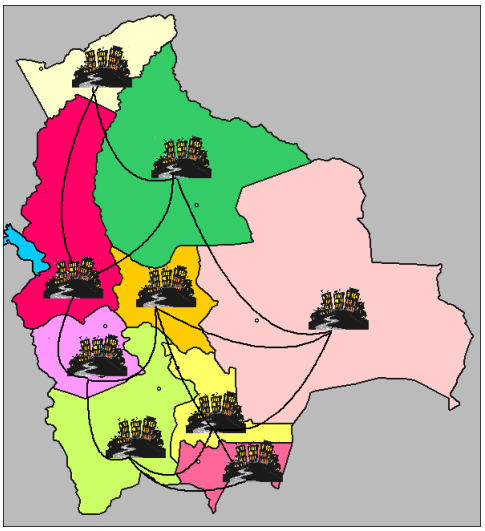


Figura 1:Grafo donde las ciudades son Vértices y las carreteras los Arcos.

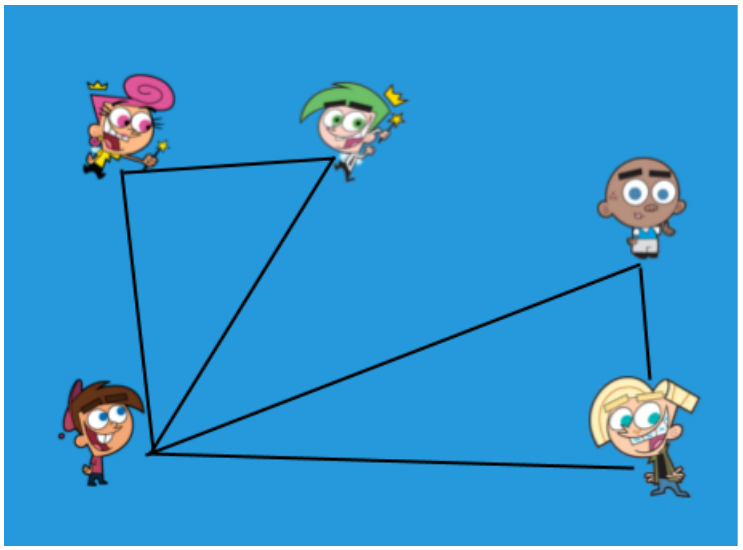
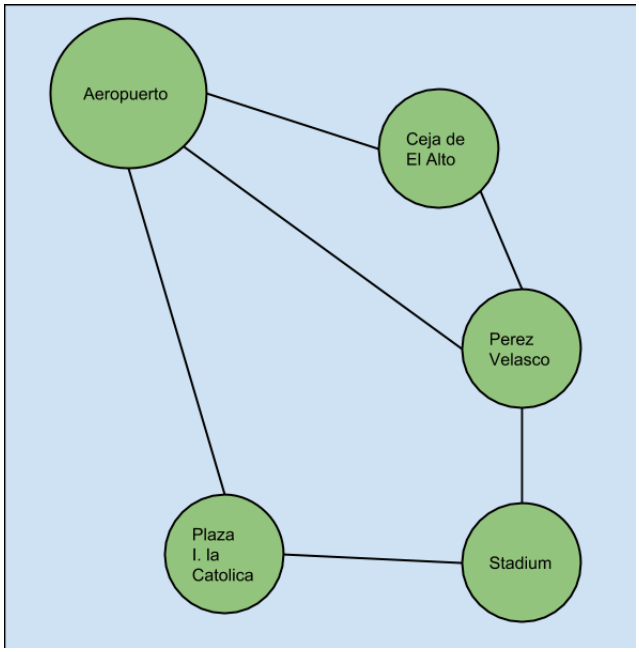


Figura 2:Grafo donde las personas son Vértices y los Arcos representan amistad.

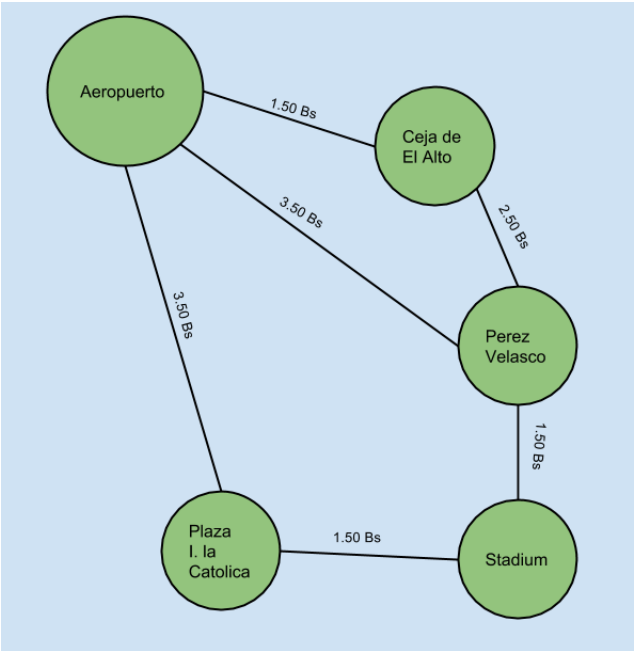
Teniendo en cuenta esto nos podemos dar cuenta que desde que nos levantamos estamos modelando grafos, estamos recorriendo grafos.

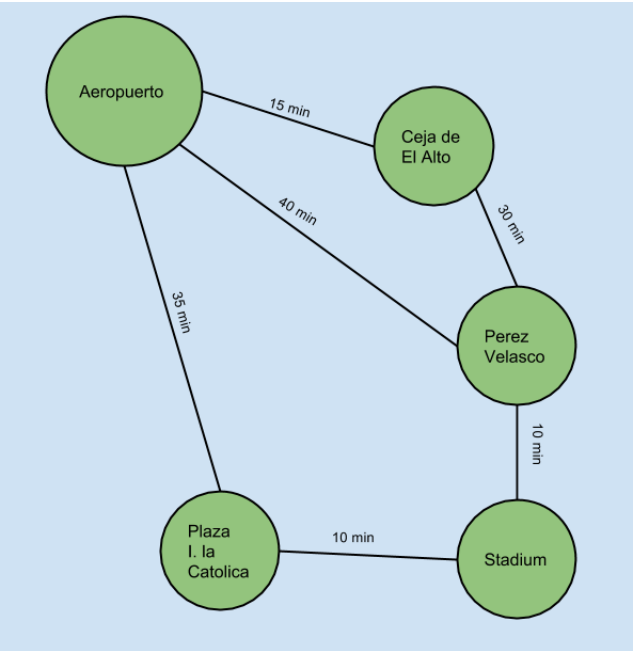
Ejemplo 1.

Una persona que se encuentre en el Stadium y quiera llegar al Aeropuerto, su cerebro inconcientemente ya esta modelando un grafo donde tenemos los lugares(Vertices) y las calles que nos llevan a ellos (Arcos):

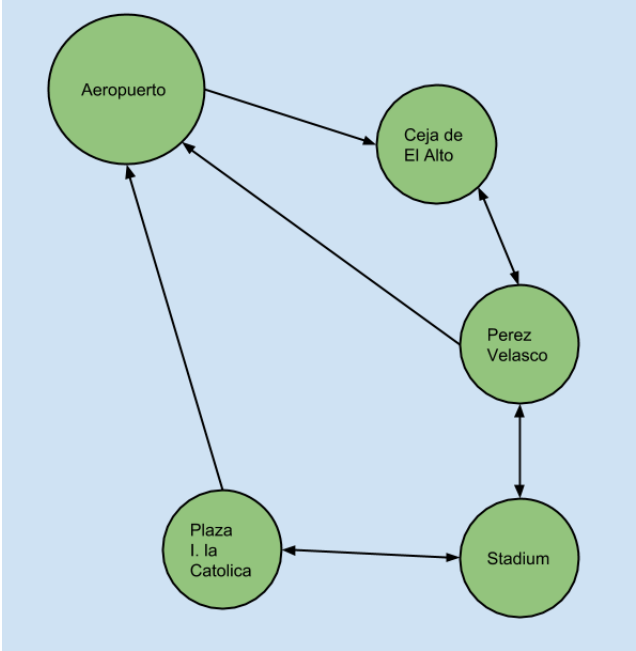


Ahora podemos ver los posibles caminos que nos llevan desde el Stadium hasta el Aeropuerto, pero ?que pasa si queremos saber por que camino gasto menos dinero o por cual llego mas rápido?





Cuando a los arcos se le da un valor se llama, Grafo Ponderado. Ahora, si es que nosotros vamos en nuestro propio automovil seria bueno saber que caminos podemos tomar:



Cuando el grafo tiene caminos que solo se pueden usar de un sentido,Ej. Perez Velasco-Aeropuerto, se le llama Grafo Dirigido.

Como podemos ver tenemos 4 tipos de grafos:

- Grafo dirigido-ponderado
- Grafo no dirigido-no ponderado
- Grafo dirigido-no ponderado
- Grafo no dirigido-ponderado

Entonces como vimos con grafos podemos representar muchísimas cosas de la vida diaria, es por esto que en las Ciencias de la Computación un Grafo es usado como una Estructura de Datos, en concreto un tipo de Dato Abstracto, es decir un conjunto de datos sobre los que podemos hacer operaciones definas. Muchos problemas, especialmente de las Olimpiadas y Competencias, tiene su solución con el uso de los grafos.

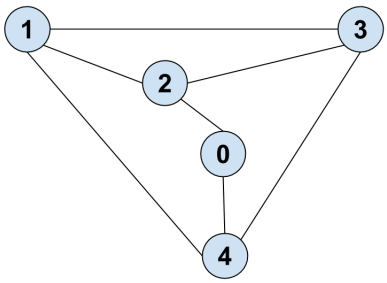
En este punto en el que ya sabemos que es un grafo, tenemos la idea de como modelarlos, surge la duda ?Como puedo representar un grafo al momento de programar?

2 Representacion de un grafo

Con todo lo aprendido estamos listos, para representar el grafo, para esto usaremos las 2 formas mas comunes de represnetar un grafo:

2.1 Matriz de Adyacencia

La forma mas fácil (pero no la mas eficiente) de representar un grafo es usando una Matriz, como podemos ver a continuación:

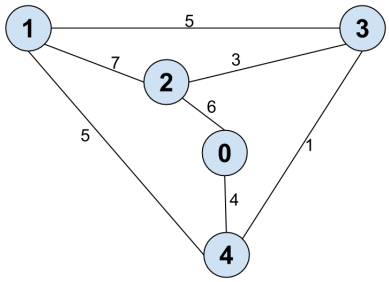


I\J	0	1	2	3	4
0	0	0	1	0	1
1	0	0	1	1	1
2	1	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

Si es un grafo sin pesos, podemos usar una Matriz booleana donde marcamos '1' o "true" si es que existe un arco del vértice 'i' al vértice 'j' y colocamos '0' o "false" en el caso que no haya un arco entre esos vértices, acá tenemos el código general para representar un grafo como Matriz de Adyacencia:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main(){
    bool AdjMatrix[100][100];
    int Nodes, Edges;
    scanf("%d %d",&Nodes, &Edges);
    //Numero de Vertices y Arcos
    memset(AdjMatrix, false, sizeof(AdjMatrix));
    for (int i = 0; i < Edges; ++i){
        int u,v;
        scanf("%d %d",&u,&v);
        //Camino de 'u' a 'v';
        AdjMatrix[u][v]=true;
        AdjMatrix[v][u]=true;
    }
}
```

Si los arcos tienen pesos, es decir si es ponderado, lo colocamos en la casilla (i,j), como se muestra a continuación:

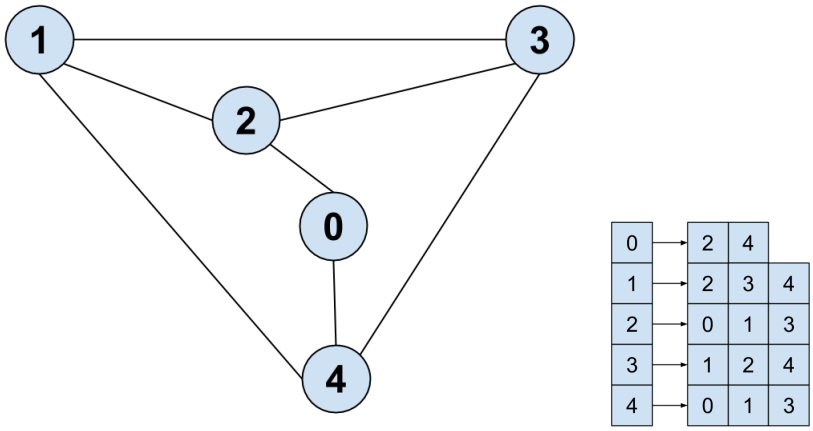


I\J	0	1	2	3	4
0	0	0	6	0	4
1	0	0	7	5	5
2	6	7	0	3	0
3	0	5	3	0	1
4	4	5	0	1	0

```cpp
#include <bits/stdc++.h>
using namespace std;
int main(){
    int AdjMatrix[100][100];
    int Nodes, Edges;
    scanf("%d %d",&Nodes, &Edges);
    //Numero de Vertices y Arcos
    memset(AdjMatrix, 0, sizeof(AdjMatrix));
    for (int i = 0; i < Edges; ++i){
        int u,v,w;
        scanf("%d %d %d",&u,&v,&w);
        //Camino de 'u' a 'v' con peso 'w';
        AdjMatrix[u][v]=w;
        AdjMatrix[v][u]=w;
    }
}
```

2.2 Lista de Adyacencia

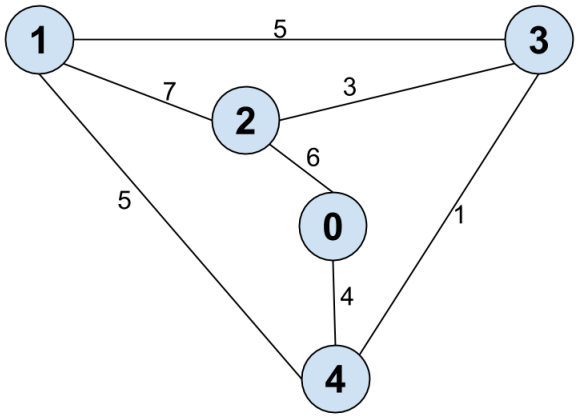
Esta es la forma mas usual de representación de un grafo, la detallamos a continuación:

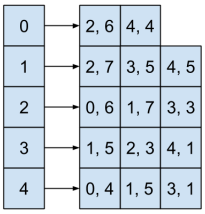


Para representarlo en forma de Lista de Adyacencia usamos un vector de vectores de enteros, en el cual la pocision 'i' tiene un vector con todos los nodos que tengan un arco hacia 'i'. El codigo en C++ es el siguiente:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main(){
    vector<vector<int> > AdjList;
    int Nodes, Edges;
    scanf("%d %d",&Nodes, &Edges);
    //Numero de Vertices y Arcos
    AdjList.assign(Nodes, vector<int>());
    for (int i = 0; i < Edges; ++i){
        int u,v;
        scanf("%d %d",&u,&v);
        //Camino de 'u' a 'v';
        AdjList[u].push_back(v);
        AdjList[v].push_back(u);
    }
}
```

En el caso de un grafo ponderado usaremos un un vector de vectores igual que antes, solo que ahora en lugar de guardar un entero guardaremos un par de enteros con el peso y el nodo:






```cpp
#include <bits/stdc++.h>
using namespace std;
int main(){
    vector<vector<pair<int,int> > > AdjList;
    int Nodes, Edges;
    scanf("%d %d",&Nodes, &Edges);
    //Numero de Vertices y Arcos
    AdjList.assign(Nodes, vector<pair<int,int> >());
    for (int i = 0; i < Edges; ++i){
        int u,v,w;
        scanf("%d %d %d",&u,&v,&w);
        //Camino de 'u' a 'v' con peso 'w';
        AdjList[u].push_back(make_pair(v,w));
        AdjList[v].push_back(make_pair(u,w));
    }
}
```