II Semaine 2

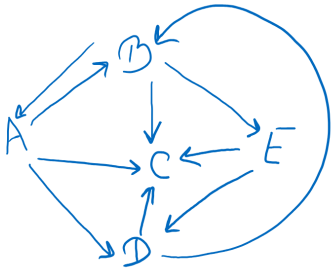
II.1 Lundi 6 avril ♥

1–❑ Considérons le graphe définie le dictionnaire d'adjacence :

```
Dico={"A" :["B","C","D"], "B" :["A","C","E"], "C" :[], "D" :["B","C"], "E" :["C","D"]}
```

a) Faites un dessin du graphe.

Solution:



b) Donner la liste des points qui seront visités si on fait un parcours en largeur en partant de `A`.

Solution: Lors d'un parcours en largeur, on commence par visiter tous les sommets successeurs du sommet de départ, puis tous les successeurs (non encore visités) du sommet suivant... Soit : $A-B-C-D-E$

Informatiquement, on utilise une file (premier entré, premier sorti) :

sommet	sommets en attente	liste des sommets visités
A	[B,C,D]	[A]
B	[C,D]	[A,B]
	[C,D,E]	[A,B]
C	[D,E]	[A,B,C]
D	[E]	[A,B,C,D]
E	[]	[A,B,C,D,E]

c) Donner la liste des points qui seront visités si on fait un parcours en profondeur partant de `B`.

Solution: Lors d'un parcours en profondeur, on commence par visiter un successeur du premier, puis un successeurs du deuxième sommet visité, ... puis on revient au premier et visiter le deuxième successeur, ... Soit : $B-E-D-C-A$

Informatiquement, on utilise une pile (premier entré, dernier sorti) :

sommet	sommets en attente	liste des sommets visités
B	[A,C,E]	[B]
E	[A,C]	[B,E]
	[A,C,D]	[B,E]
D	[A,C]	[B,E,D]
C	[A]	[B,E,D,C]
A	[]	[B,E,D,C,A]

2–❑ Écrire une fonction `sommets(G)` qui renvoie la liste des sommets du dictionnaire d'adjacence `G` d'un graphe.

Solution:

```
1 def sommets(G):
2     L=[c for c in G] # les clés sont les sommets du graphe
3     return L
```