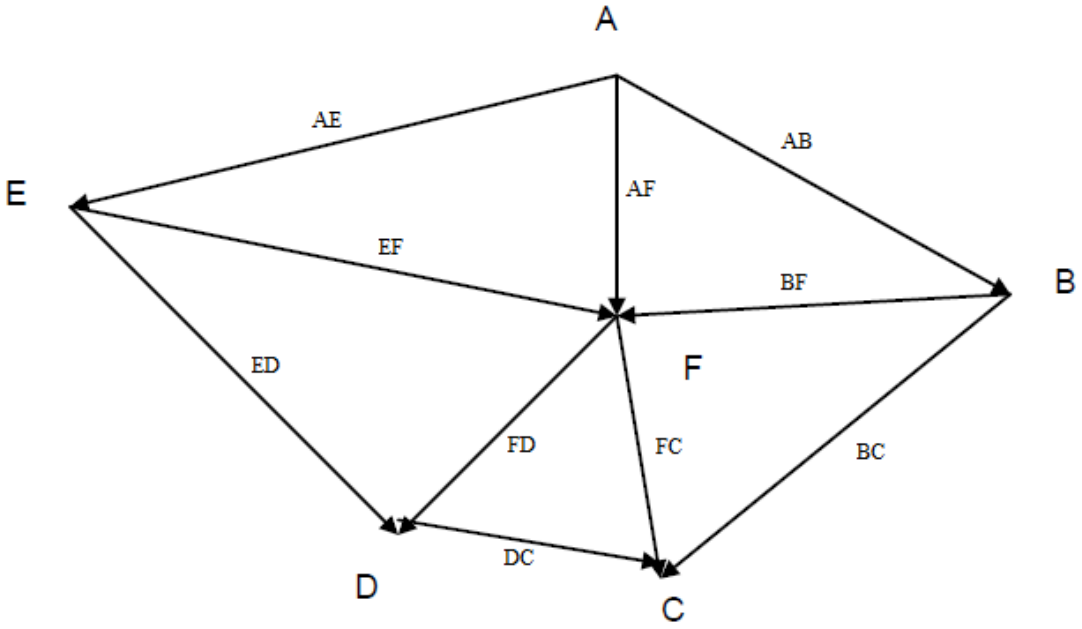
Graphes

1 Considérons le graphe suivant :



Dessinez sa représentation sous forme de :

a) Liste d'arcs
b) Matrice d'adjacence
c) Liste d'adjacence

2. **Implémentation**. Sur moodle, vous trouverez un projet Graph qui contient un squelette de code pour implémenter une liste d'adjacence, une liste d'arcs et une matrice d'adjacence. Ce projet contient :

- Un fichier `flight.xml` contenant des informations à propos de vols aériens. Ce fichier contient un certain nombre d'aéroports (élément `airport`) identifié par un code appelé code iata. Pour chaque aéroport, on a une liste de vol partant de cet aéroport (élément `flight`). Pour chaque vol, on a le code iata de l'aéroport de destination ainsi que la compagnie (attribut `airline`) .
- Une classe abstraite `Graph` qui va lire le document xml et construira un graphe en utilisant les méthodes `ajouterSommet` et `ajouterArc`. Ces deux méthodes sont abstraites et doivent être implémentées dans les classes `ListeDArc`, `ListeDAdjacence` et `MatriceDAdjacence`.
- Une classe `Airport` jouant le rôle de Sommet et une classe `Flight` jouant le rôle d'arc.

Dans les classes `ListeDArc`, `ListeDAdjacence` et `MatriceDAdjacence`, implémentez les méthodes manquantes et remplissez le tableau ci-dessous en donnant les coûts (n : nombre de sommets, m : le nombre d'arcs) :

	Liste d'arcs	Liste d'adjacence	Matrice d'adjacence
`ajouterSommet(Sommet s)`			
`ajouterArc(Arc a)`			
`arcsSortants(Sommet s)`			
`sontAdjacents(Sommet s1, Sommet s2)`			

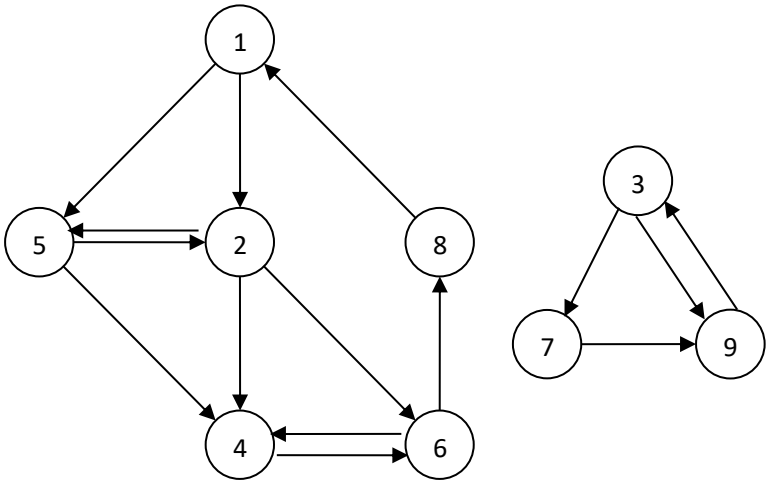
Testez vos méthodes avec chacune des 3 classes. Le résultat attendu est le suivant :

```
Flight [source=JFK, destination=MUC, airline=Lufthansa]
Flight [source=JFK, destination=LAX, airline=American Airlines]
Flight [source=JFK, destination=LHR, airline=American Airlines]
Flight [source=JFK, destination=IST, airline=Turkish Airlines]
Flight [source=JFK, destination=FCO, airline=American Airlines]
Flight [source=JFK, destination=ORD, airline=American Airlines]
Flight [source=JFK, destination=BCN, airline=American Airlines]
Flight [source=JFK, destination=PEK, airline=Air China]
Flight [source=JFK, destination=MAD, airline=American Airlines]
Flight [source=JFK, destination=ATL, airline=Air France]
Flight [source=JFK, destination=AMS, airline=Delta Air Lines]
Flight [source=JFK, destination=FRA, airline=Air France]
Flight [source=JFK, destination=CDG, airline=American Airlines]
Flight [source=JFK, destination=DEN, airline=JetBlue Airways]
Flight [source=JFK, destination=DXB, airline=JetBlue Airways]
Flight [source=JFK, destination=DFW, airline=American Airlines]
----------------------
false
true
```

Voici quelques explications supplémentaires sur les attributs de la matrice d'adjacence :

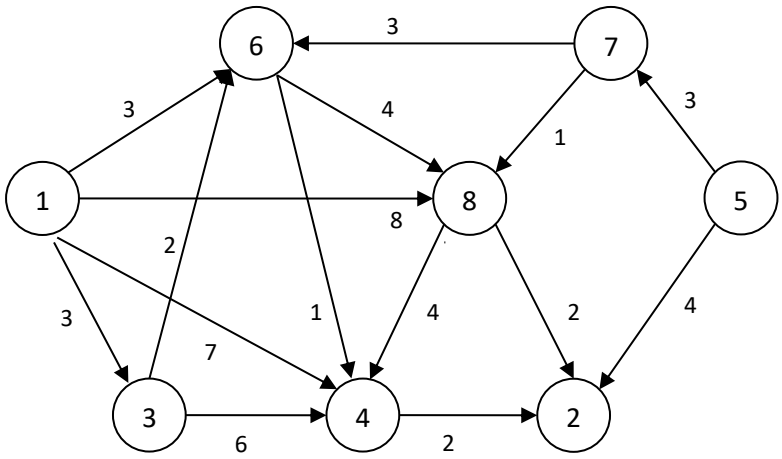
- `correspondanceIndiceAirport` : ce dictionnaire fait la correspondance entre les indices des lignes/colonnes de la matrice et les codes iata des différents aéroports. Pour le fichier flight.xml, cette structure devrait contenir les infos suivantes :
  `{0=FCO, 1=AMS, 2=FRA, 3=JFK, 4=DEN, 5=IST, 6=STN, 7=DME, 8=LAX, 9=LGW, 10=ORD, 11=MAD, 12=CDG, 13=EWR, 14=ATL, 15=DFW, 16=DXB, 17=IAH, 18=LHR, 19=MUC, 20=PEK, 21=BCN}`
- `correspondanceAirportIndice` : ce dictionnaire est l'inverse du dictionnaire précédent. Il va retenir la correspondance entre les codes iata des aéroports avec les indices des lignes/colonnes de la matrice. Pour le fichier flight.xml, cette structure devrait contenir les infos suivantes :
  `{ORD=10, EWR=13, LAX=8, AMS=1, CDG=12, IST=5, DEN=4, STN=6, BCN=21, JFK=3, DXB=16, MAD=11, IAH=17, FCO=0, FRA=2, DFW=15, LHR=18, PEK=20, ATL=14, MUC=19, DME=7, LGW=9}`
- `matrice` : la matrice d'adjacence. A l'indice (i,j) de cette matrice, on retient la compagnie (`airline`) s'il y a un vol entre l'aéroport correspondant à l'indice i et l'aéroport correspondant à l'indice j. Si il n'y a pas de vols entre ces deux aéroports, l'indice (i,j) contient `null`. Par exemple, l'indice (0,0) contiendra `null` car il n'y a pas de vol entre `FCO` et `FCO`. L'indice (0,1) contiendra « `Alitalia` » car Alitalia est la compagnie reliant `FCO` et `AMS`.
  Remarque : il y a au maximum un vol entre un aéroport source et un aéroport destination.

3. Considérons le graphe suivant :



a) Pour ce graphe, dessinez l'arbre obtenu lors d'un parcours en profondeur (DFS) commençant au sommet 1 (les arcs sont essayés dans l'ordre croissant de leur destination).
b) Pour ce graphe, dessinez l'arbre obtenu lors d'un parcours en largeur (BFS) commençant au sommet 1 (les arcs sont essayés dans l'ordre croissant de leur destination). Numérotez les arcs selon l'ordre du parcours.

4 Considérons le graphe ci-dessous :



Appliquez l'algorithme de Dijkstra pour trouver les chemins les plus courts à partir du sommet 1. Donnez toutes les étapes!