ALGORITHMES DE RECHERCHE DES CHEMINS DE POIDS MINIMAUX À ORIGINE UNIQUE

Description du problème

- On se place dans le cas d'un graphe orienté dans lequel à chaque arc xy on associe une valeur V(xy).
- A partir d'un sommet de départ d nous allons rechercher les chemins de poids minimaux vers chacun des sommets du graphe.

- Application traditionnelle : Route des vacances, mais pas que.

Spécifications

- Données :
  - G = (X, U, V) un graphe pondéré
  - s un sommet de G
- Question :
  - Pour chaque sommet y de G quel est le chemin de poids minimal permettant d'aller de s à y ?

Poids d'un chemin d'un circuit

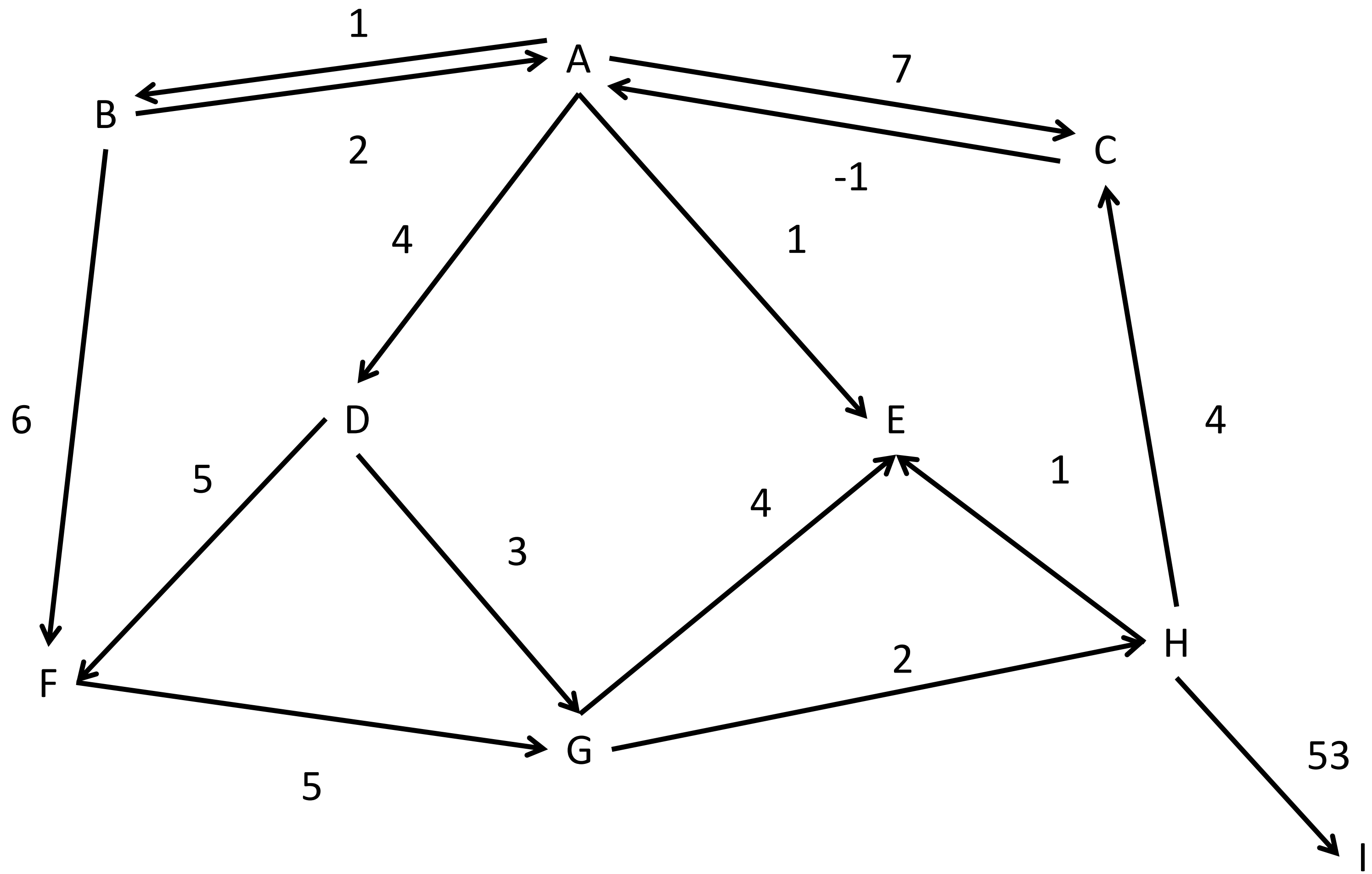
- Soit $\mu = x_1, x_2, x_3, \ldots, x_{k-1}, x_k$ un chemin du graphe pondéré $G = (X, U, V)$ alors on définit le poids du chemin μ comme :

$$v(\mu) = v(x_1x_2) + v(x_2x_3) + \ldots + v(x_{k-1}x_k)$$

- Soit $\mu = x_1, x_2, x_3, \ldots, x_{k-1}, x_k, x_1$ un circuit du graphe pondéré $G = (X, U, V)$ alors on définit le poids du circuit μ comme :

$$v(\mu) = v(x_1x_2) + v(x_2x_3) + \ldots + v(x_{k-1}x_k) + v(x_kx_1)$$

Exemple



Exemple : Représentation

	A	B	C	D	E	F	G	H	I
A	0	1	7	4	1	∞	∞	∞	∞
B	2	0	∞	∞	∞	6	∞	∞	∞
C	1	∞	0	∞	∞	∞	∞	∞	∞
D	∞	∞	∞	0	∞	5	3	∞	∞
E	∞	∞	∞	∞	0	∞	∞	∞	∞
F	∞	∞	∞	∞	∞	0	5	∞	∞
G	∞	∞	∞	∞	4	∞	0	2	∞
H	∞	∞	∞	∞	1	∞	∞	0	53
I	∞	∞	∞	∞	∞	∞	∞	∞	0

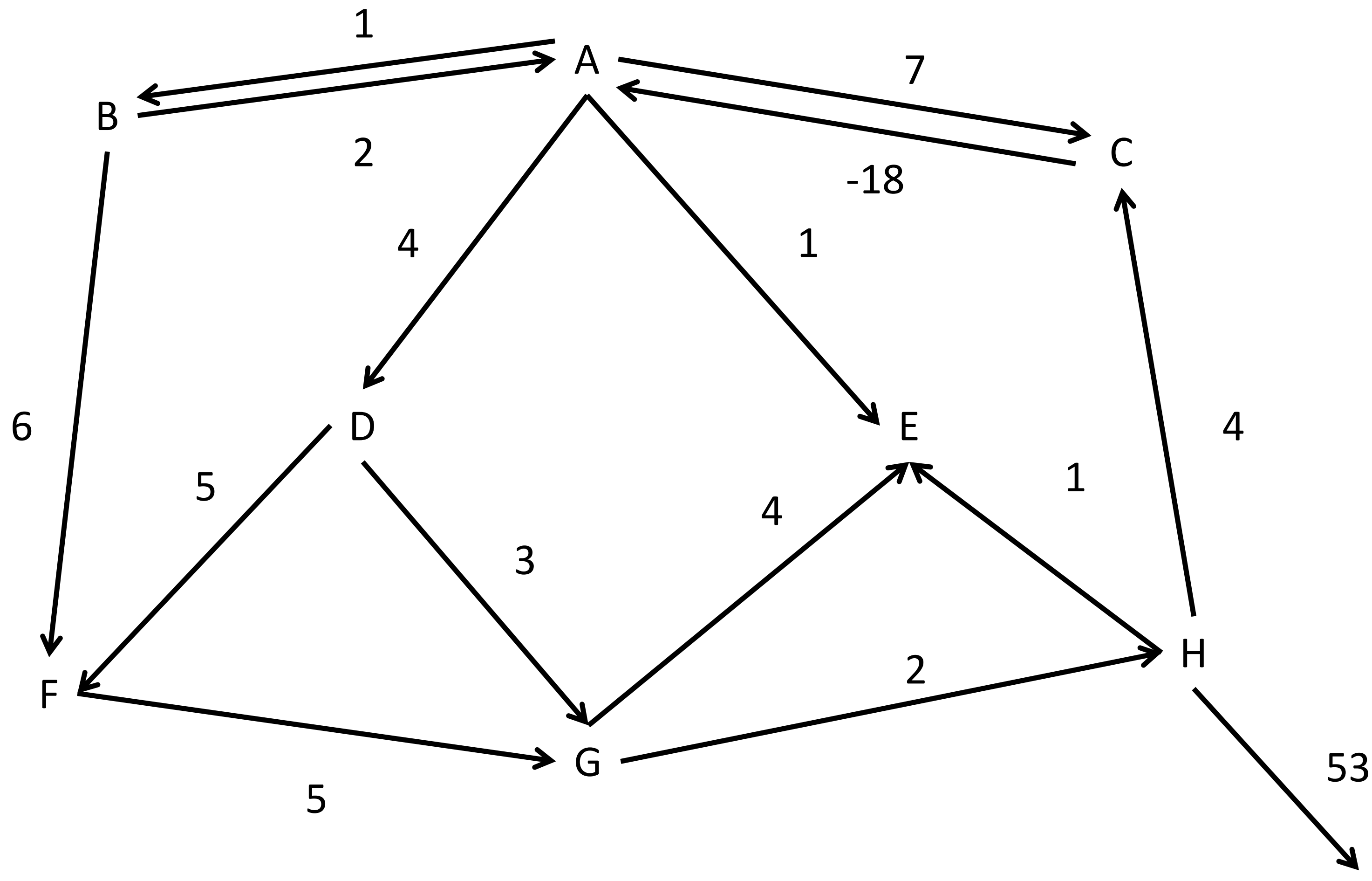
0 est l'élément neutre de l'adition.

Préconditions

- Il faut que l'ensemble des descendants du sommet de départ s soit égal à X.

- Le graphe G ne doit pas contenir de circuits de poids négatif.

ADGHCA circuit de poids négatif



ADGHCA circuit de poids négatif

- AB est un chemin de A vers B de poids 1
- ADGHCAB est un chemin de A vers B de poids -4
- ADGHCADGHCAB est un chemin de A vers B de poids -9
- ADGHCADGHCADGHCAB est un chemin de A vers B de poids -14 etc.

Propriété d'hérédité

Soit $\mu = x_1, x_2, x_3, \ldots, x_{k-1}, x_k$ un chemin de poids minimal de x_1 à x_k du graphe pondéré $G = (X, U, V)$ alors : $\mu' = x_1, x_2, x_3, \ldots, x_{k-1}$ est un chemin de poids minimal de x_1 à x_{k-1}.

Preuve

- Supposons le contraire : $\mu' = x_1, x_2, x_3, \dots, x_{k-1}$ n'est pas un chemin de poids minimal de x_1 à x_{k-1}.
- Il existe un chemin $\mu'' = x_1, y_2, y_3, \dots, y_{t-1}, x_{k-1}$ tel que $v(\mu'') < v(\mu')$.
- Soit $\mu''' = x_1, y_2, y_3, \dots, y_{t-1}, x_{k-1}, x_k$ alors
  - μ''' est un chemin de G
  - $v(\mu''') = v(\mu'') + v(x_{k-1}, x_k) < v(\mu') + v(x_{k-1}, x_k) = v(\mu)$

Propriété

- S’il n’existe pas de circuits de poids négatif alors pour tout couple de sommet (a, b) il existe toujours un chemin élémentaire de poids minimal.

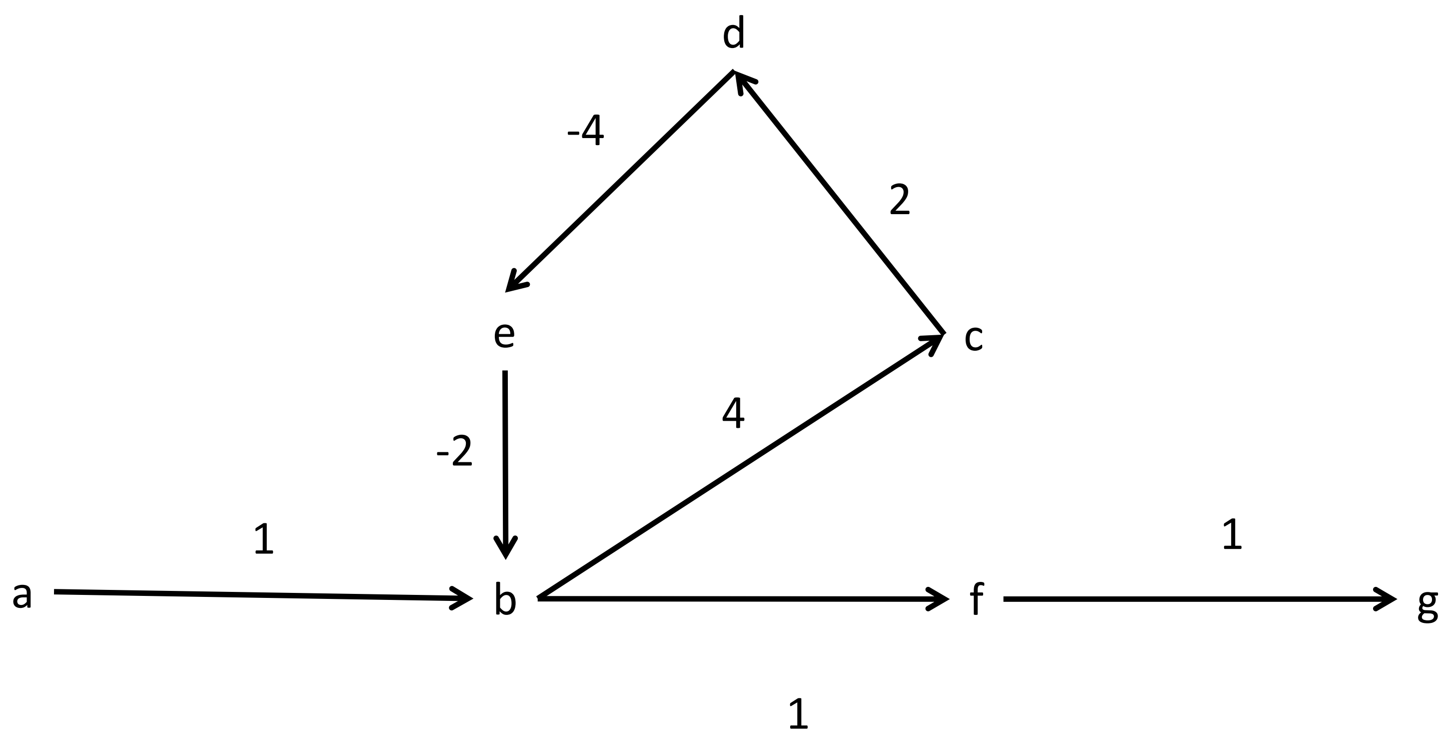
Preuve

- Soit $\mu = x_1, x_2, x_3, \ldots, x_{k-1}, x_k$ un chemin de poids minimal de x_1 à x_k du graphe G. Deux cas doivent être considéré :
  - μ est un chemin élémentaire de G : CQFD
  - Cas 2 sur seconde page

Preuve (suite)

– μ n'est pas un chemin élémentaire de G. Donc μ contient au moins un circuit. Il existe deux indices i et j tels que $0<i<j<k+1$ tels que $x_i = x_j$, il en résulte que $\mu'=x_i, x_{i+1}, \ldots, x_{j-1}, x_j$ forme un circuit du graphe G. Or $v(\mu') \geq 0$. La suite de sommets obtenu en retirant le cycle, $\mu''= x_1, x_2, x_3, \ldots, x_i, x_{j+1}, \ldots, x_{k-1}, x_k$ est un chemin du graphe G de plus nous pouvons affirmer que $v(\mu) = v(\mu'') + v(\mu')$. Donc $v(\mu') \geq 0$ implique $v(\mu) \geq v(\mu'')$. CQFD

Illustration



Illustration

- abcdebcdebcdeb…cdeb…cdebfg est un chemin de a à f de poids minimal
- abfg un chemin élémentaire de a à f de poids minimal de même poids que le précédent

Conséquence

- Dans un graphe sans circuits de poids négatif, les chemins de poids minimaux permettent de construire un arbre couvrant.

- C’est la conséquence des deux propositions précédentes.

Algorithme de Dijkstra (hypothèse)

- En plus des hypothèses déjà formulées
  - Pas de circuits de poids négatifs
  - L'ensemble des descendants de s est X
- On suppose que tous les poids sont positifs

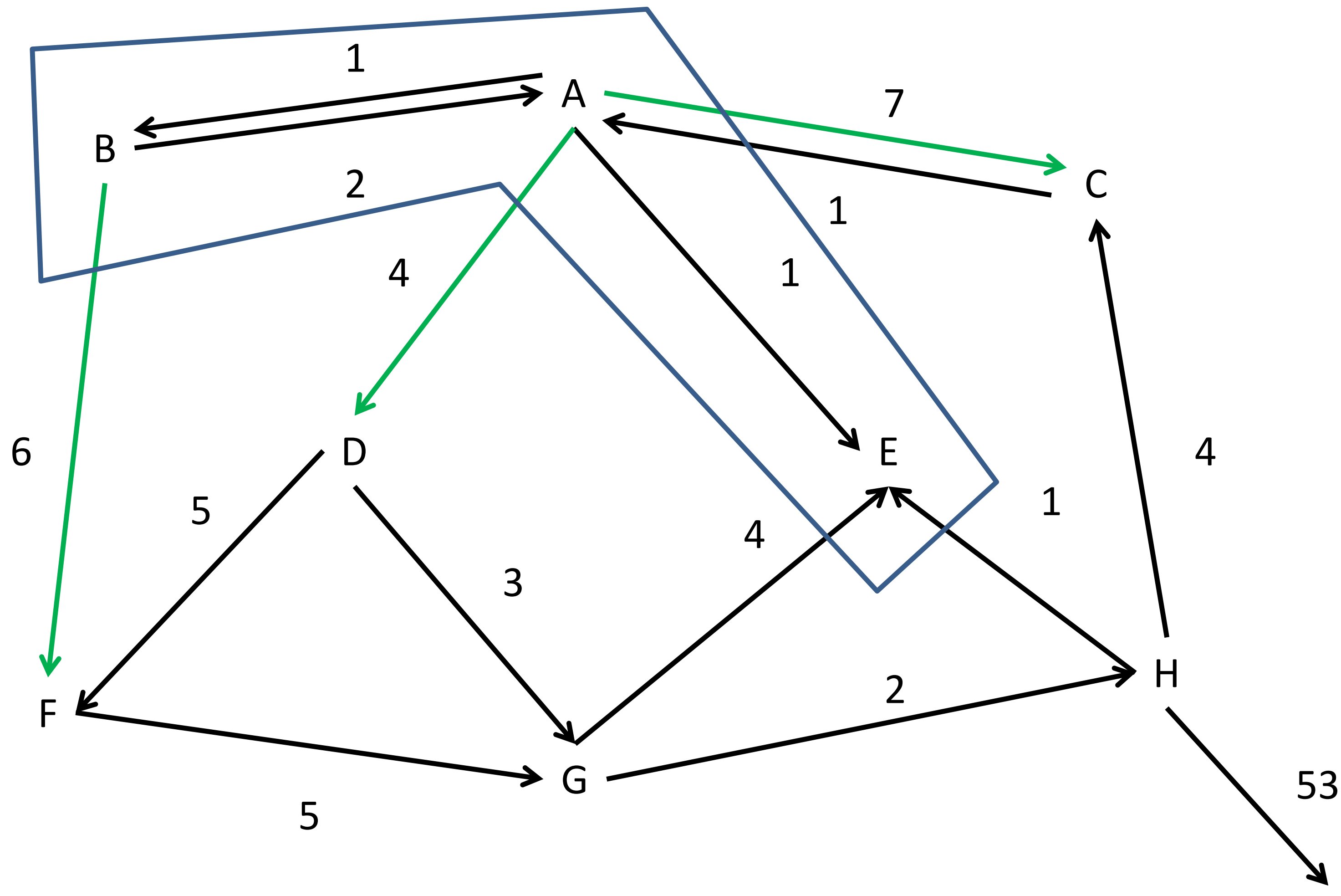
Dijkstra (idée de l'algorithme)

- Supposons que nous ayons un nombre positif NbPos et deux ensembles (fermé et cocycle) tels que :
  - Si un sommet x est dans fermé alors le chemin de poids minimal entre s et x est de poids inférieur ou égal à NbPos
  - Si un sommet x n'est pas dans fermé alors le chemin de poids minimal entre s et x est de poids supérieur ou égal à NbPos

Dijkstra (idée de l’algorithme)

- Cocycle est l’ensemble des arcs ayant leur origine dans fermé et leur extrémité à l’extérieur de fermé
- Enfin pour chaque sommet x dans fermé D[x] donne le poids du chemin de poids minimal calculé par notre algorithme entre s et x.

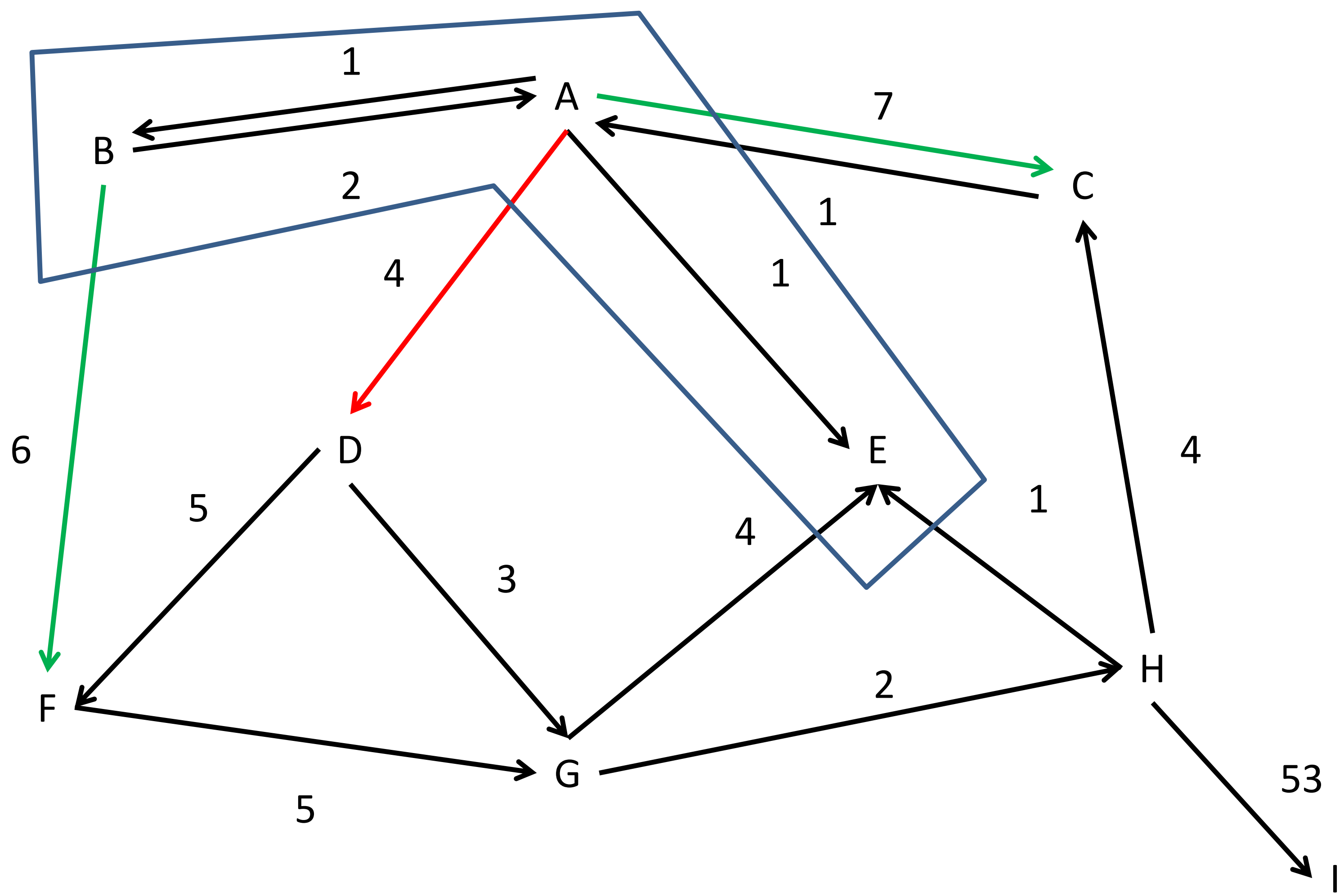
A	B	C	D	E	F	G	H	I
0	1	∞	∞	1	∞	∞	∞	∞



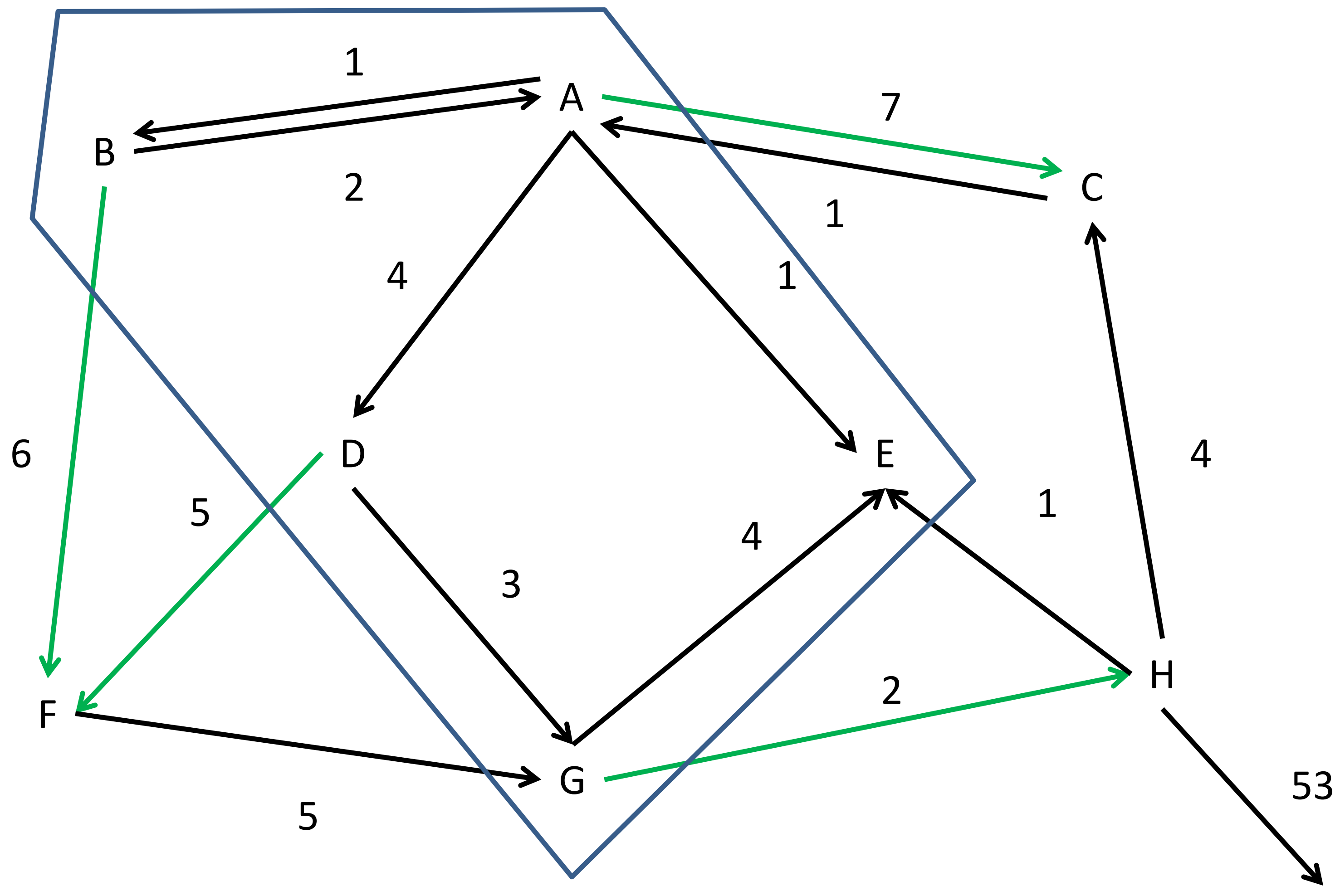
Idée d'une étape de l'algorithme

- Choisir dans cocycle l'arc zt tel que la valeur D[z] + v(zt) soit minimale
- Mettre t dans fermé
- D[t] ← D[z] + v(zt)
- Ajuster l'ensemble cocycle

A	B	C	D	E	F	G	H	I
0	1			1				



A	B	C	D	E	F	G	H	I
0	1		4	1		7		



Valeurs initiales

- Pour tout x <> s D[x] = ∞ sauf D[s] = 0
- fermé = {s}
- cocycle = {(y,sy,val) / y ϵ Succ(s) et val = v(sy)}

Condition finale

- cocycle est vide

Algo Dijkstra (entête)

- Algo CPMDijkstra
- Données :
  - G = (X,U,V) un graphe pondéré
  - s un sommet de G
- Résultats :
  - GCPM = (X, U’, V) un graphe pondéré
  - D : tableau de nombre indicé par les sommets
- Variables :
  - fermé ensemble de sommets
  - cocycle ensemble de triplets (sommet, arc, valeur)

Algo Dijkstra (code 1)

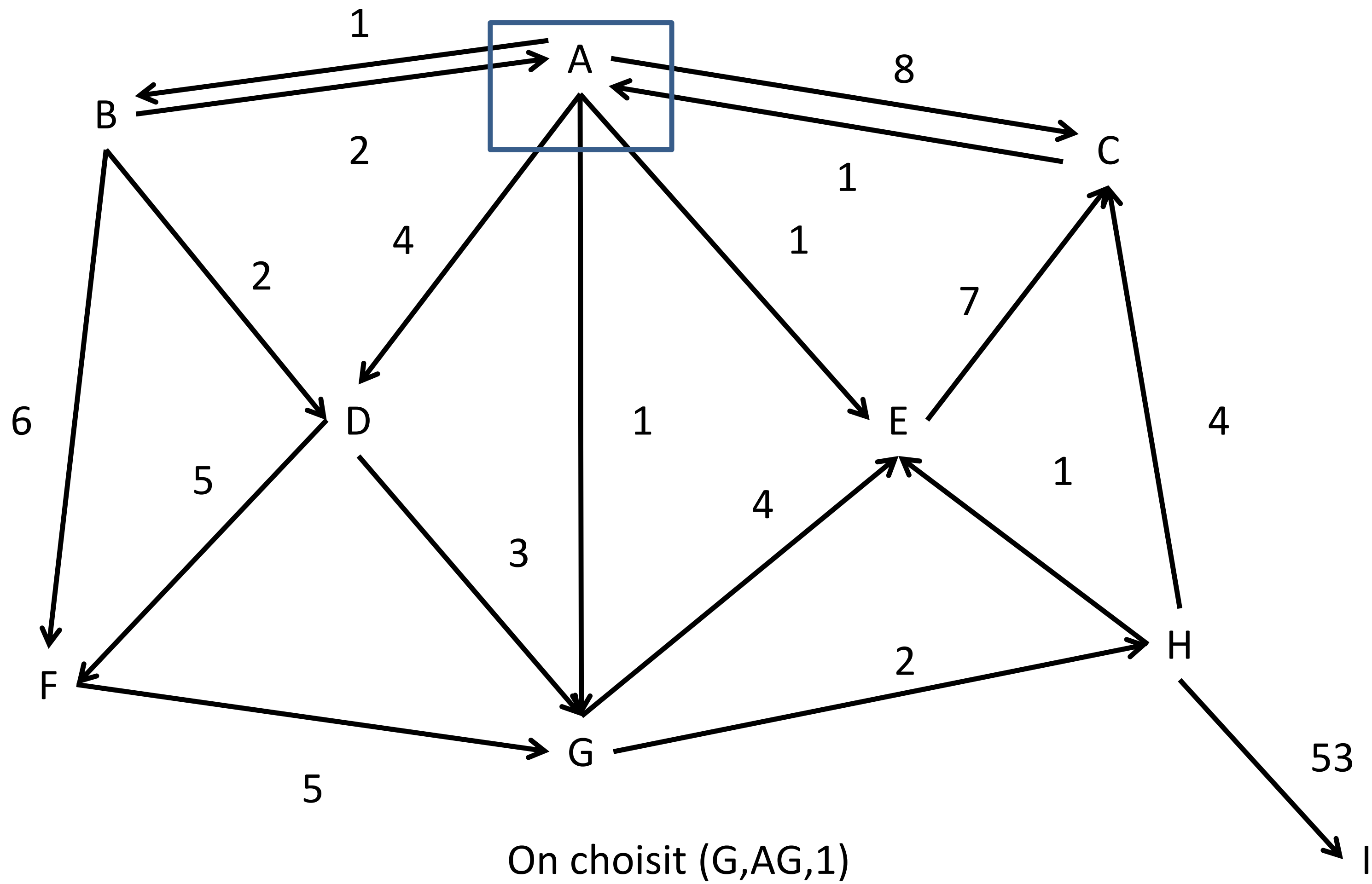
```
DébutCode
    U’ ← {}; fermé ← {s}
    Pour chaque sommet t de X faire
        D[t] ← ∞;
    FinPour
    D[s] ← 0; cocycle ←{}
    Pour chaque successeur y de s faire
        Insérer (y, sy, v(sy)) dans cocycle;
    FinPour
```

Algo Dijkstra (code 2)

```
Tant que cocycle ≠ {} faire
    Choisir (u, vu, val) tel que val minimal dans cocycle
    Retirer (u, vu, val) de cocycle
    Si non (u ∊ fermé) alors
        Insérer u dans fermé; D[u] ← val; insérer vu dans U'
        Pour chaque t ∊ (Succ(u) – fermé) faire
            Insérer (t, ut, D[u] + v(ut)) dans cocycle
        FinPour
    FinSi
FinTantQue
FinCode
```

A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	∞	∞	∞	∞

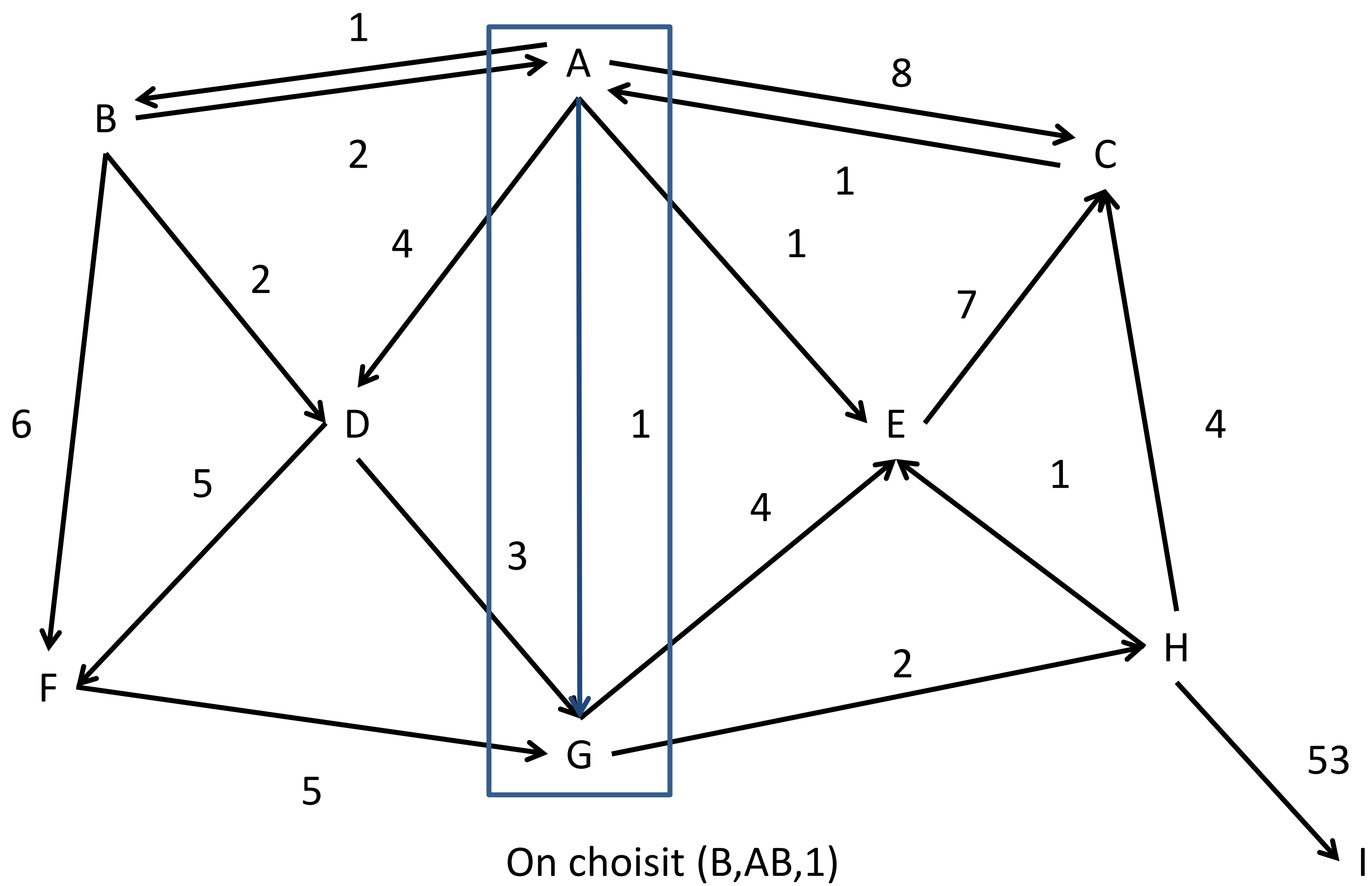
cocycle = {(B,AB,1), (C,AC,8), (D,AD,4), (E,AE,1), (G,AG,1)}



On choisit (G,AG,1)

A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	∞	1	∞	∞

cocycle = {(B,AB,1), (C,AC,8), (D,AD,4), (E,AE,1), (E,GE,5), (H,GH,3)}



On choisit (B,AB,1)

Question

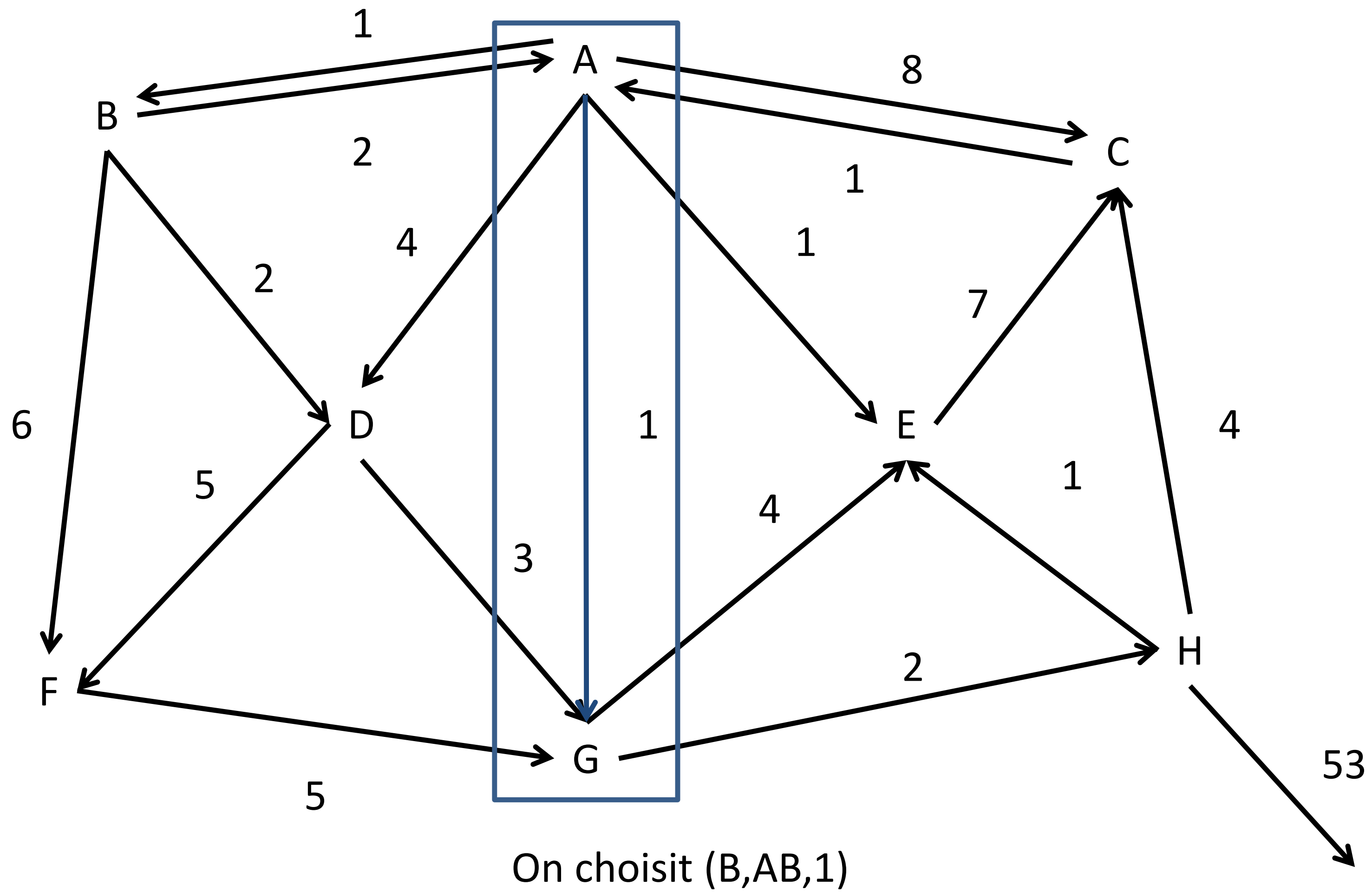
- Doit-on conserver les deux triplets (E, AE, 1) et (E, GE, 5) ?

- Non puisque le chemin passant par G est plus long que le chemin passant par A

- On conserve seulement le triplet (E, AE, 1)

A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	∞	1	∞	∞

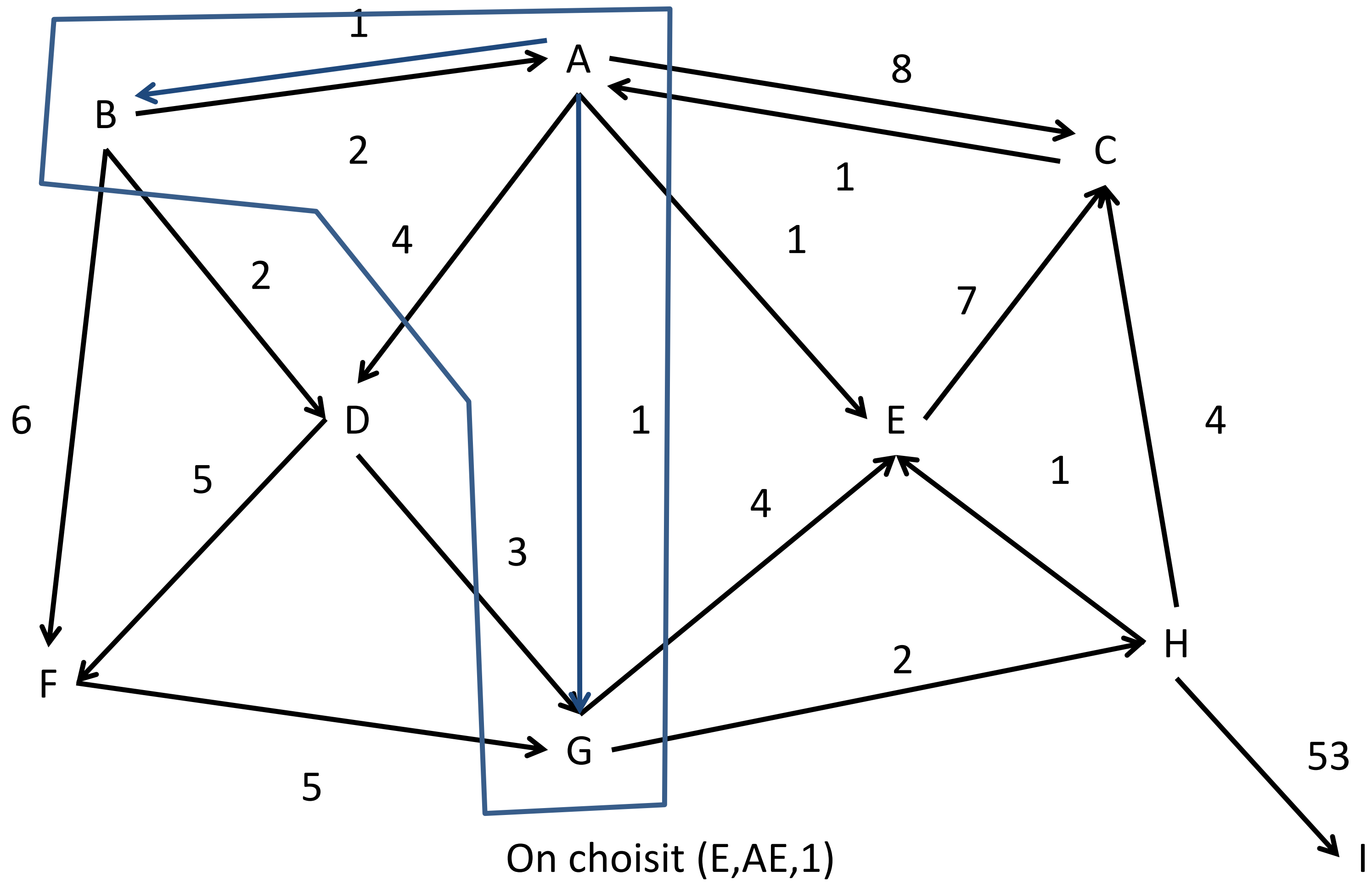
cocycle = {(B,AB,1), (C,AC,8), (D,AD,4), (E,AE,1), (H,GH,3)}



On choisit (B,AB,1)

A	B	C	D	E	F	G	H	I
0	1	∞	∞	∞	∞	1	∞	∞

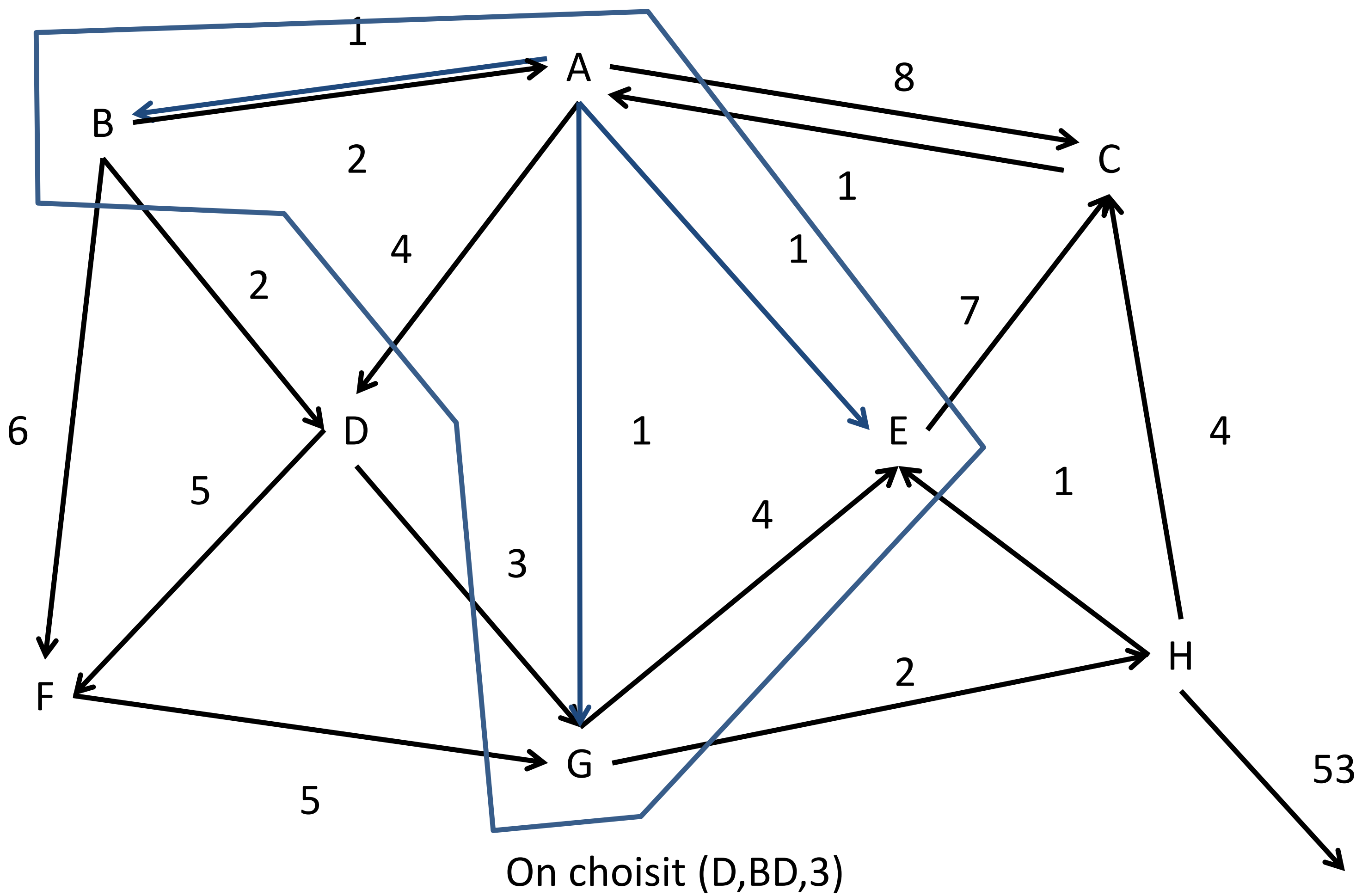
cocycle = {(C,AC,8), (D,BD,3), (E,AE,1), (H,GH,3), (F,BF,7)}



On choisit (E,AE,1)

A	B	C	D	E	F	G	H	I
0	1	∞	∞	1	∞	1	∞	∞

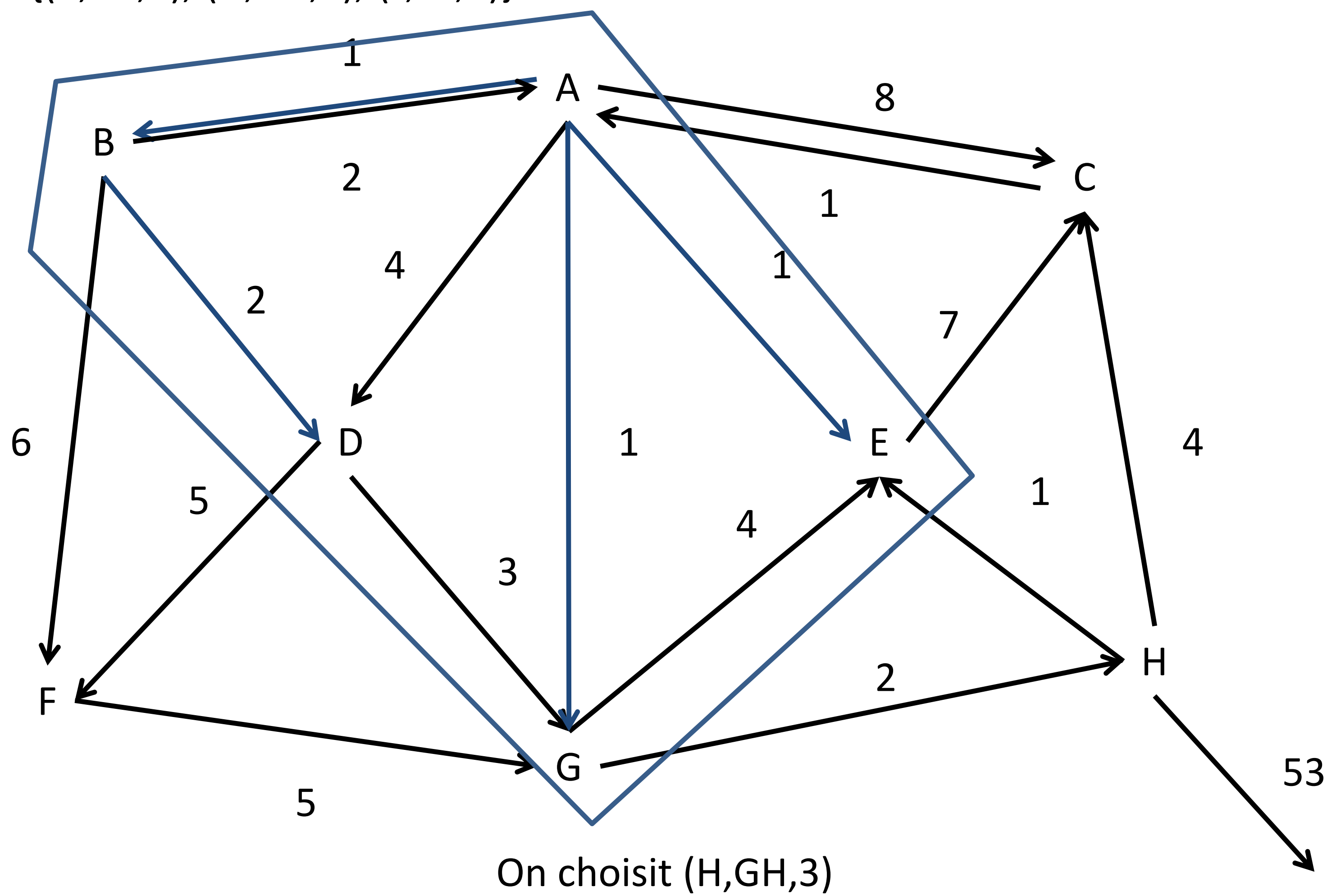
cocycle = {(C,AC,8), (D,BD,3), (H,GH,3), (F,BF,7)}



On choisit (D,BD,3)

A	B	C	D	E	F	G	H	I
0	1	∞	3	1	∞	1	∞	∞

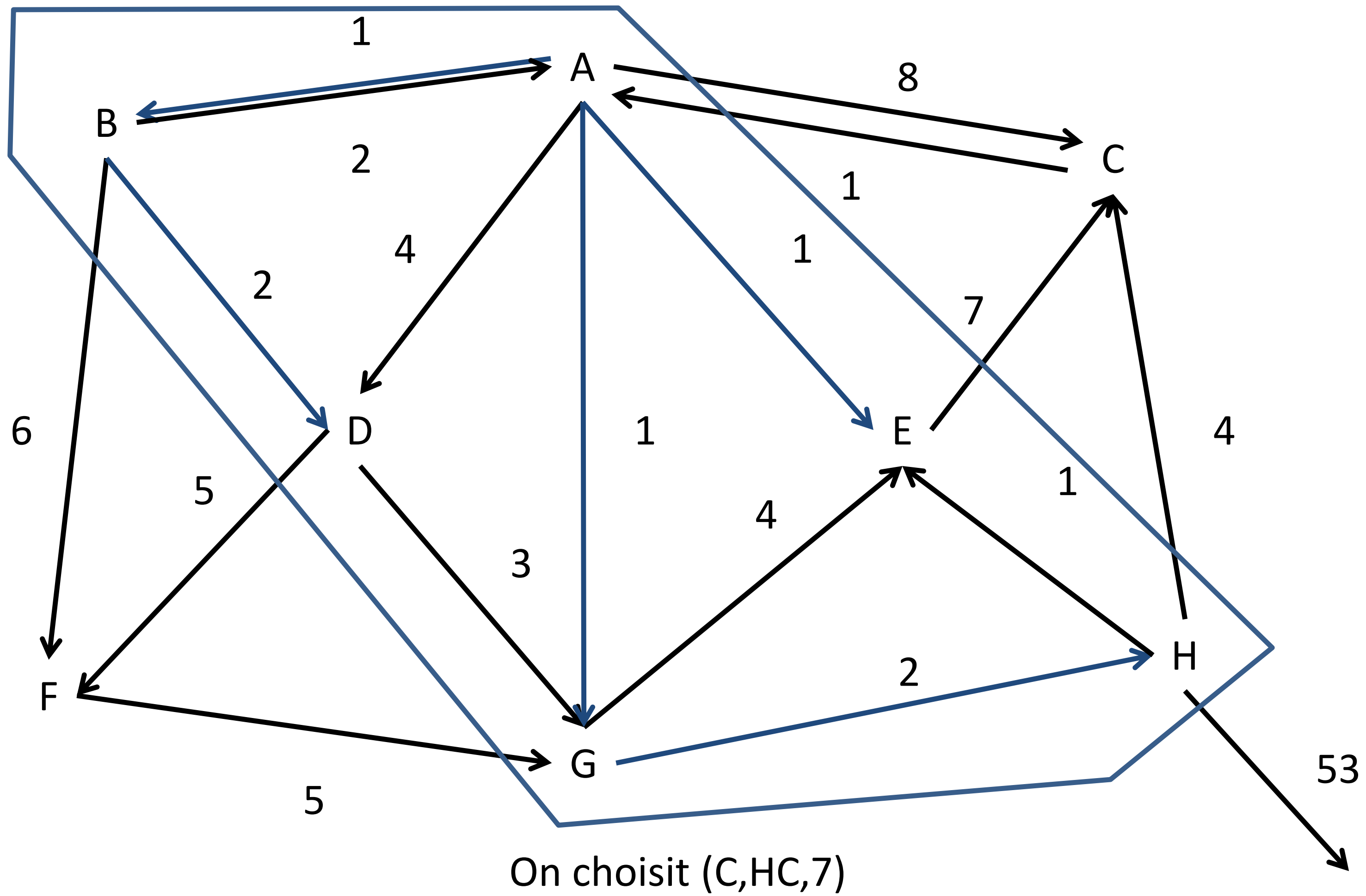
cocycle = {(C,AC,8), (H,GH,3), (F,BF,7)}



On choisit (H,GH,3)

A	B	C	D	E	F	G	H	I
0	1	∞	3	1	∞	1	3	∞

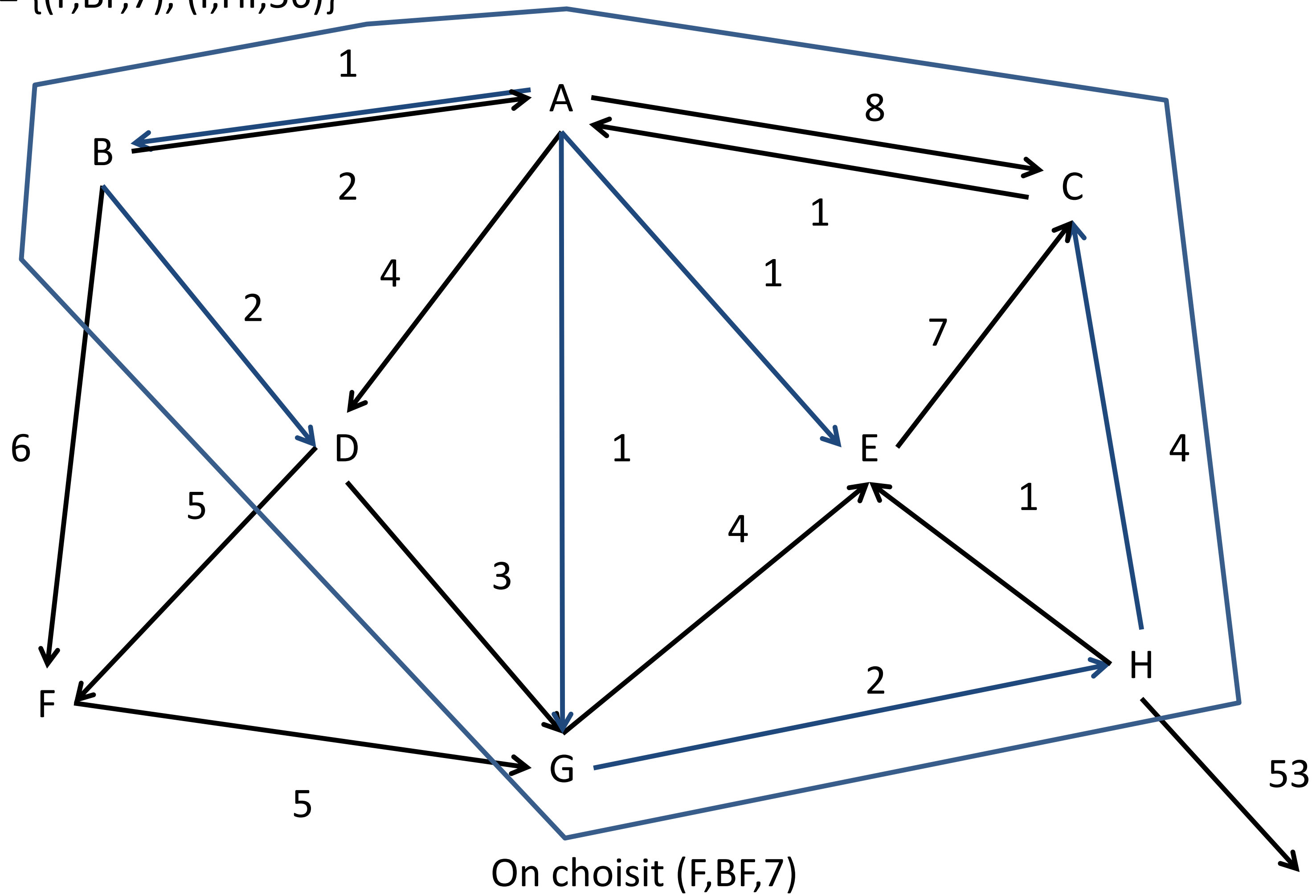
cocycle = {(C,HC,7), (F,BF,7), (I,HI,56)}



On choisit (C,HC,7)

A	B	C	D	E	F	G	H	I
0	1	7	3	1	∞	1	3	∞

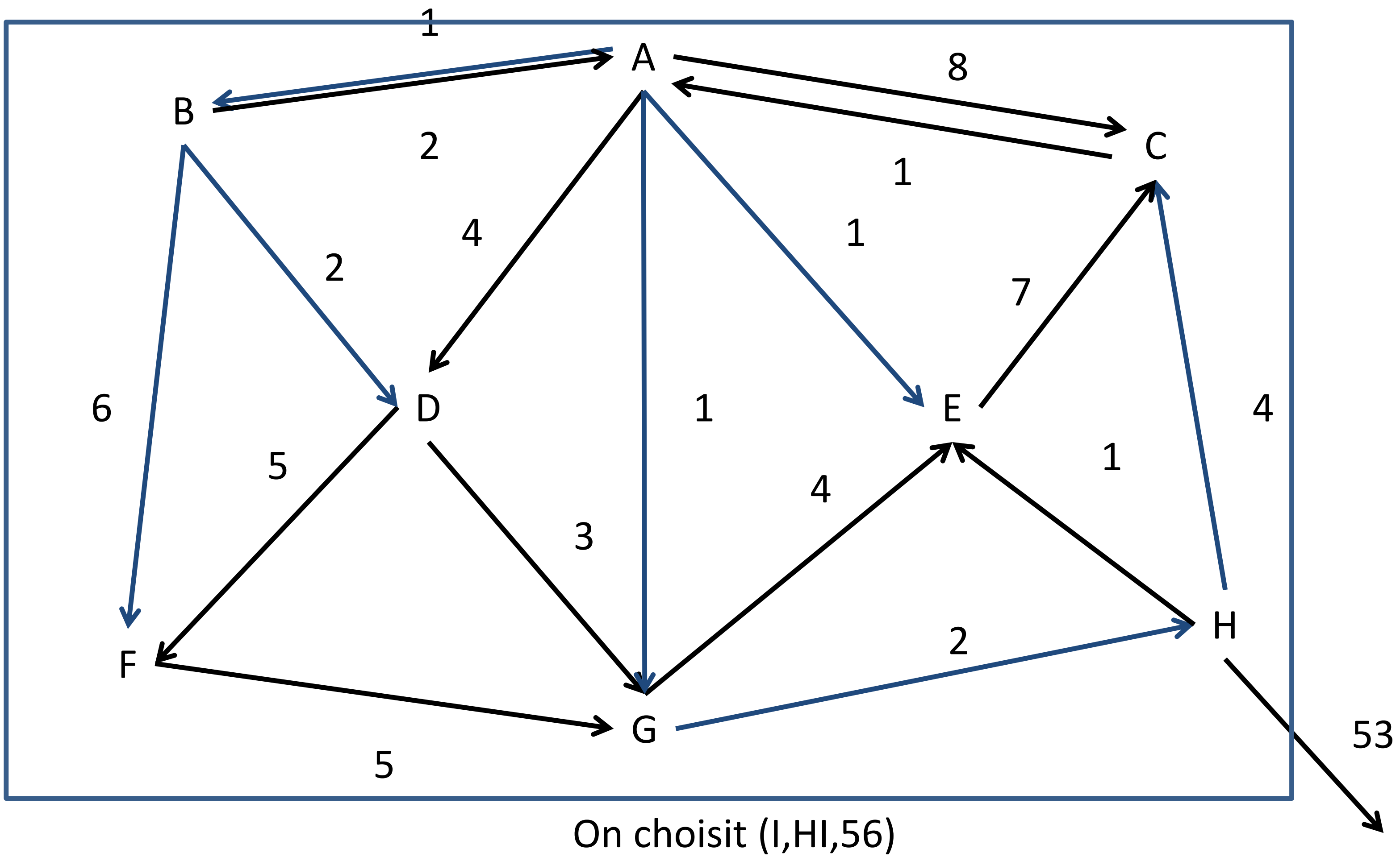
cocycle = {(F,BF,7), (I,HI,56)}



On choisit (F,BF,7)

A	B	C	D	E	F	G	H	I
0	1	7	3	1	7	1	3	∞

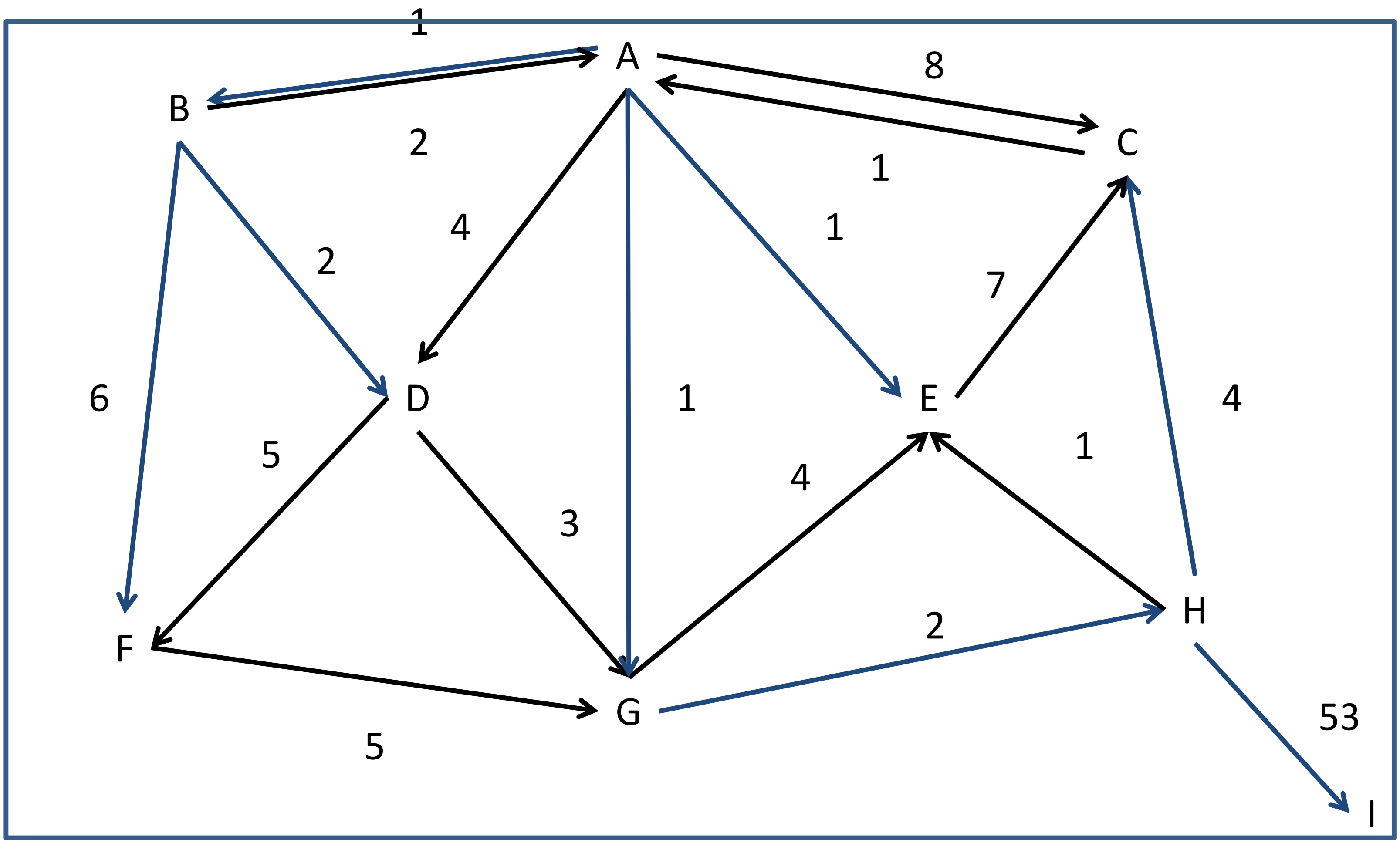
cocycle = {(I,HI,56)}



On choisit (I,HI,56)

A	B	C	D	E	F	G	H	I
0	1	7	3	1	7	1	3	56

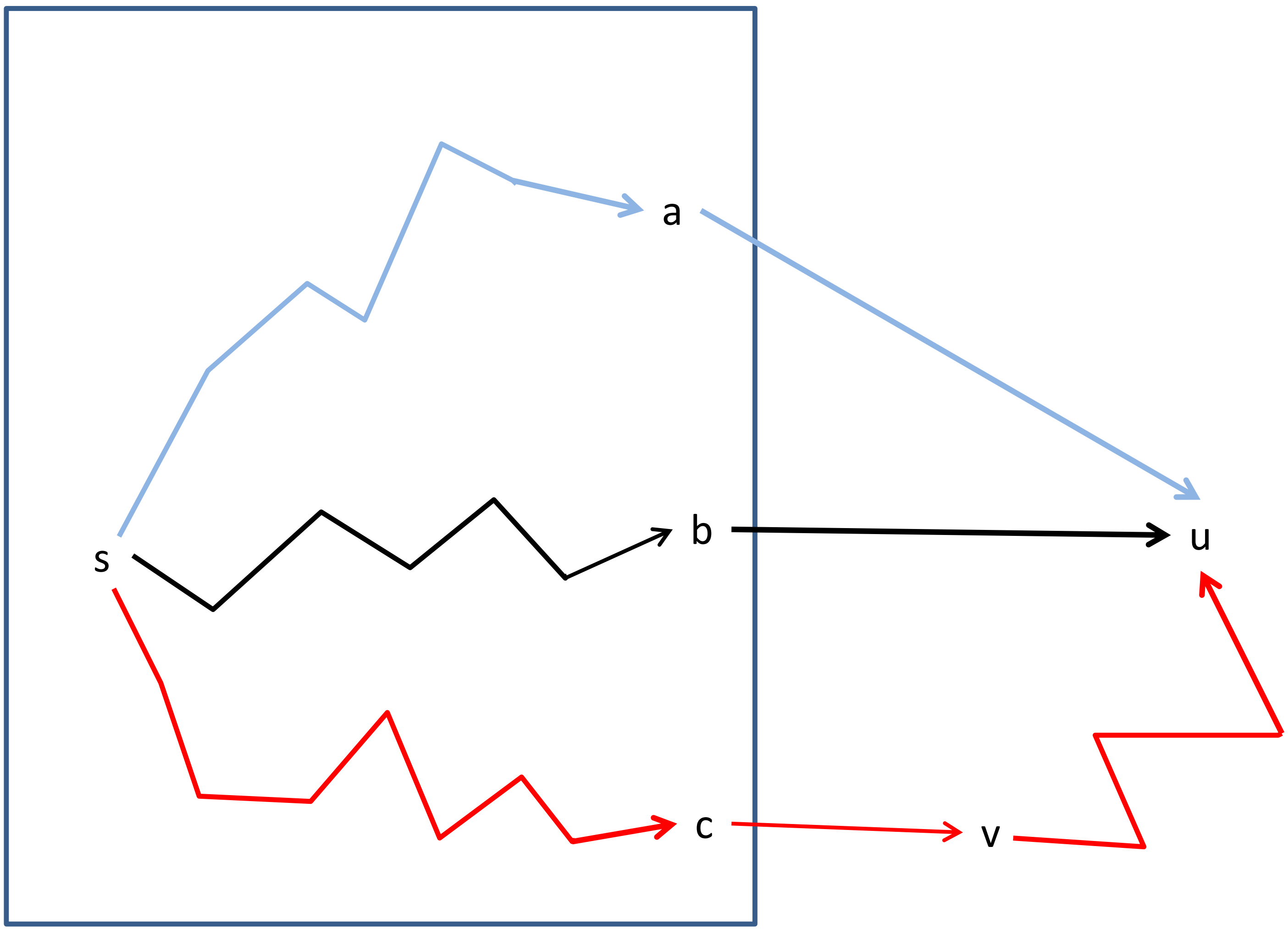
cocycle = {}



Fin de l’algo

Pourquoi a-t-on les chemins de poids minimaux ?

- La preuve se fait par l'absurde, on se place à l'étape où l'on introduit un sommet u dans fermé. On a alors dans GCPM un chemin qui va de s à u qui a pour poids D[u].
- Si ce chemin n'est pas le plus court il y a deux cas à étudier
- Posons que le chemin trouvé soit s…au


a
b
s
u
c
v

Cas 1 noir

- Si le chemin optimal est de la forme s…bu avec tous les sommets du chemin compris entre s et b dans fermé (Chemin noir)
- Alors c'est le triplet (u,bu,D[b]+v(bu)) qui devrait être traité puisqu'il est de valeur inférieure au triplet (u,au,D[a]+v(au))

Cas 2 Rouge

- Le chemin optimal est de la forme s…cv…u avec tous les sommets du chemin compris entre s et c dans fermé et v en dehors
- Le poids de ce chemin est donc inférieur à notre chemin s…au. Comme toutes les valeurs sont positives le poids de s…cv…u est donc supérieur au poids du chemin s…cv
- v est donc choisi avant u contradiction

Complexité

- n^2 pour une représentation par matrice
- m log n pour une représentation par liste d'adjacence (même structure de donnée que Prim)

FIN DE LA PREMIÈRE PARTIE