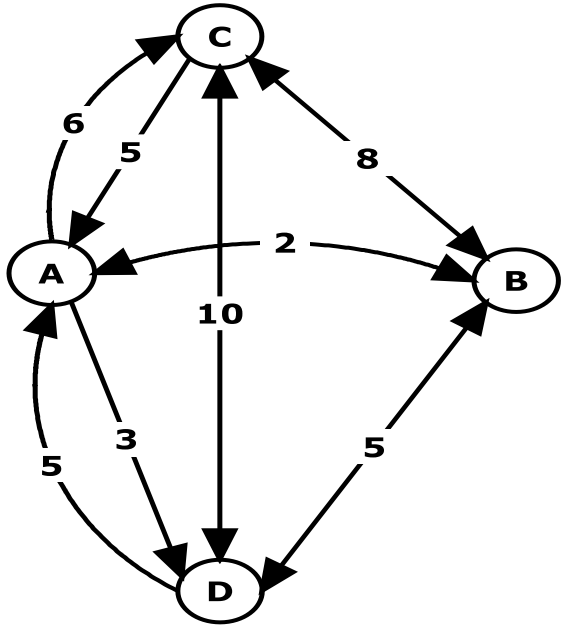
Problème du voyageur de commerce

Problème : trouver les **circuits** de longueur minimale passant par **tous** les sommets d'un graphe **une fois et une seule** (circuit hamiltonien)

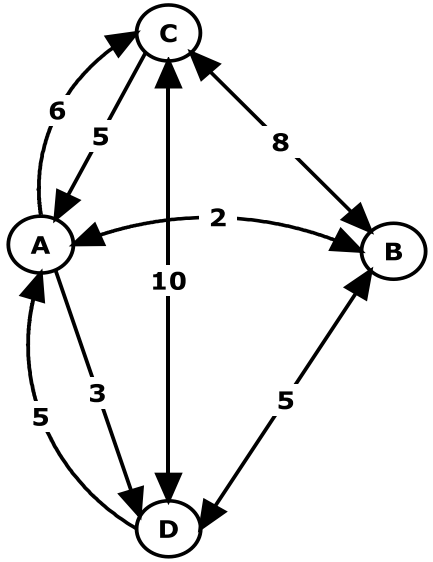
Problème du voyageur de commerce

- « Un voyageur de commerce doit visiter une et une seule fois un nombre fini de villes et revenir à son point d'origine. Trouvez l'ordre de visite des villes qui minimise la distance totale parcourue par le voyageur »



Problème du voyageur de commerce

- « Un voyageur de commerce doit visiter une et une seule fois un nombre fini de villes et revenir à son point d'origine. Trouvez l'ordre de visite des villes qui minimise la distance totale parcourue par le voyageur »

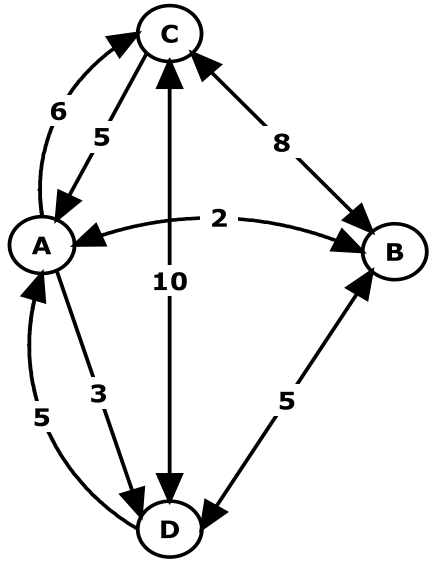


Problèmes de classe NP : problèmes pour lesquels une solution peut être difficile à trouver mais pour lesquels on peut montrer en temps polynomial qu'une solution donnée est optimale ou non. ***Le problème du voyageur de commerce est un problème de classe NP***

Problèmes de classe P : problèmes pour lesquels il existe un algorithme polynomial en la taille des données permettant de les résoudre. ***Les problèmes de recherche de chemin minimum sont des problèmes de classe P.***

Problème du voyageur de commerce

- « Un voyageur de commerce doit visiter une et une seule fois un nombre fini de villes et revenir à son point d'origine. Trouvez l'ordre de visite des villes qui minimise la distance totale parcourue par le voyageur »



Une question à 1 million de dollard !

En 2000, l'institut Clay de mathématiques a publié une liste de sept problèmes, appelés aussi problèmes du millénaire, considérés parmi les plus importants des problèmes mathématiques non résolus.

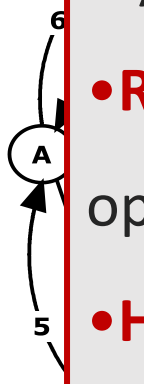
Le problème « P=NP » fait partie de ces problèmes.

On ne connait pas pour l'instant d'algorithme permettant de trouver une solution du problème du voyageur de commerce en temps polynomial *(Si on trouvait un tel algorithme pour ce problème, on pourrait alors résoudre l'ensemble des problèmes NP-complet en temps polynomial).*

Problème du voyageur de commerce

- « Un voyageur de commerce doit visiter une et une seule fois un nombre fini de villes et revenir à son point d'origine. Trouvez l'ordre de visite des villes qui minimise la distance totale parcourue par le

mmets



Il y a donc plusieurs types de méthodes :

- **Recherche exhaustive** : trouver des stratégies pour obtenir les solutions optimales (mais en temps exponentiel)
- **Heuristiques** : trouver des solutions approximatives (souvent des méthodes gloutonnes de bon sens) en temps polynomial

i problèmes du non résolus.

ne du voyageur de commerce en temps polynomial *(Si on trouvait un tel algorithme pour ce problème, on pourrait alors résoudre l'ensemble des problèmes NP-complet en temps polynomial).*

Deux heuristiques (gloutonnes)

Algorithme des plus proches voisins

Algorithme de la meilleure insertion

Heuristique du plus proche voisin

- En partant de B : 23

  $B \xrightarrow{2} A \xrightarrow{3} D \xrightarrow{10} C \xrightarrow{8} B$

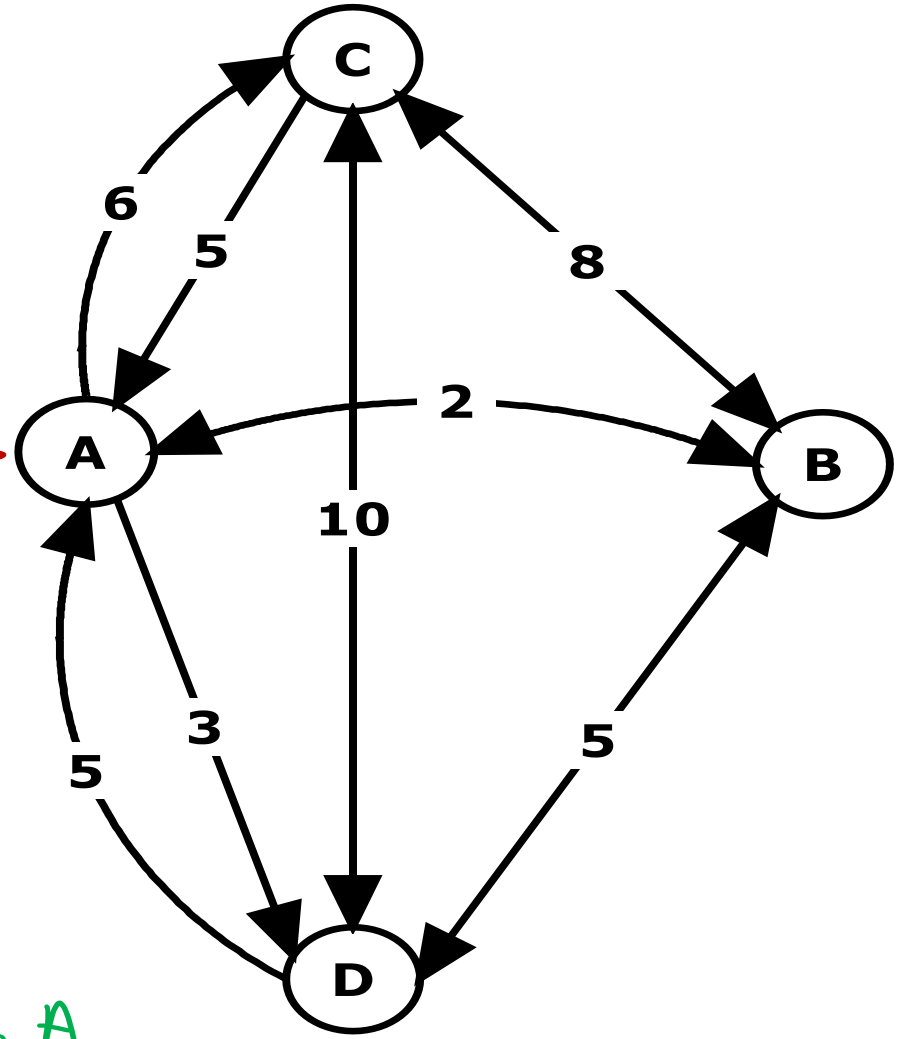
- En partant de C : 22

  $C \xrightarrow{5} A \xrightarrow{2} B \xrightarrow{5} D \xrightarrow{10} C$

- En partant de A : 22

  $A \xrightarrow{2} B \xrightarrow{5} D \xrightarrow{10} C \xrightarrow{5} A$

Plus proche voisin



Circuit de poids minimum : 21

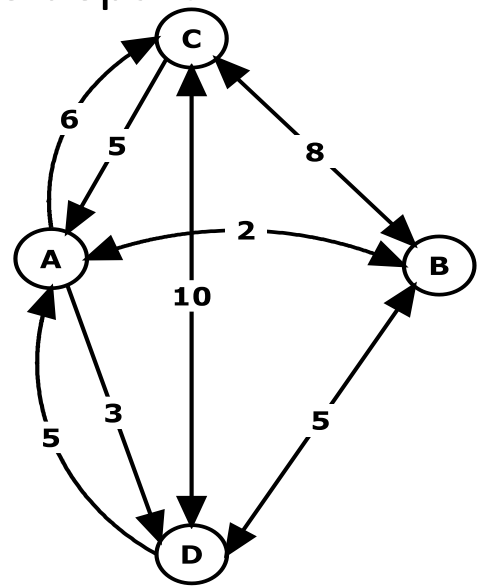
$A \xrightarrow{3} D \xrightarrow{5} B \xrightarrow{8} C \xrightarrow{5} A$

Heuristique du plus proche voisin

- On part d'un sommet arbitraire à partir duquel on va au sommet le plus proche, puis de ce dernier au plus proche… jusqu'à revenir au sommet de départ.

Appliquer la méthode au graphe ci-contre :

- En partant du sommet B : 23
- En partant du sommet C : 22
- En partant du sommet A : 22



- Donnez un exemple de situation (graphe) dans laquelle la méthode du plus proche voisin est peu adaptée.

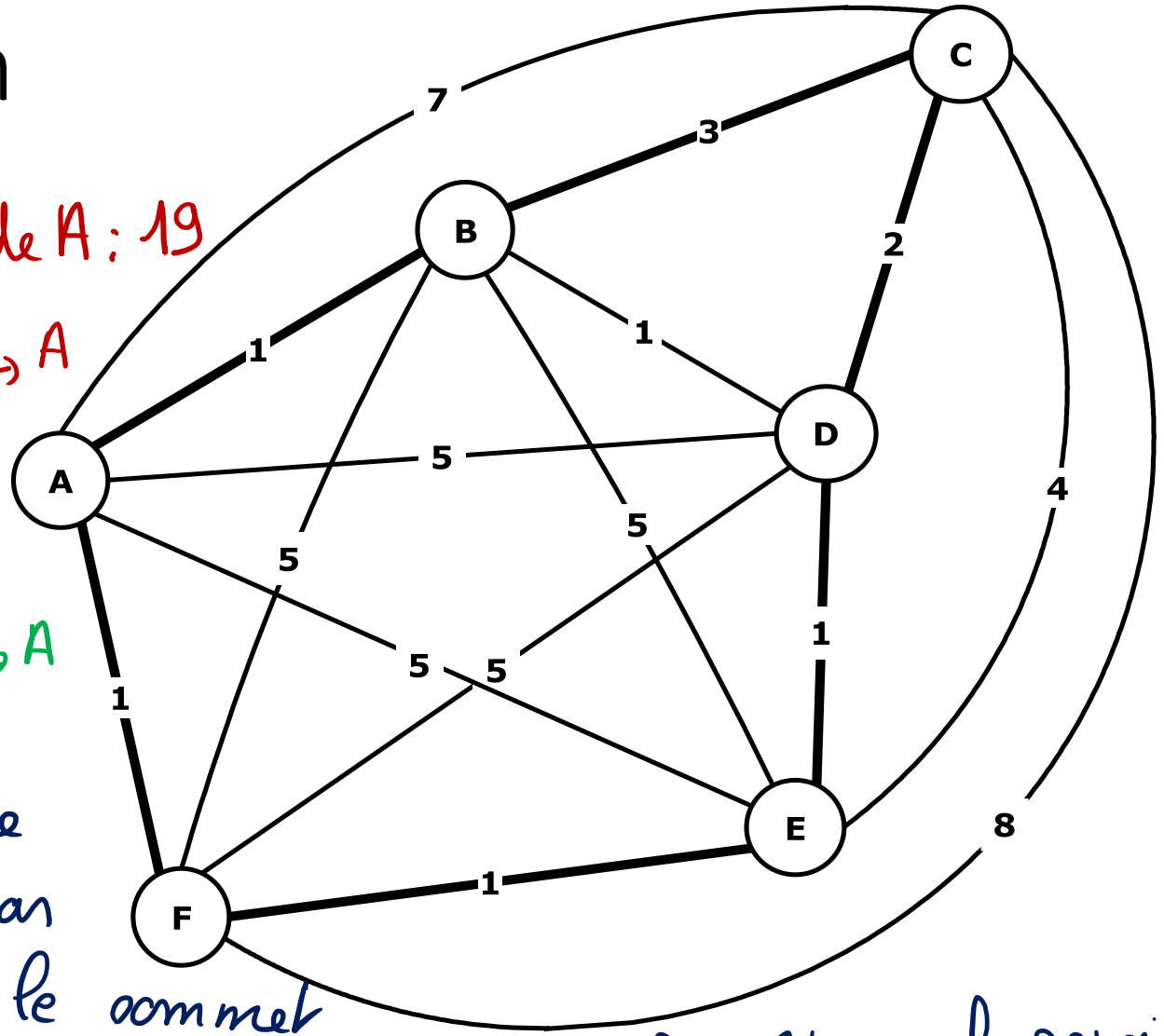
Plus proche voisin

Plus proche voisin en partant de A : 19

A $\xrightarrow{1}$ B $\xrightarrow{1}$ D $\xrightarrow{1}$ E $\xrightarrow{1}$ F $\xrightarrow{8}$ C $\xrightarrow{7}$ A

Circuit (évident) de poids : 9 minimum :

A $\xrightarrow{1}$ B $\xrightarrow{3}$ C $\xrightarrow{2}$ D $\xrightarrow{1}$ E $\xrightarrow{1}$ F $\xrightarrow{1}$ A



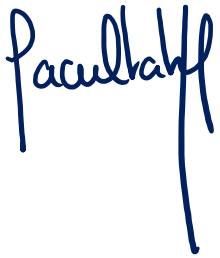
Ici le résultat de l'heuristique du plus proche voisin n'est pas bon car nous avons "ignoré" le sommet C alors que nous en étions "assez" proche (en B) et le coût pour le rejoindre à partir de F est très élevé.

Plus proche voisin

PYTHON

Programmer l'heuristique des plus proches voisins pour le problème du voyageur de commerce.

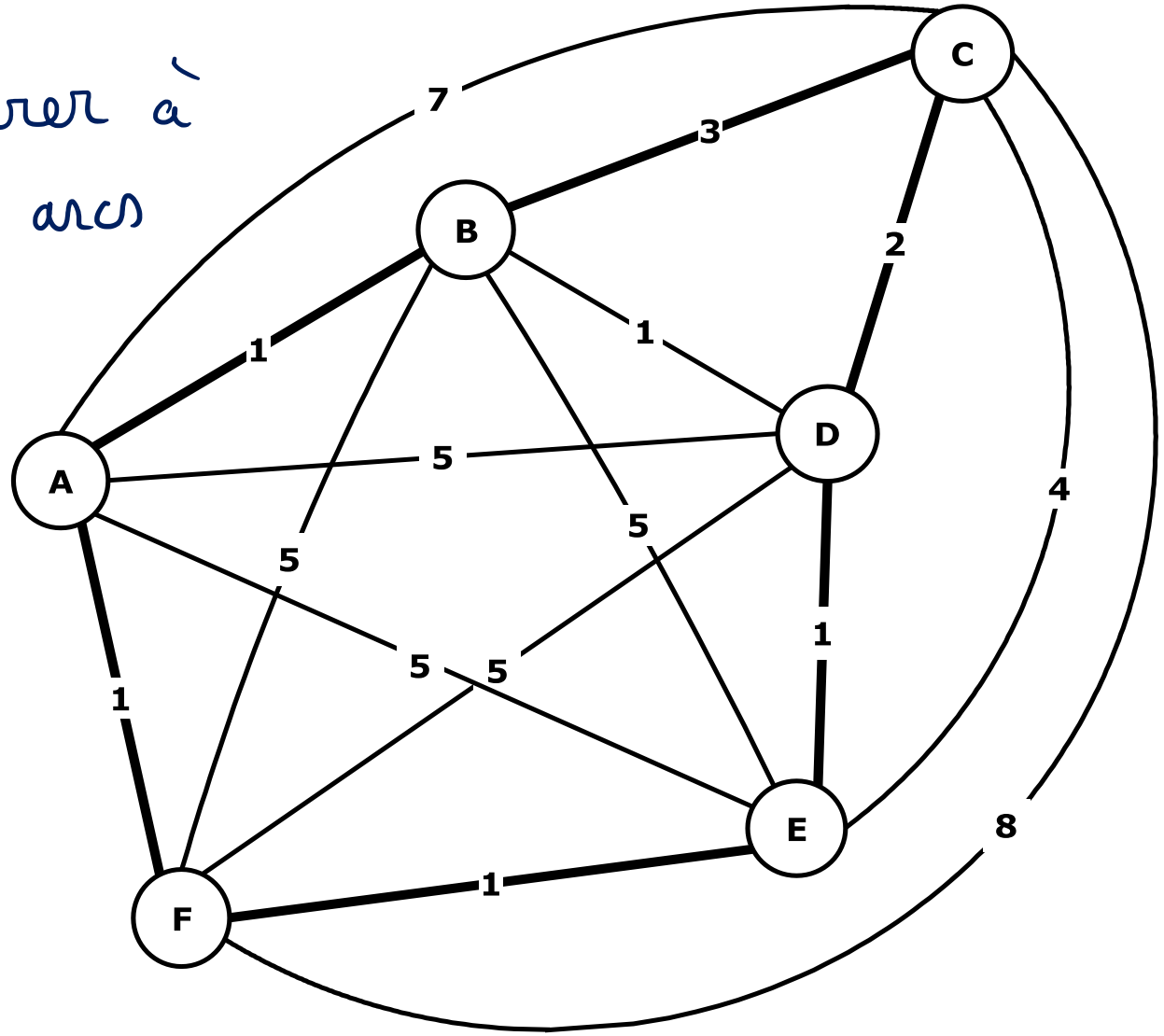
1. Créer une fonction **ppv(som,G,liste_exc)** qui pour un sommet **som**, un graphe **G** et une liste **liste_exc** renvoie le plus proche voisin de **som** dans G qui ne soit pas dans **liste_exc.** En cas de non existence d'un tel sommet, on peut par exemple renvoyer la valeur -1.



1. Créer une fonction **parcours_ppv(som,G)** qui renvoie la *distance* et *la liste des sommets parcourus* en partant de **som** et en appliquant la méthode du plus proche voisin dans **G.**

Meilleure insertion ?

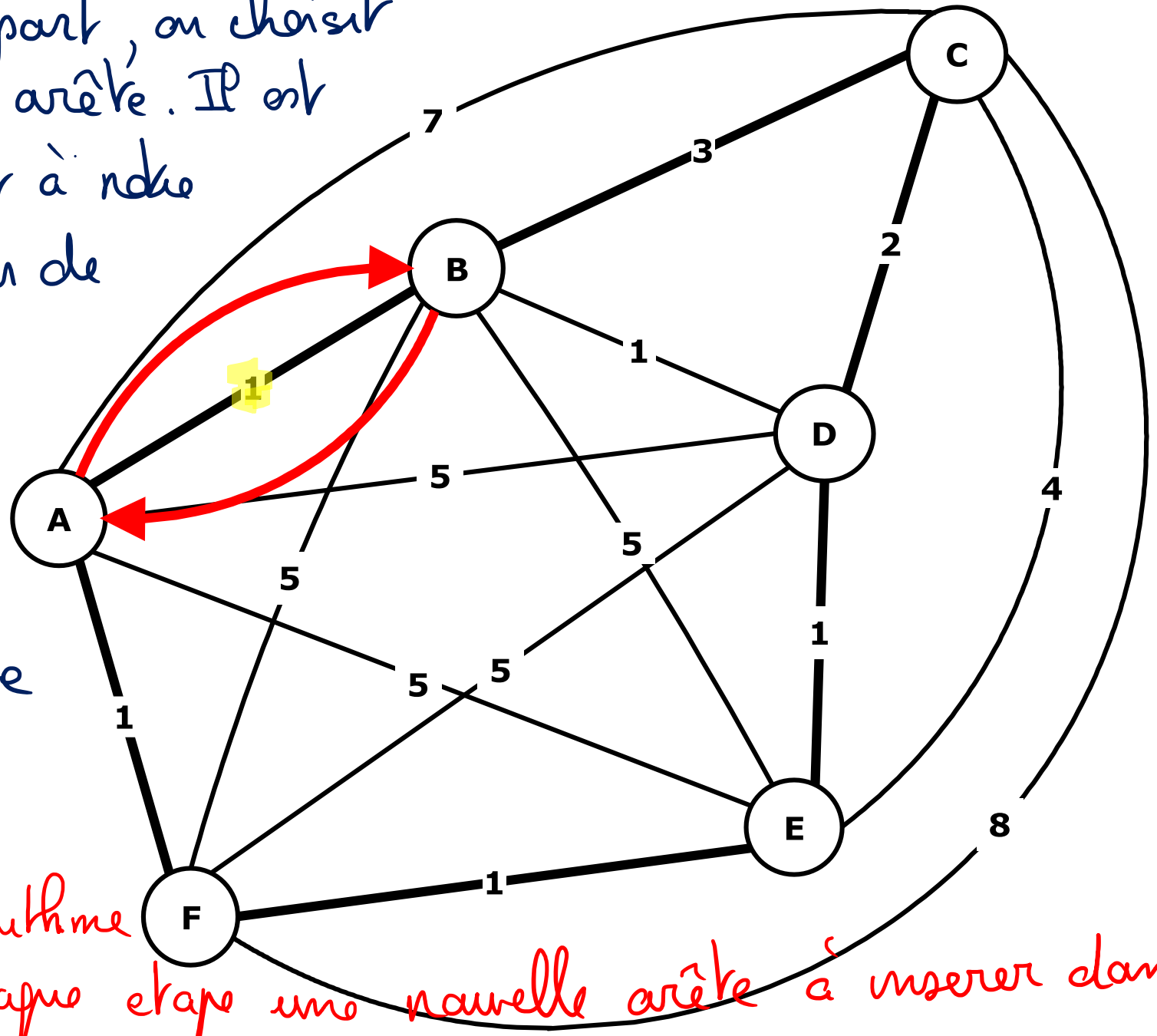
l'idée est d'insérer à chaque étape des arcs à un circuit ...



Pour le circuit de départ, on choisit pour commencer une arête. Il est pertinent, par rapport à notre problème, de la choisir de poids minimum.

Ici nous avons choisi $\{A, B\}$

Ce qui nous donne le circuit (A, B, A)

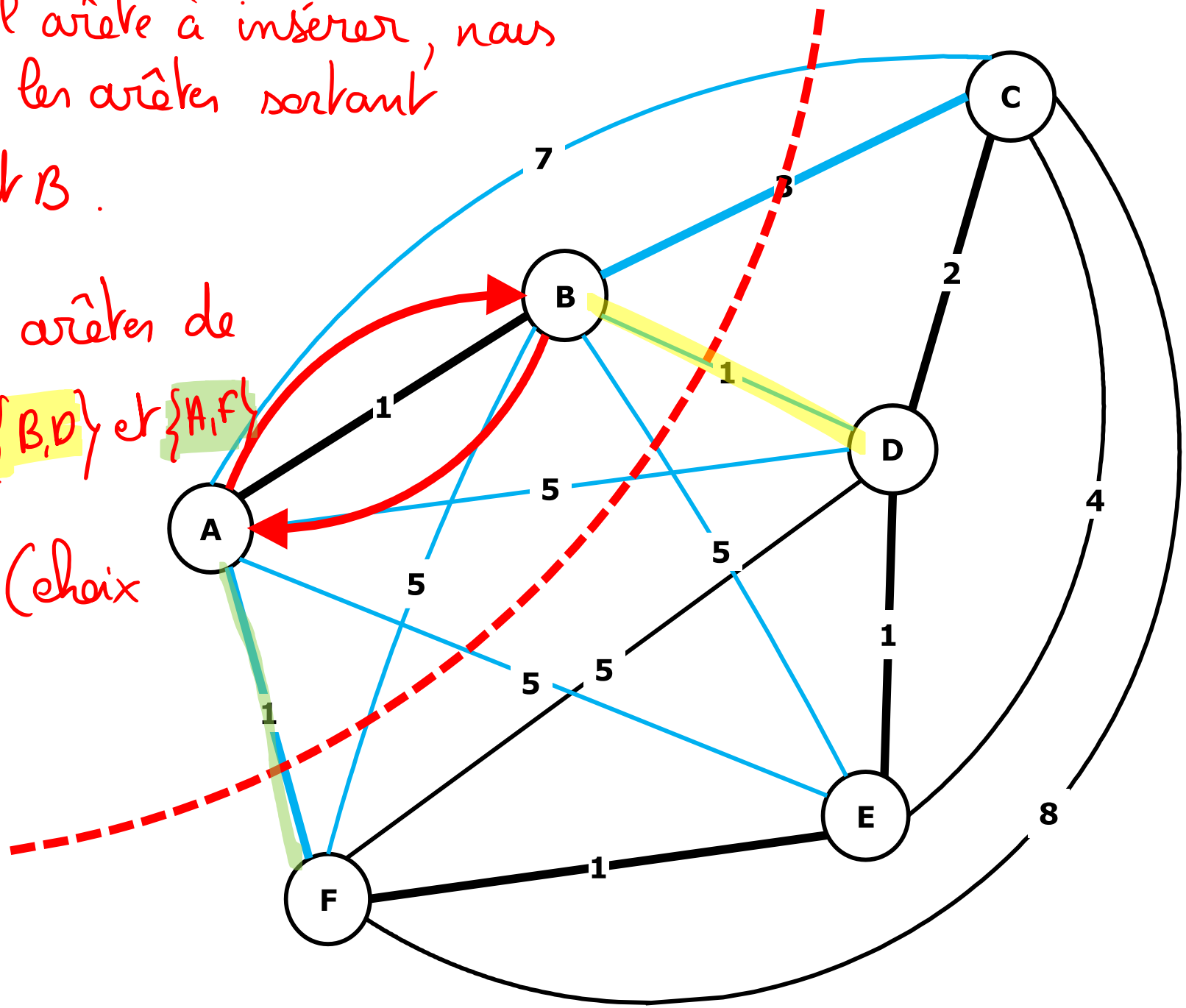


Le principe de l'algorithme est de choisir à chaque étape une nouvelle arête à insérer dans notre circuit.

Pour le choix de l'arête à insérer, nous regardons toutes les arêtes sortant des sommets A et B.

Il y a ici deux arêtes de poids minimum : {B,D} et {A,F}

On choisit {B,D} (choix arbitraire)



Nous avons choisi $\{B, D\}$

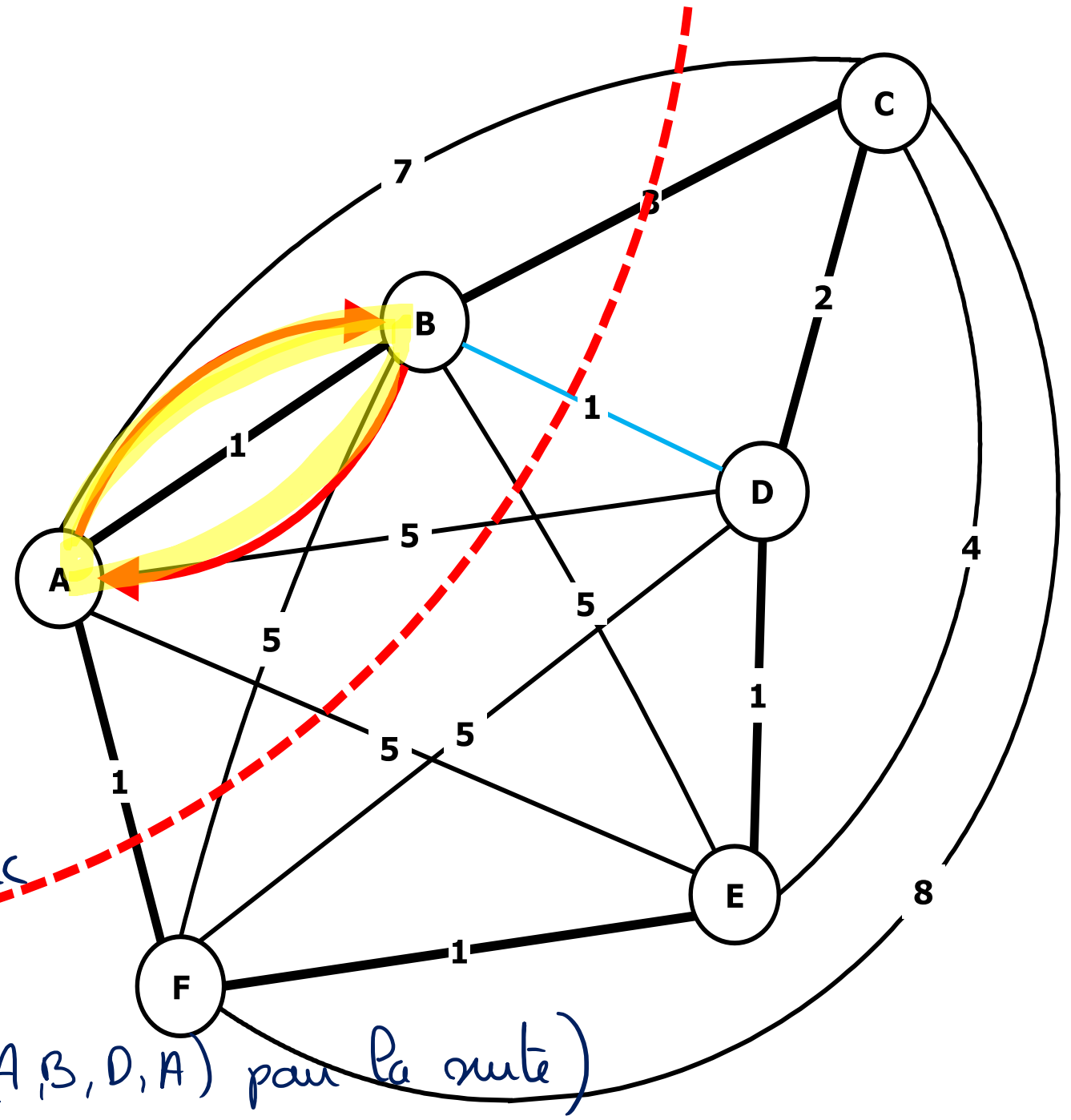
Nous pouvons donc insérer soit (B, D) soit (D, B) dans le circuit de départ (A, B, A)

Insertion de (B, D) : nous obtenons le circuit (A, B, D, A)

Insertion de (D, B) : nous obtenons le circuit (A, D, B, A)

Les deux circuits (A, B, D, A) et (A, D, B, A) ont le même poids donc PEU IMPORTE

(nous avons choisi (A, B, D, A) pour la suite)

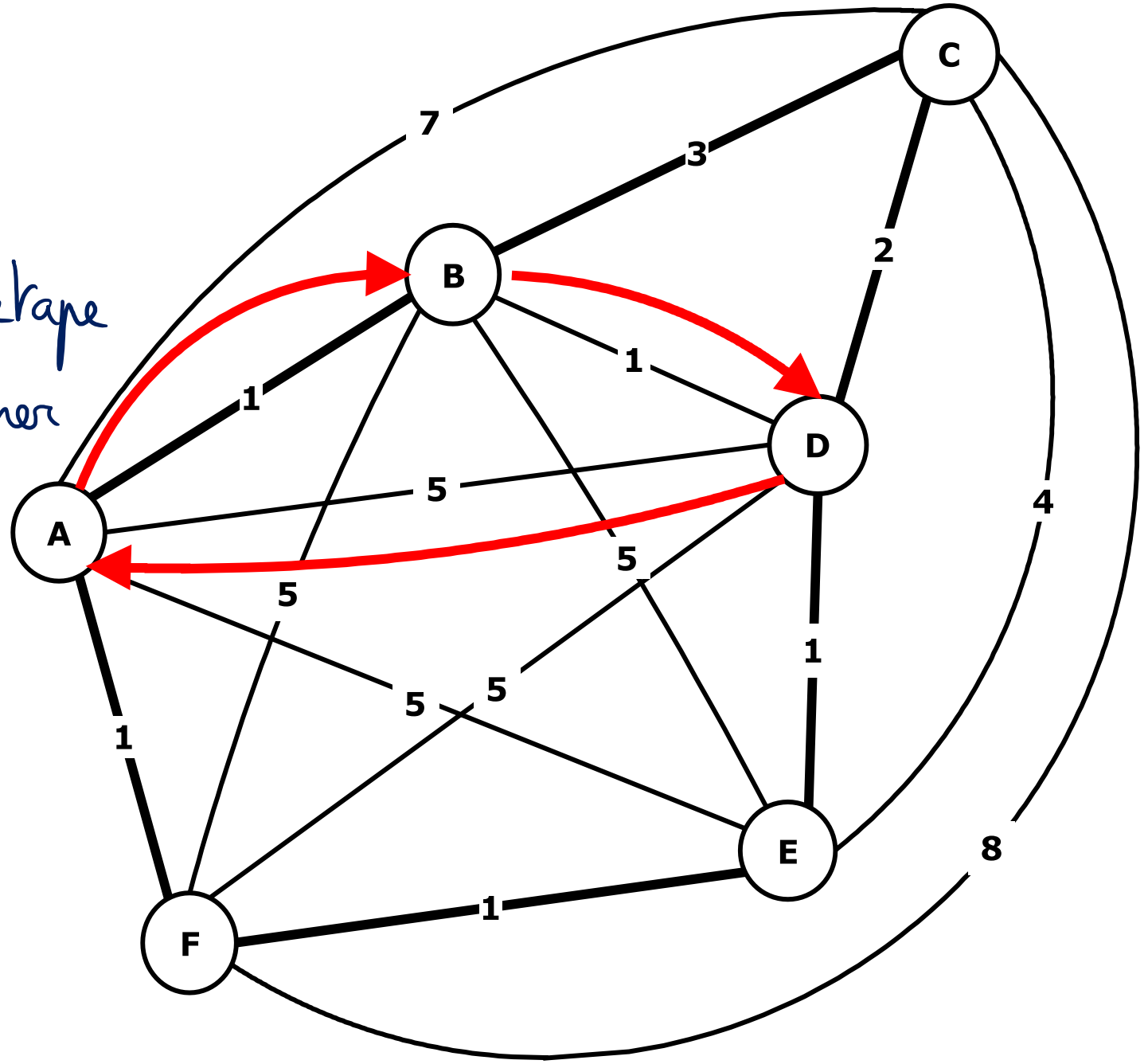


Circuit de départ

(A, B, D, A)

Nous procédons comme dans l'étape précédente, nous allons examiner les arêtes qui partent de l'ensemble des sommets déjà choisis dans le circuit :

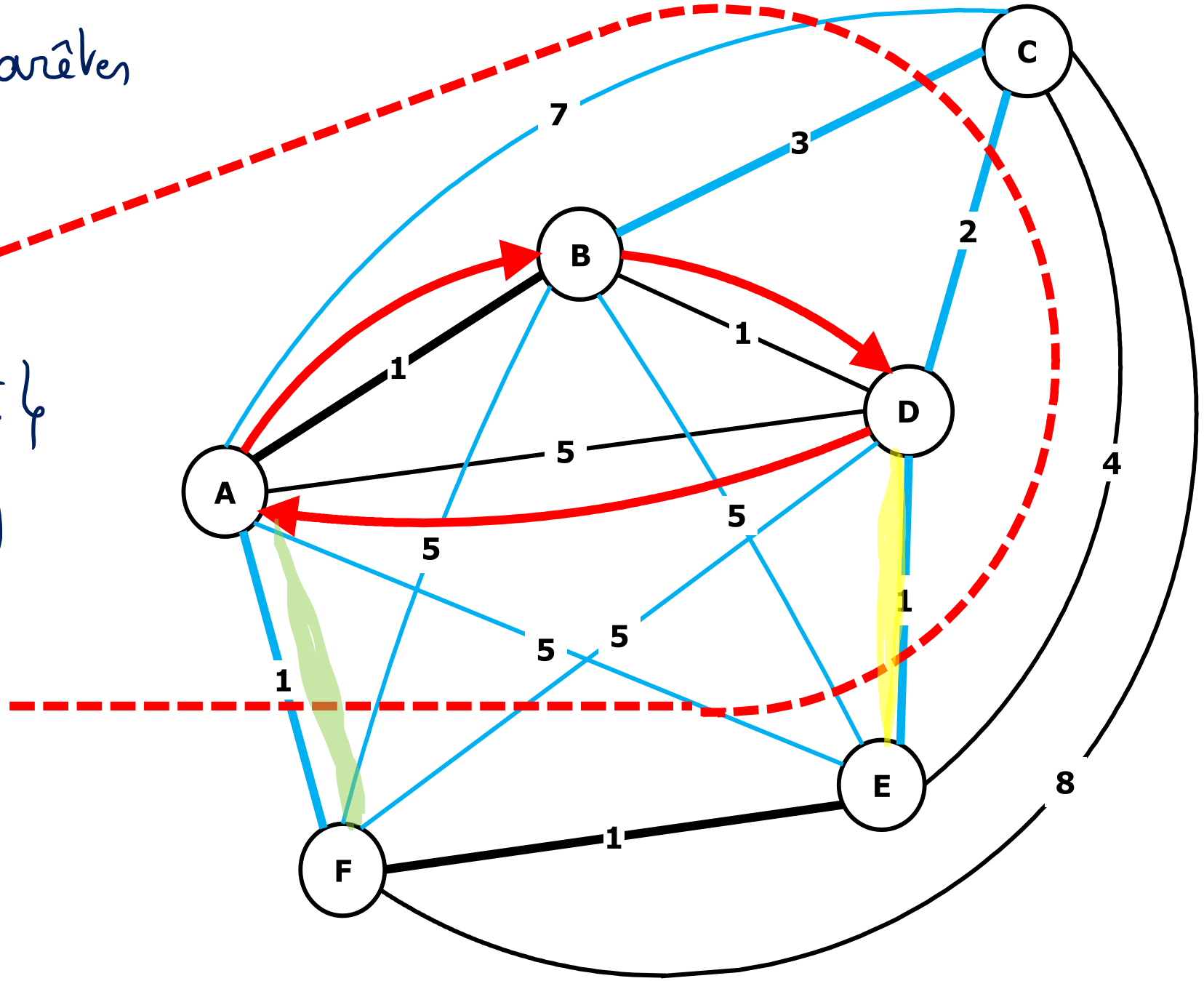
{A, B, D}



Ici nous avons deux arêtes de poids minimum :

$\{A, F\}$ et $\{D, E\}$

Nous choisissons $\{D, E\}$ (choix arbitraire)

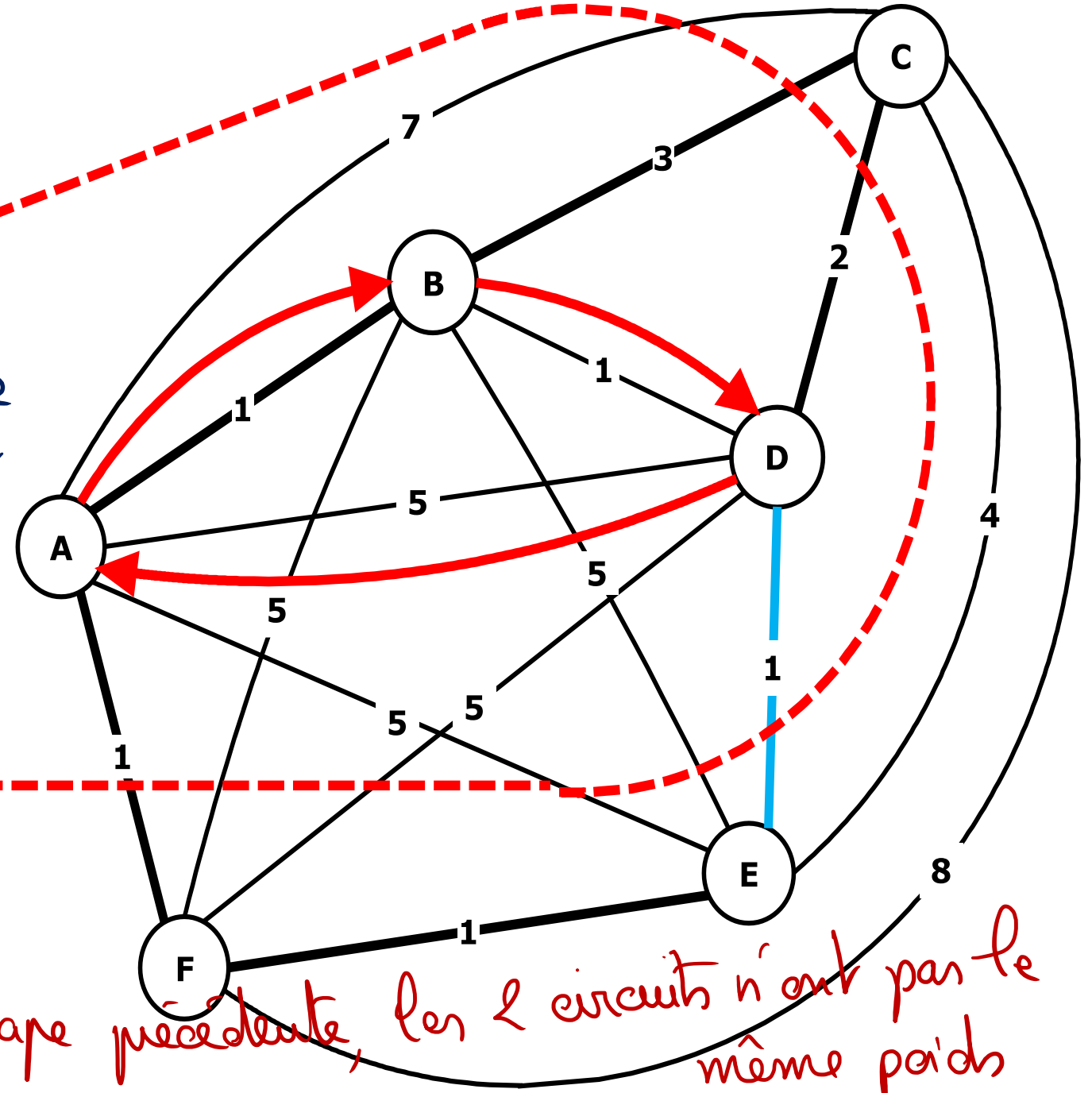


Nous avons choisi $\{D, E\}$.

Il y a deux façons de l'insérer dans le circuit (A, B, D, A) :

- Soit en la parcourant dans le sens (D, E) → ce qui donnerait le circuit (A, B, D, E, A)

- Soit en la parcourant dans le sens (E, D) → ce qui donnerait le circuit (A, B, E, D, A)

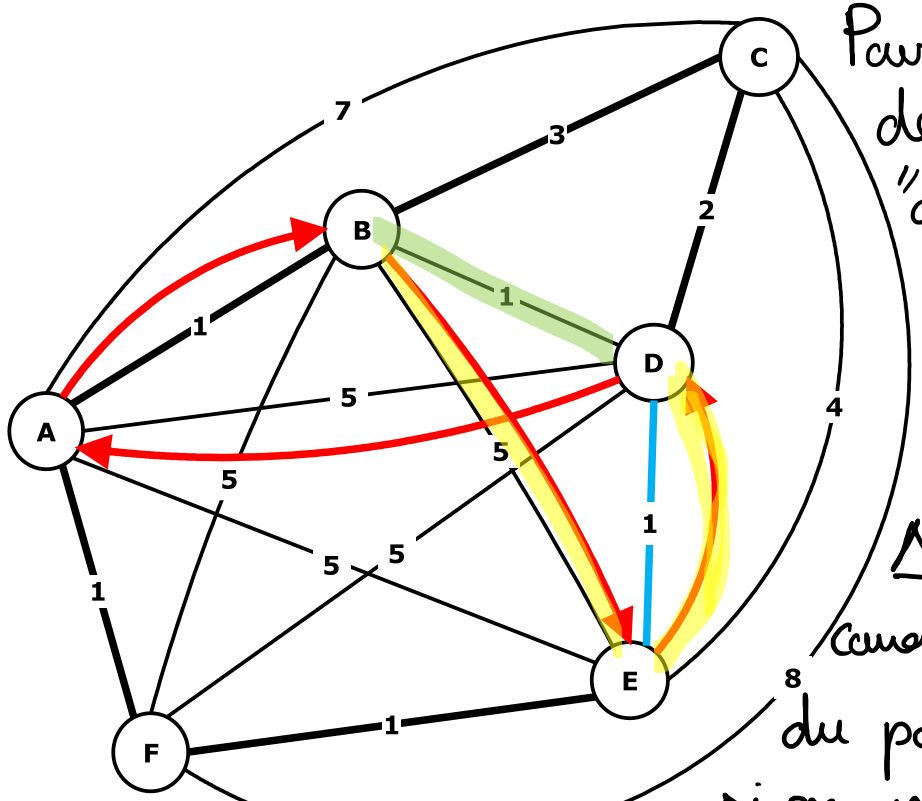


Cette fois-ci, contrairement à l'étape précédente, les 2 circuits n'ont pas le même poids

Circuit de départ (A, B, D, A) – Insertion de l'arête {E, D}

Insertion de (E, D) :

(A, B, D, A) ⟶ (A, B, E, D, A)



Si on insère (E, D) dans le circuit (A, B, D, A), l'arc (B, D) est remplacé par (B, E, D)

$\Delta(E,D) = BE + ED - BD = 5 + 1 - 1 = 5$

Pour choisir entre les deux on calcule les "delta" : $\Delta(E,D)$ et $\Delta(D,E)$

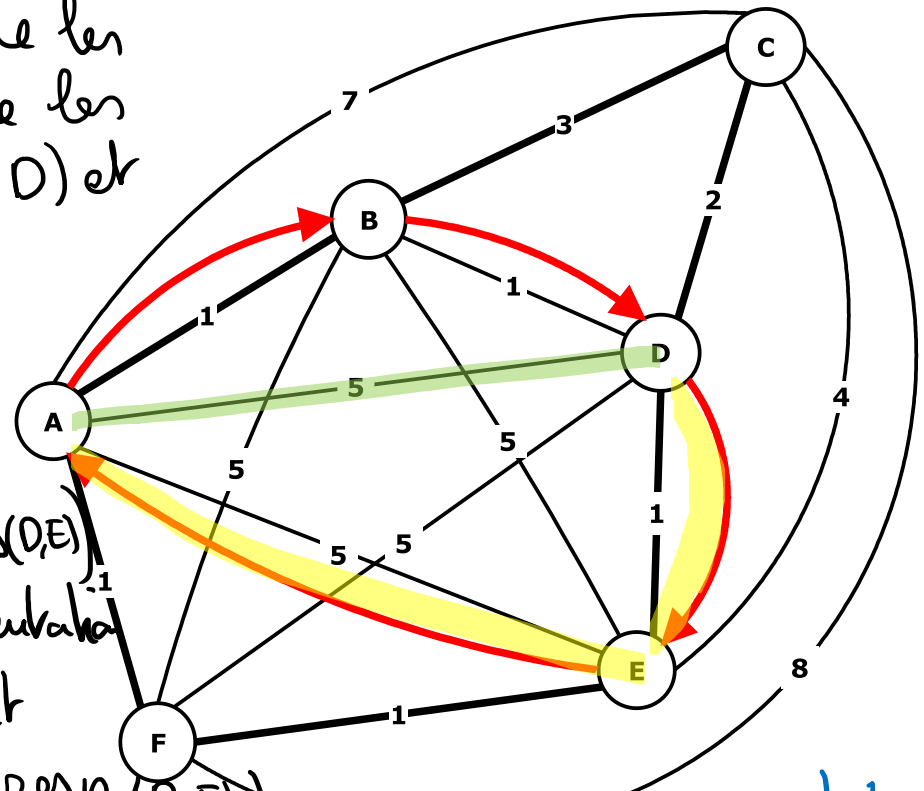
?

$\Delta(E,D)$ (resp. $\Delta(D,E)$) correspond à l'augmentation du poids du circuit si on insère (E, D) (resp (D, E))

$\Delta(E,D) > \Delta(D,E)$
On choisit donc d'insérer (D, E)

Insertion de (D, E) :

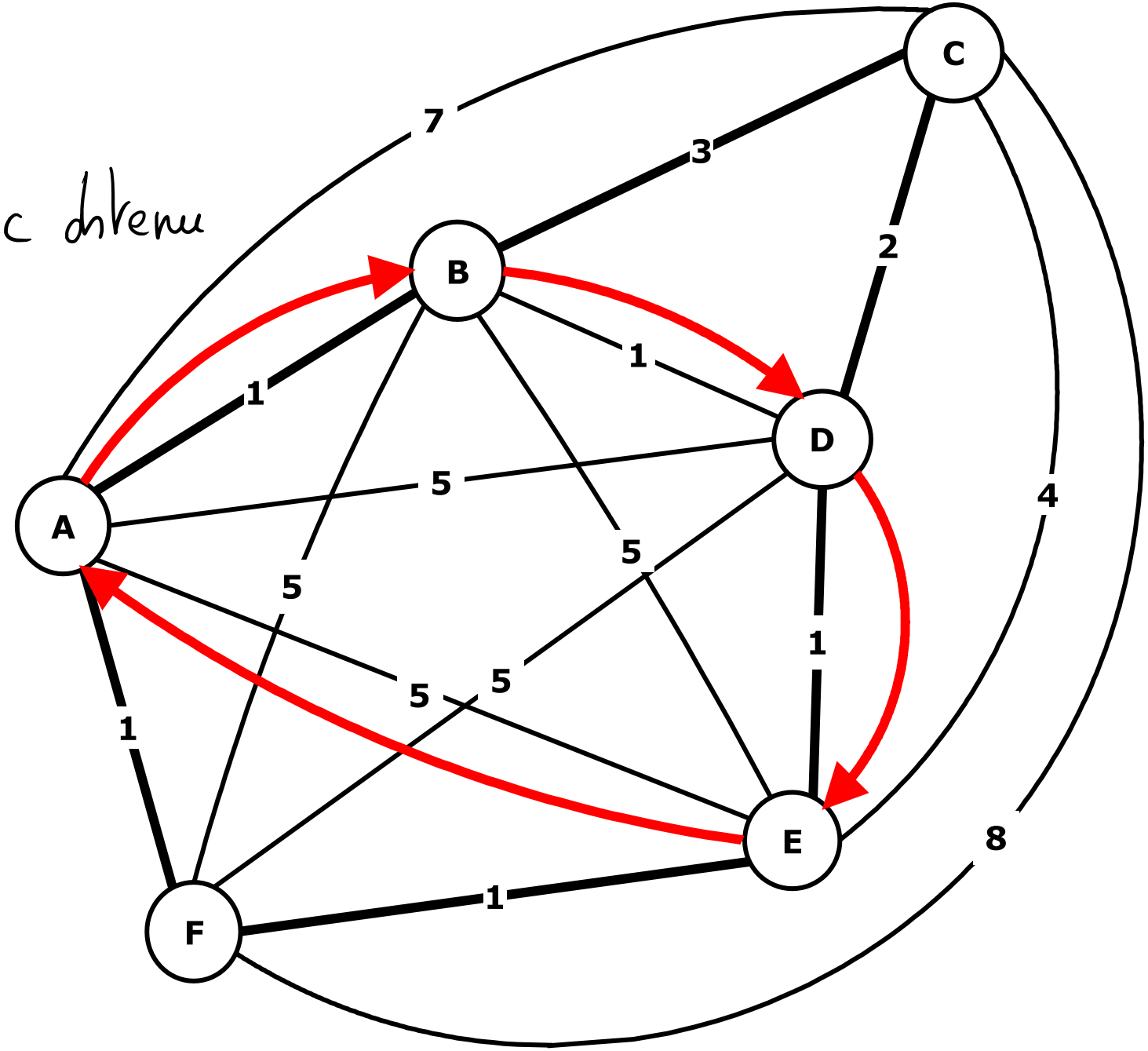
(A, B, D, A) ⟶ (A, B, D, E, A)



Si on insère (D, E) dans le circuit (A, B, D, A) l'arc (D, A) est remplacé par (D, E, A)

$\Delta(D,E) = DE + EA - DA = 1 + 5 - 5 = 1$

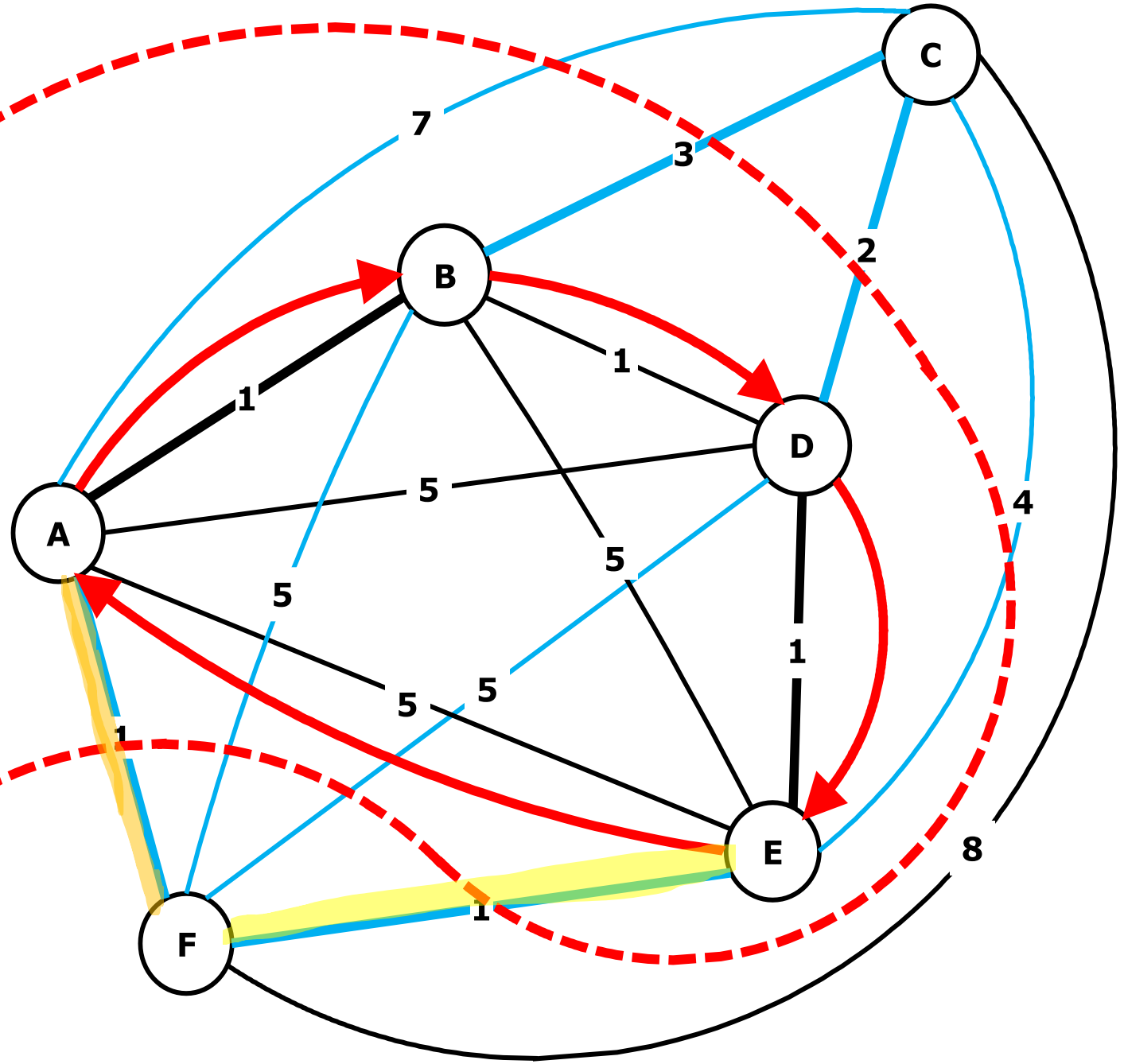
Nous avons donc obtenu le circuit (A, B, D, E, A)



Etape suivante : comme pour les précédentes, nous examinons les arêtes partant de l'ensemble des sommets du circuit obtenu : $\{A, B, D, E\}$

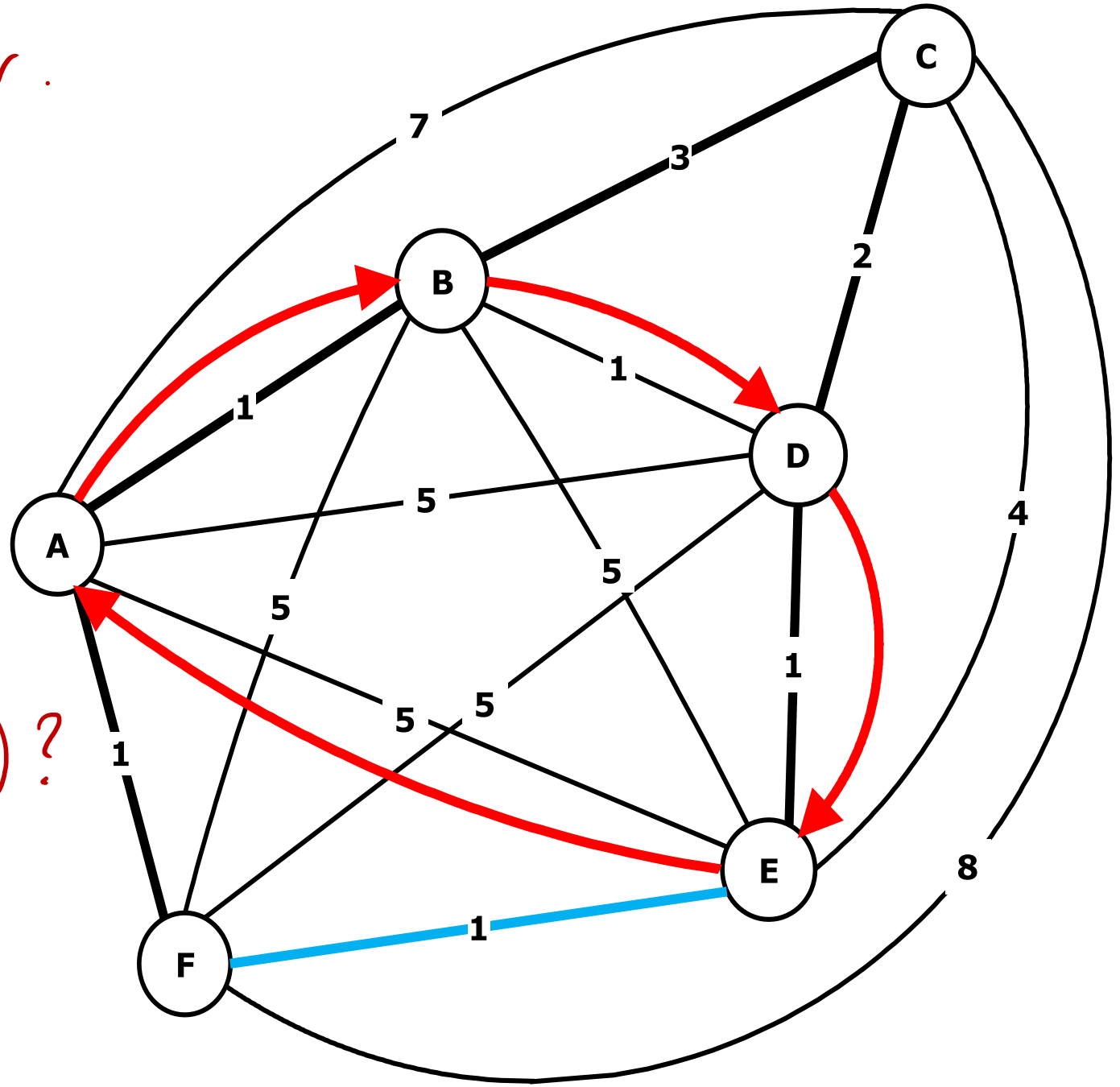
Il y a deux arêtes de poids minimal : $\{A, F\}$ et $\{E, F\}$

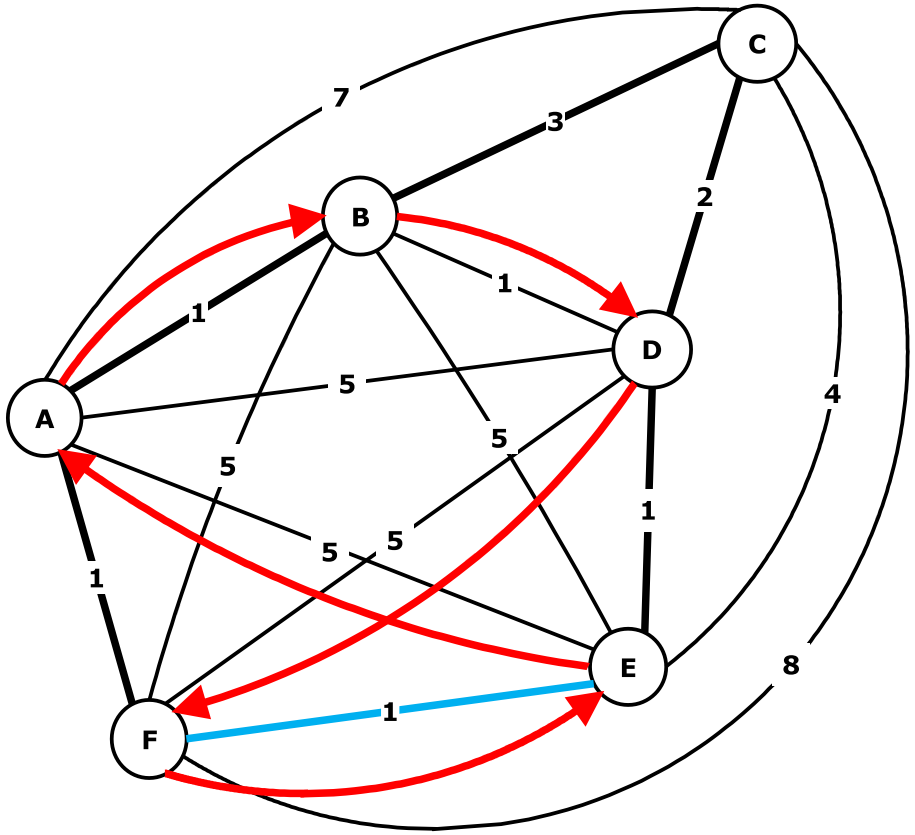
Nous choisissons $\{E, F\}$ (arbitrairement)



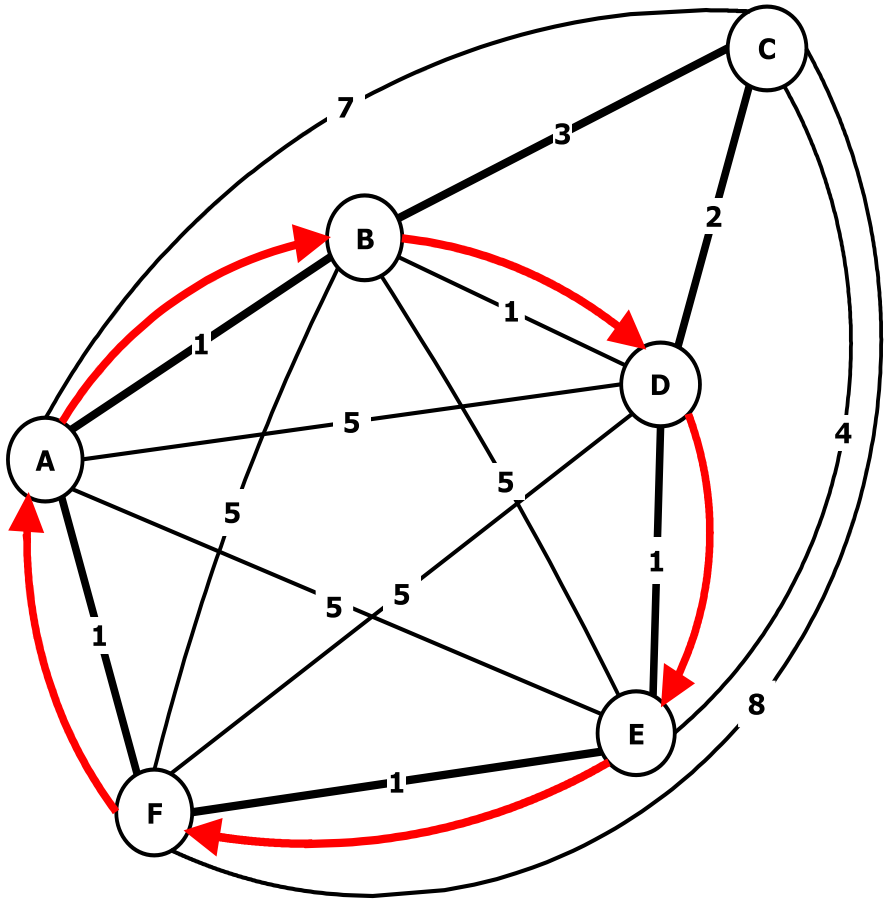
Même question que précédemment.
Comment insérer {E, F} ?
Dans le sens (E, F) ou dans le sens (F, E) ?

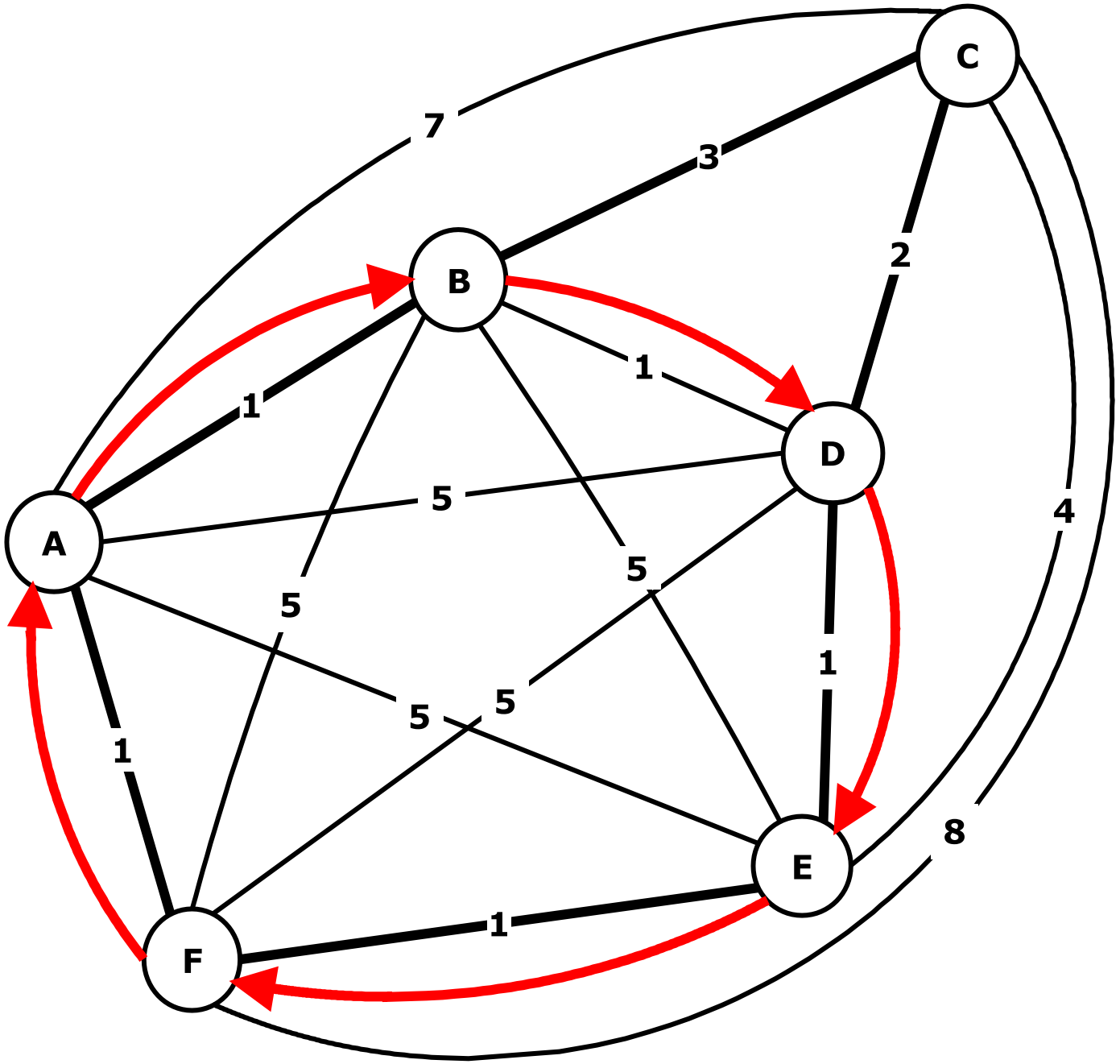
C'est à dire, pour obtenir le circuit (A, B, D, E, F, A)
ou le circuit (A, B, D, F, E, A) ?

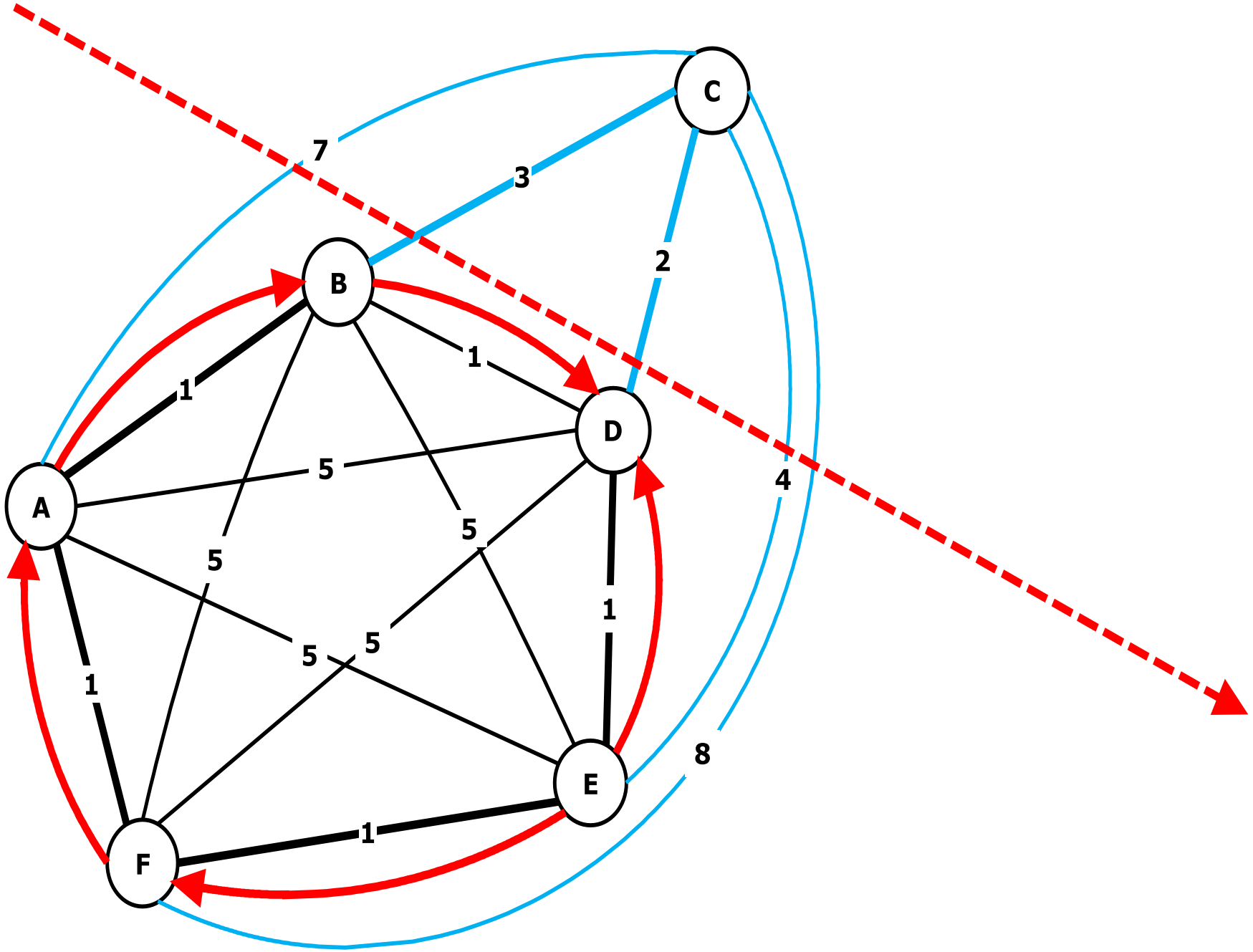


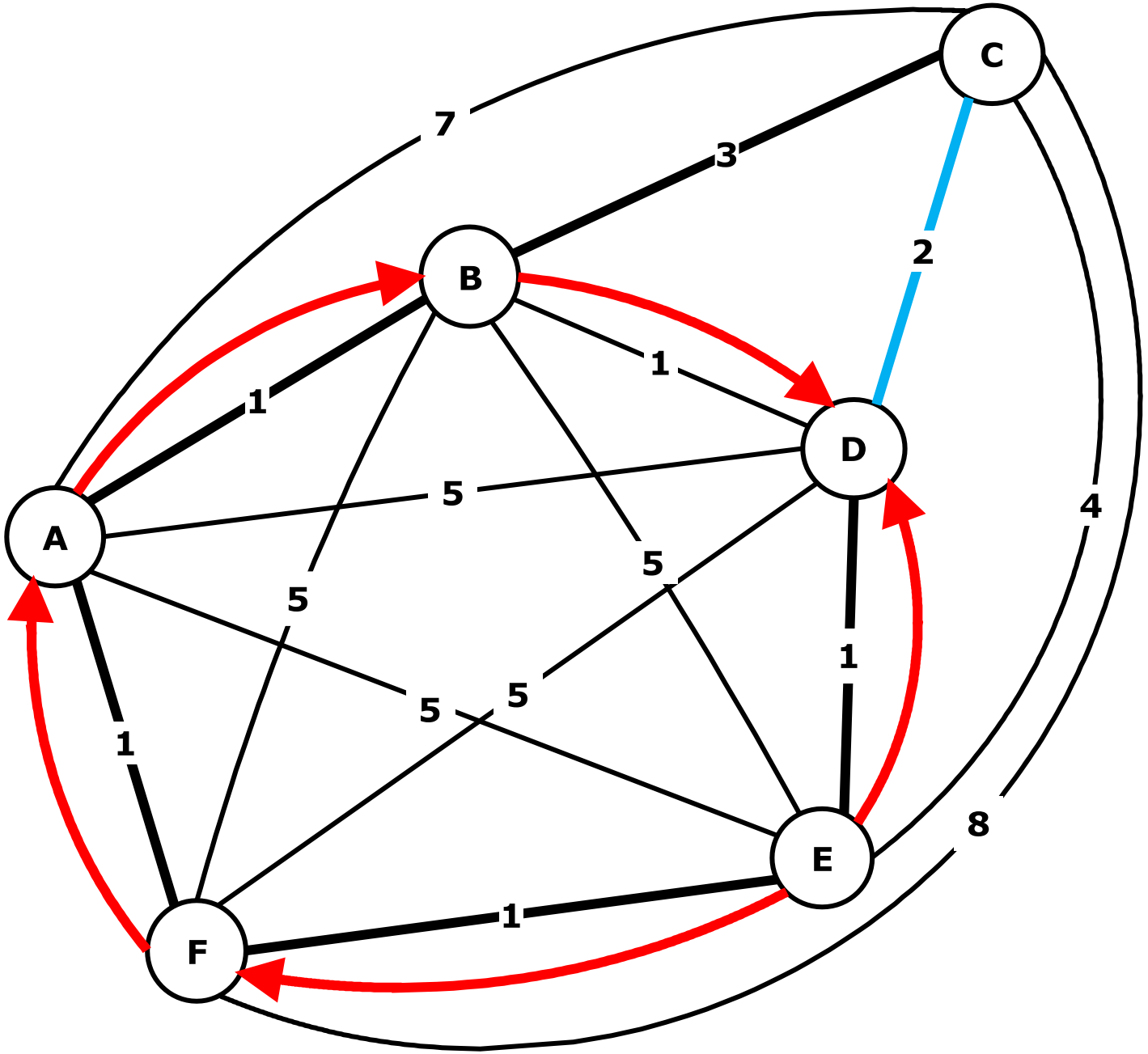

A
B
C
D
E
F
7
3
2
1
1
5
5
5
5
1
4
1
1
8

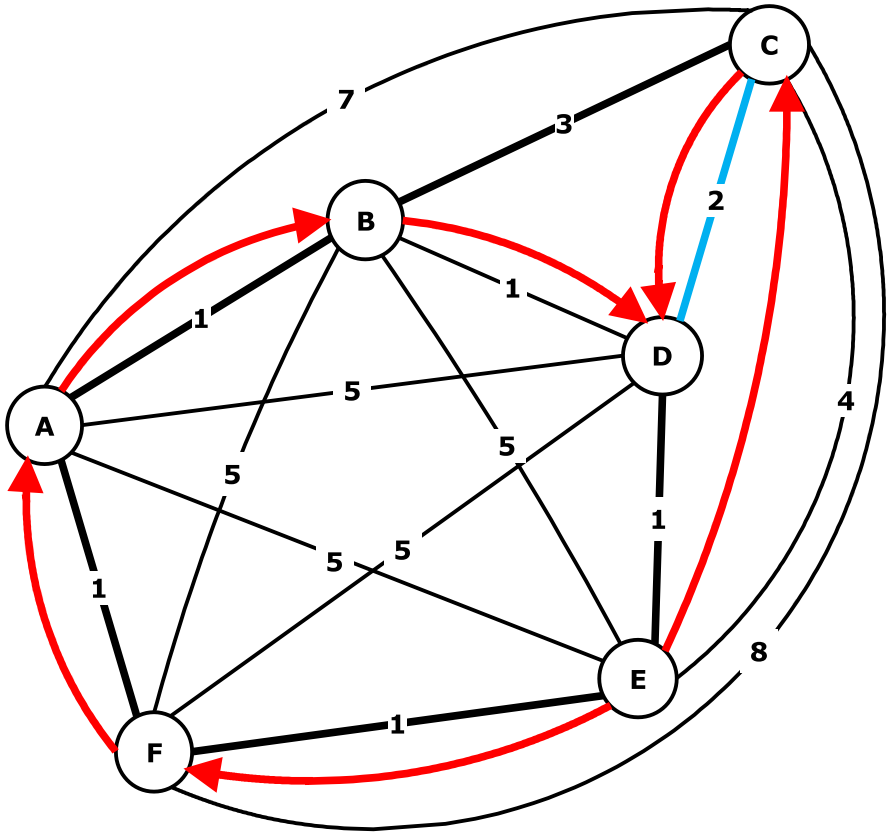
?


A
B
C
D
E
F
7
3
2
1
1
5
5
5
5
1
4
1
1
8

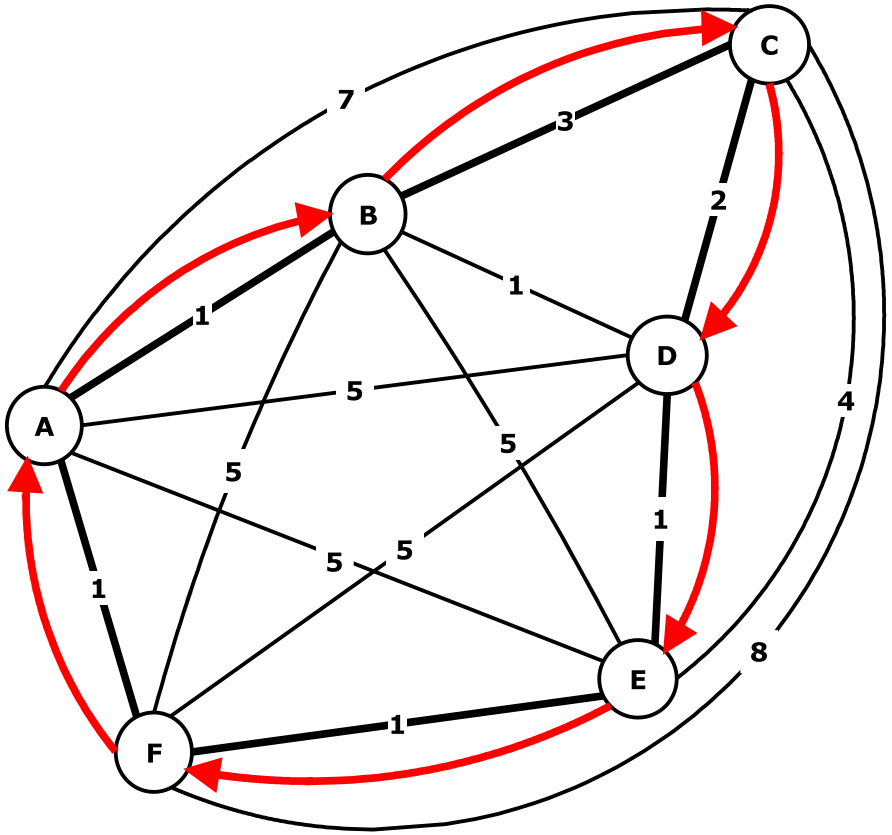

A
B
C
D
E
F
7
3
2
1
1
5
4
5
5
5
1
5
1
1
8

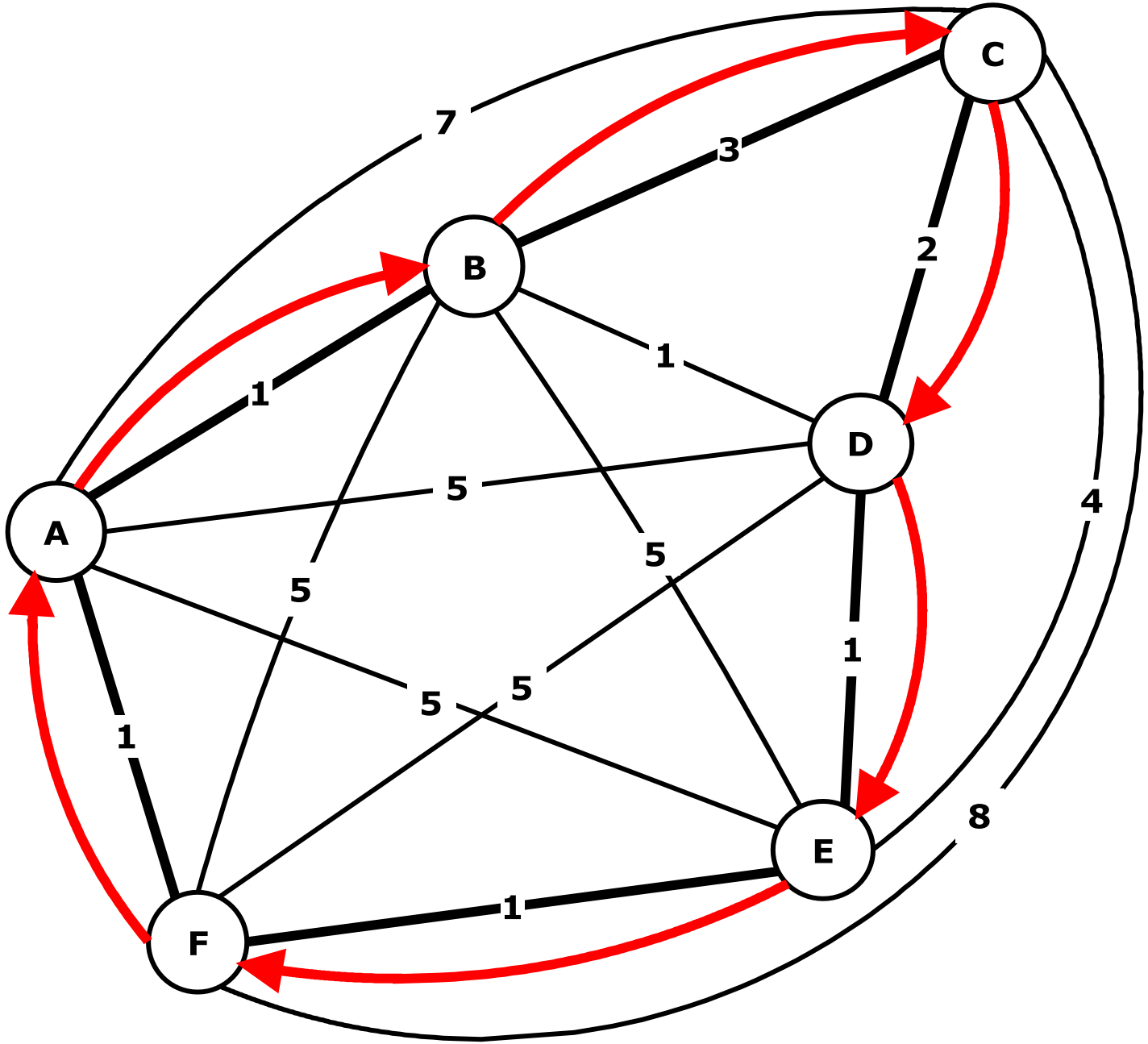

C
7
3
2
B
1
1
D
5
4
A
5
5
5
1
5
1
E
8
1
F


C
B
D
A
E
F
7
3
2
1
1
5
4
5
5
5
1
5
1
8
1


A
B
C
D
E
F
7
3
2
1
1
5
5
5
5
1
4
1
1
8

?


A
B
C
D
E
F
7
3
2
1
1
5
5
5
5
1
4
1
1
8


A
B
C
D
E
F
7
3
2
1
1
5
4
5
5
5
5
1
1
1
8

Heuristique de la meilleure insertion

Algorithme ?

Heuristique de la meilleure insertion