# Router et acheminer des informations dans un réseau

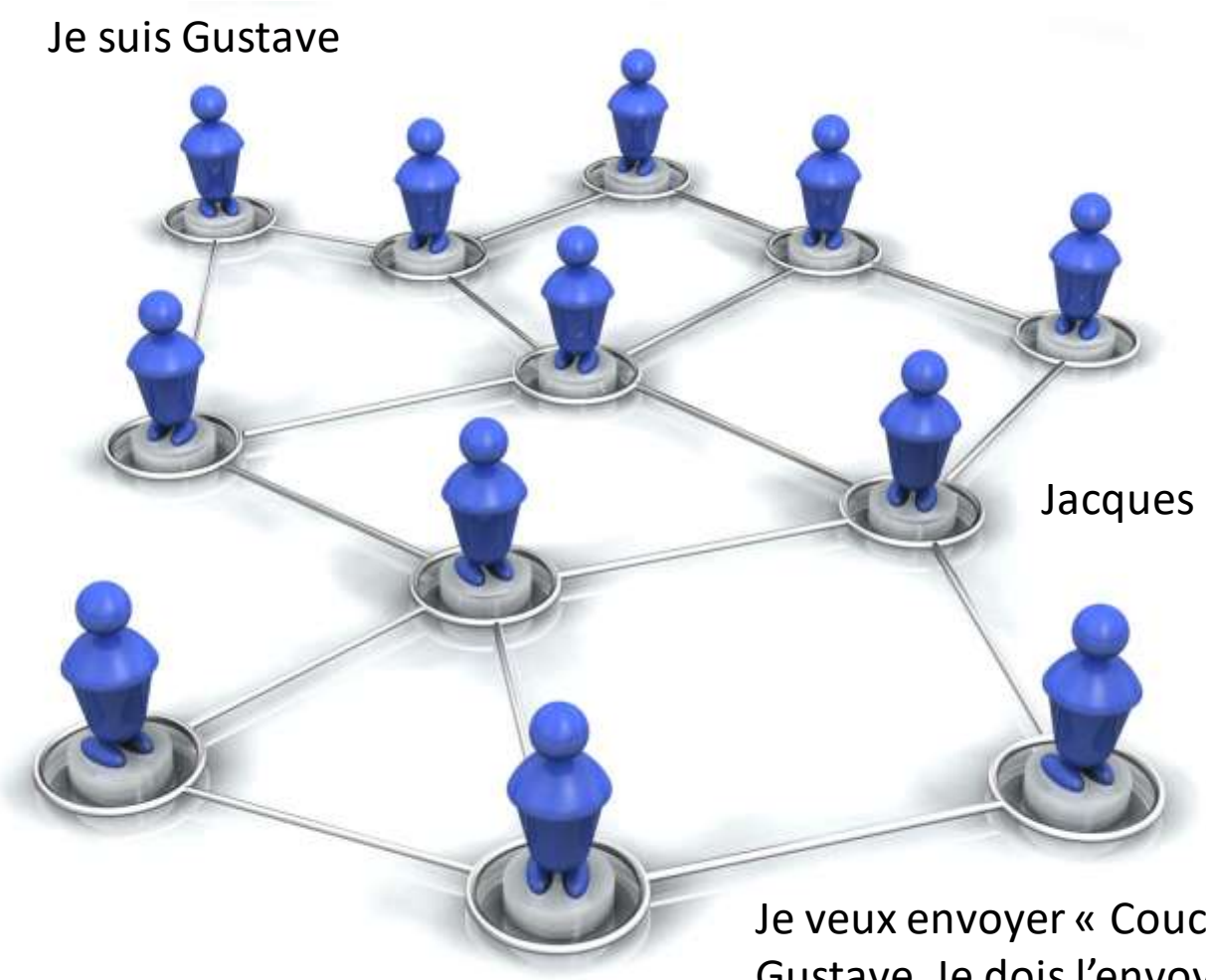
Alain Cournier

# INTRODUCTION (AÏE)

# Le problème

- Communiquer est un des fondements des systèmes distribués
- On souhaite acheminer une information $i$ vers une destination $d$.
- Or un nœud ne connait que des informations sur son voisinage. Donc si l'information est à destination du nœud ou de l'un de ses voisins dans le réseau tout va bien. Que faire dans le cas contraire ?

Je suis Gustave



Jacques

Jean

Je veux envoyer « Coucou » à Gustave. Je dois l'envoyer à Jean ou à Jacques ?

# Première Solution

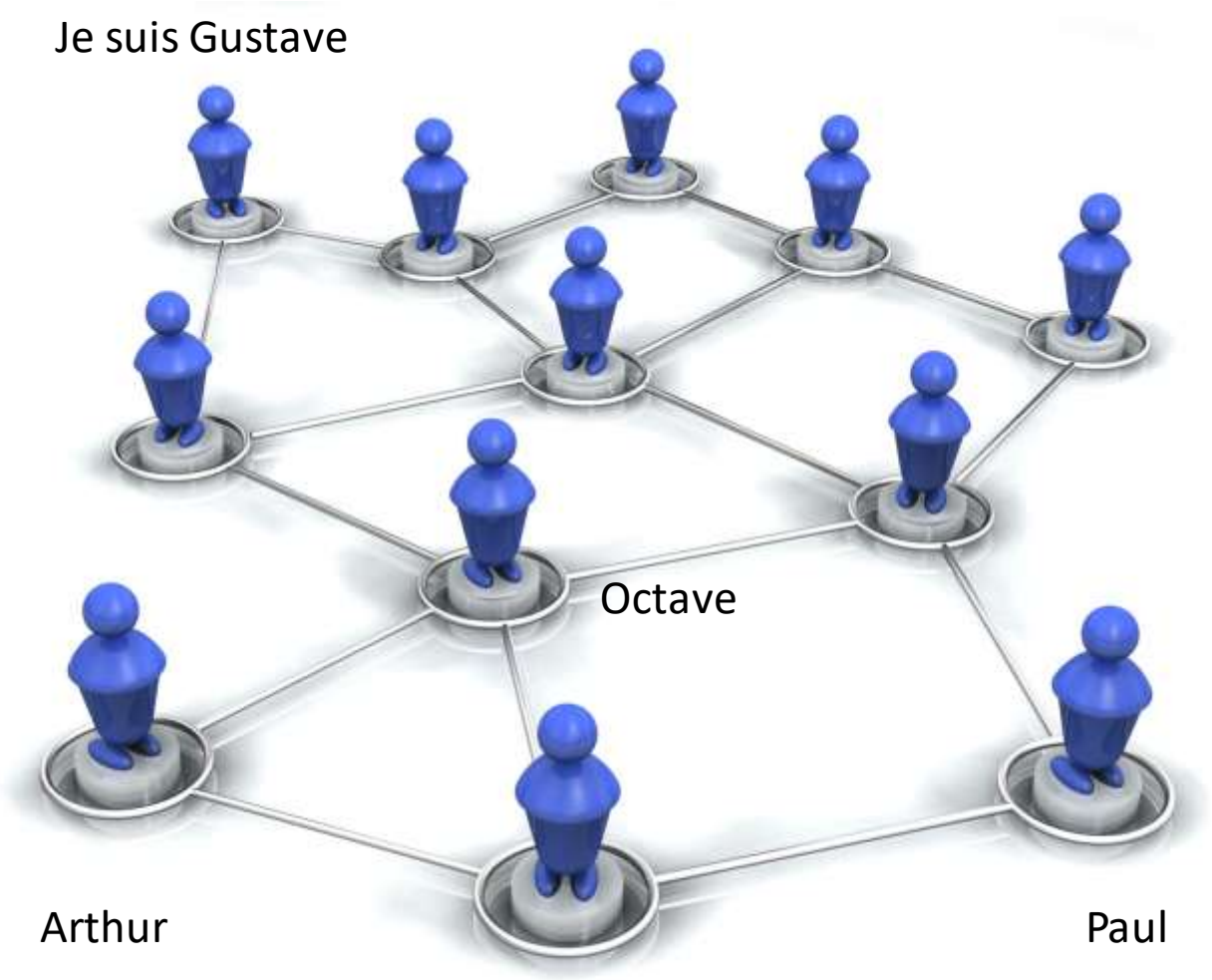
- On l'envoie à tout le monde.

- On réalise une diffusion vers tous les nœuds du réseau :
  - Avantage : si le destinataire est dans le réseau il reçoit l'information
  - Inconvénient : Problèmes de confidentialité puisque tous les nœuds ont l'information.

# Routage et plus courts chemins

- Le problème est complexe car si un nœud x veut envoyer une information à un nœud y, il ne sait pas obligatoirement où se trouve notre nœud y dans le réseau.

- Il doit donc calculer un chemin à partir du nœud x afin que l'information atteigne y




Je veux envoyer « Coucou » à Gustave. Je dois l'envoyer à Paul à Arthur ou à Octave?

# Seconde solution

- Seconde solution : choisir un chemin de x vers y. L'information circulera seulement sur ce chemin ; C'est la notion de routage.

- Avantage : Gain de confidentialité.

# Seconde solution

- Le routage est le moyen qui permet à un nœud de choisir un lien vers un de ses voisins pour envoyer une information vers une destination du réseau.
- Pour cela, on utilise souvent une table de routage.

# Objet du routage

- Qu'est-ce que le routage ?

  Le routage est le processus de sélection du chemin dans un réseau. Un réseau informatique est composé de nombreuses machines, appelées nœuds, et de chemins ou de liaisons qui relient ces nœuds. La communication entre deux nœuds d'un réseau interconnecté peut s'effectuer par de nombreux chemins différents. Le routage est le processus qui consiste à sélectionner le meilleur chemin à l'aide de certaines règles prédéterminées.

  Source : https://aws.amazon.com/fr/what-is/routing/

# Objet du routage

- Comment fonctionne le routage ?

  Les données circulent sous la forme de paquets de données. Chaque paquet possède un en-tête qui contient des informations sur la destination prévue du paquet. Lorsqu'un paquet se déplace vers sa destination, plusieurs nœuds intermédiaires peuvent être impliqués.

  Lorsqu'un paquet arrive, le routeur recherche d'abord son adresse dans une table (ou fonction) de routage qui permet de choisir le meilleur voisin pour poursuivre l'acheminement du paquet vers sa destination. Il transfère ensuite le paquet vers le nœud voisin choisi.

  Inspiré de : https://aws.amazon.com/fr/what-is/routing/

# Router mais selon quels critères ?

- Minimiser le nombre de saut ?
- Minimiser le coût du chemin ?
- Maximiser la bande passante ?
- Minimiser le risque (maximiser la sécurité) ?

# Router mais selon quels critères ?

- Doit-on avoir des chemins alternatifs ? (combien ?)
- Ce routage est il calculé une fois pour toute ?
- Ce routage peut-il se recalculer à la volée ?
- Accepte-t-il les changements de topologie ?

# Premiers problèmes liés au routage

- Comment construire les tables de routage ?
- Comment réduire la taille de ces tables ?

# Comment représenter ce chemin

- On peut calculer ce chemin
  - à la demande (lorsqu'on veut envoyer une information) : Routage réactif
  - Une fois pour toute : Routage Pro-actif.
- Dans le cas du routage pro-actif, une fois calculés ces chemins sont conservés sur les nœuds du réseau.
  - Sous forme de fonctions
  - Sous forme de tables

# COMMENT REPRÉSENTER LES INFORMATIONS NÉCESSAIRES AU ROUTAGE ?

# Le problème : Et pour Amiens ?

# Le problème

- Un nœud lorsqu'il reçoit un message doit être capable de déterminer par quel lien le cheminement du message vers sa destination devra se poursuivre.
- Il existe au moins trois façon de résoudre ce problème

# Solution 1 : Le message contient la route qu'il doit suivre

- C'est la solution du GPS qui calcule une fois la route entre le point de départ et le point d'arriver. Après quoi il sait quelle route emprunter à chaque carrefour.
- Inconvénients :
  - l'expéditeur connait tous les chemins vers tous les destinataires.
  - La longueur du message.

# Solution 2 : Le nœud connait la direction à prendre pour chaque destination

- Idéal pour le message qui ne doit transporter que sa destination.
- Inconvénient : Chaque nœud connait toutes les destinations du réseau.

# Solution 3 : Utiliser des informations externes

- Je suis au Groenland.
- Je veux aller aux Açores

- Cap au sud !

- C’est par exemple le routage géographique.

# Discution

- La solution 1 est utilisé dans les réseaux fortement dynamiques
- La solution 2 est utilisé dans les réseaux fixes
- Il existe des solutions mixtes par exemple dans les réseaux de téléphonie cellulaire.

# CODER UNE TABLE DE ROUTAGE

# Codage d’une table

- Coder une table de routage, c’est établir sur chaque nœud du réseau une correspondance entre une identité (la destination) et un canal.

# Exemple : poids des arêtes 1

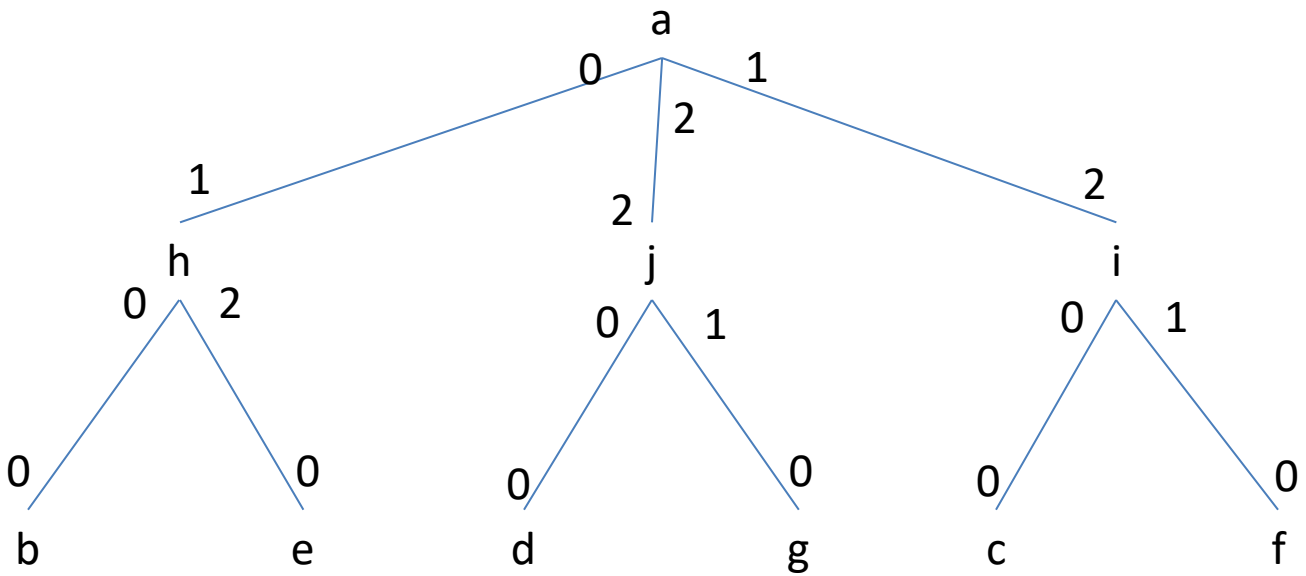



Table de routage de a

| **Dest** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** |
|---|---|---|---|---|---|---|---|---|---|---|
| Canal | Loc. | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |

# Exemple : poids des arêtes 1

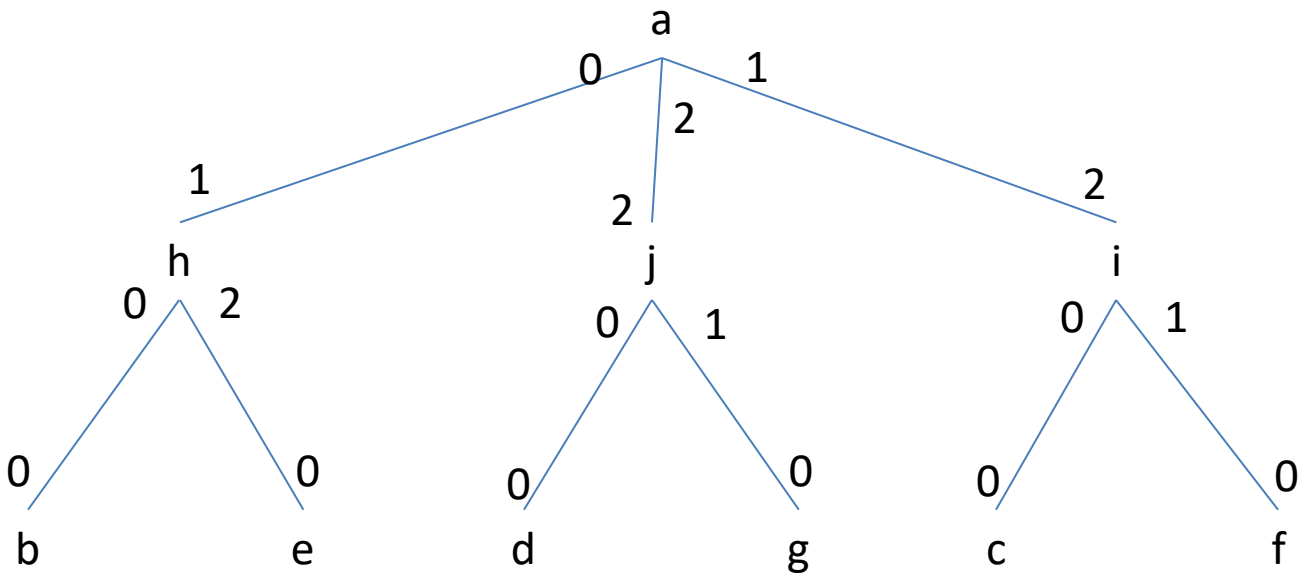



Table de routage de a

| canal | 0 | 1 | 2 |
|---|---|---|---|
| Dest | <b, e, h> | <d, g, j> | <c, f, i> |

# LIMITER LA TAILLE DES TABLES DE ROUTAGE IMPLANTÉES SUR UN NŒUD

# Taille de la table de routage

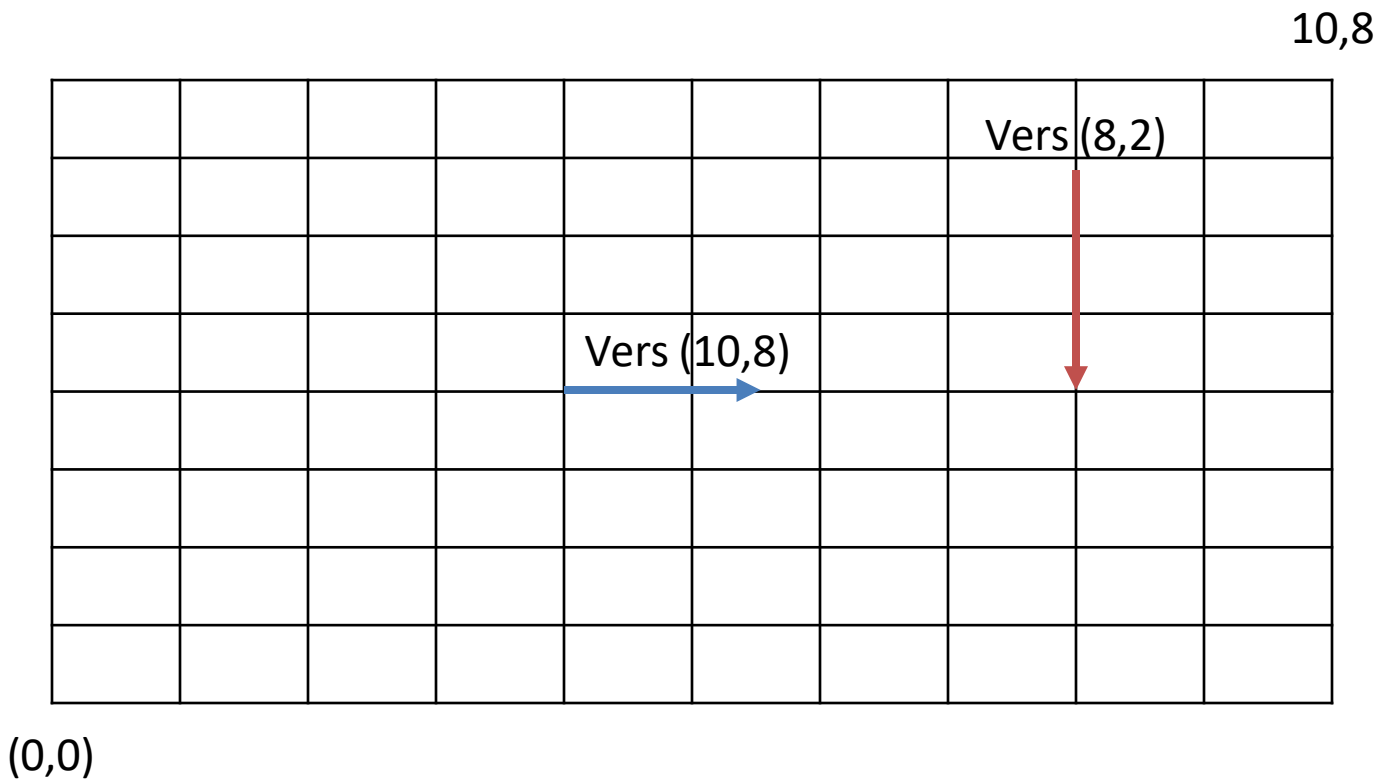
- Ici pour la taille d'une table de routage on retient l'espace utilisé pour représenter cette information.

# Aide par la topologie

- Dans une grille (à 1, 2 ou 3 dimensions) les coordonné du nœud (courant) et de la destination suffissent pour déterminer quel voisin prolongera le chemin (si toutes les arêtes ont le même poids)

- Il faut que l'étiquetage des nœuds soit cohérent

# Exemple sur une grille



# Aide par la topologie

- Il en est de même pour des topologies telles que :
  - Les anneaux
  - Les tores
  - Les arbres
  - Les hypergraphes

# Cas simple : Arbre avec numérotation préfixe (ou postfixe)

- Dans ce cas la table de routage (de chaque nœud) associe un intervalle (càd un couple d'identités) à chaque canal.
- Donc localement, une table de routage aura pour encombrement :

$$2\delta \log(n)$$

Avec $\delta$ le degré du nœud et n le nombre de nœuds du réseau.

- Soit au global un espace de 4m log(n) où m est le nombre de liens du réseau.

# Exemple : poids des arêtes 1

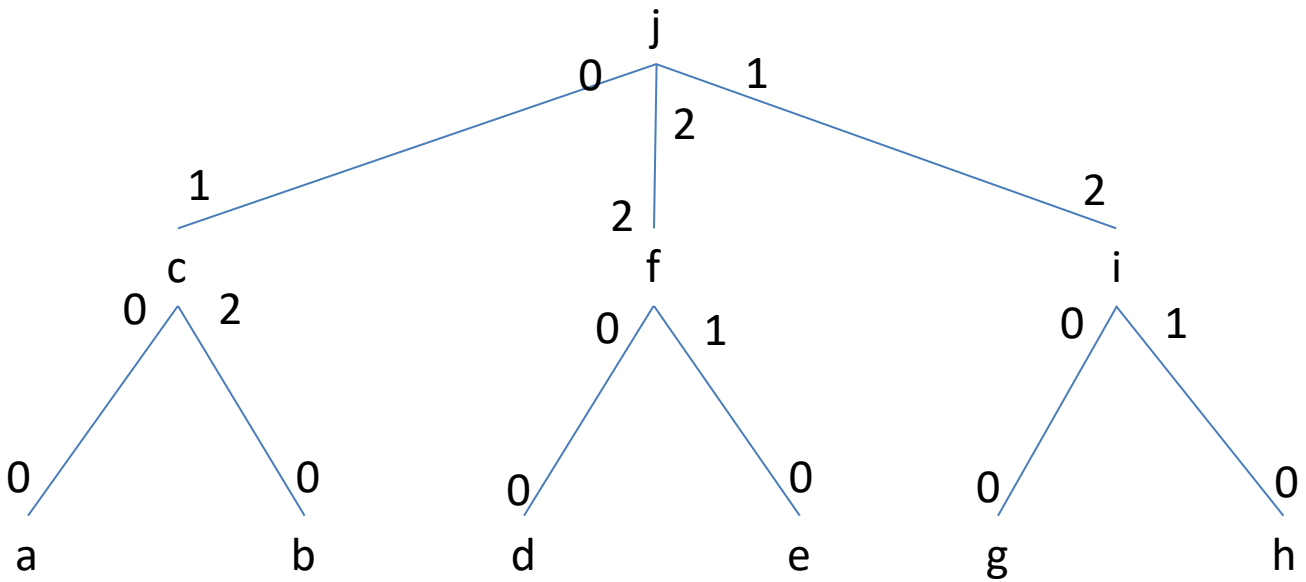



Table de routage de j

| **canal** | **0** | **1** | **2** |
|---|---|---|---|
| Dest | [a..c] | [g..i] | [d..f] |

# Routage par intervalle

- Une table de routage telle que pour tout nœud x :
  - Les destinations associées à chaque canal peuvent être représentées par un intervalle
  - L'intersection des intervalles de deux canaux distincts est vide
  - Chaque destination est présente dans un canal.
- Est un routage par intervalle

# Exemple : poids des arêtes 1

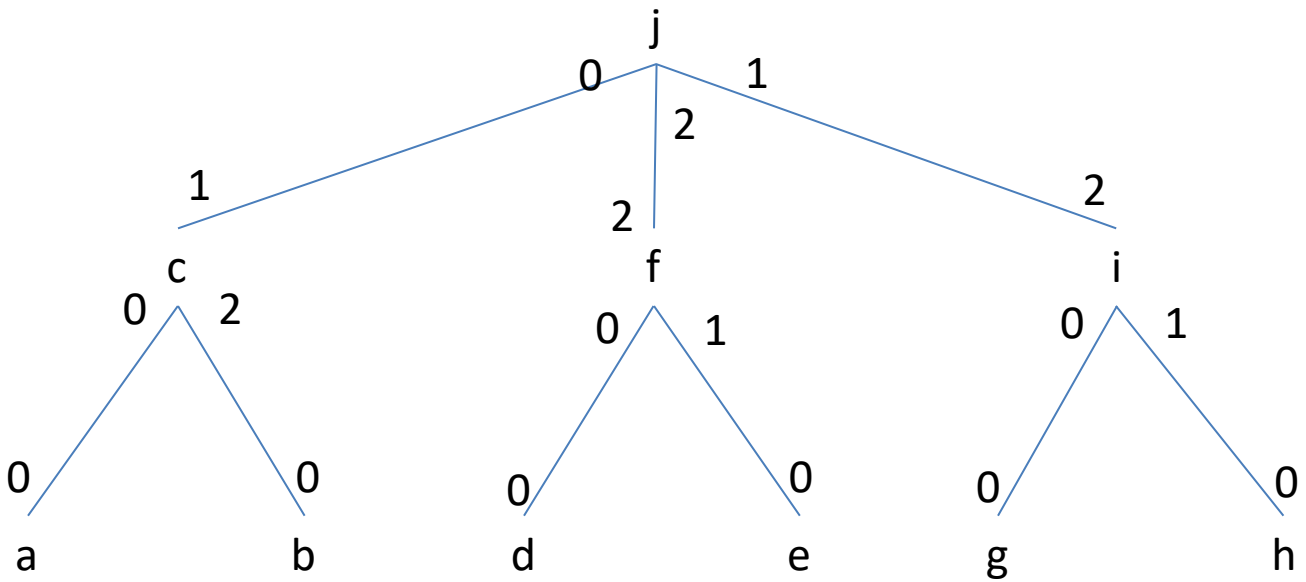



Table de routage de f

| canal | 0 | 1 | 2 |
|---|---|---|---|
| Dest | [d] | [e] | [g..c] |

# Routage par k intervalles

- Une table de routage telle que pour tout nœud x :
  - Les destinations associées à chaque canal peuvent être représentées par k intervalles
  - L'intersection des intervalles de deux canaux distincts est vide
  - Chaque destination est présente dans un canal.
- Est un routage par k intervalles

# Avantages/Inconvénient

- Rendre les tables locales plus compactes

- L'identité des nœuds doit être adaptée au réseau (Renommage).
- Dans le cas général on perd l'acheminement du message par le chemin de poids minimal.

# Exercices

- Montrez que les parcours d'un arbre en préfixé, postfixé et infixé donne un routage par intervalle sur chaque nœud.

- Montrez que si x est un point d'articulation, alors il est possible de créer au moins un intervalle de routage pour chaque nœud. Du réseau.

# Et quand la topologie est quelconque ?

- Il est toujours possible de calculer un arbre couvrant et de faire le routage sur cet arbre
- Conséquence : on renonce au plus court chemin.

- Dans le cas général, on ne sais pas borner avec précision le rapport entre la distance dans le réseau et la distance dans l'arbre couvrant.
  - 2*Diamètre si c'est un arbre en largeur
  - n pour un arbre en profondeur

# Et quand la topologie est quelconque ?

Autre solution :

- On fait des sous-groupes

- Des groupes de sous-groupes

- Des sur-groupes de groupes
- …

# Et quand la topologie est quelconque ?

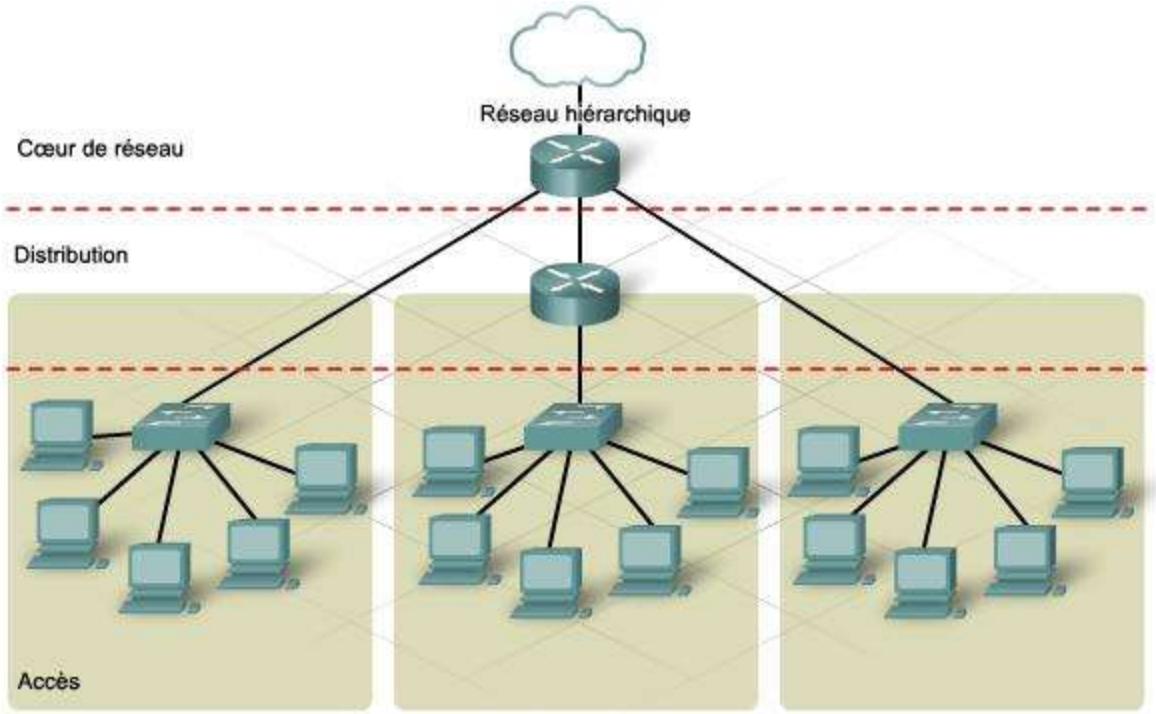
- Contraintes
  - Les sous-groupes, les groupes et les sur-groupes sont connexe.
  - La communication à l'intérieur d'un ensemble se fait uniquement en utilisant les entités de cet ensemble

# Et quand la topologie est quelconque ?

- Contraintes
  - Si a et b sont deux éléments d'un même ensemble et c un élément d'un autre ensemble alors les messages de c vers a et b sont acheminés de la même façon jusqu'à l'arrivé dans l'ensemble commun à a et b.
  - Assurer un étiquetage des nœuds cohérents permettant de minimiser la taille de la table de routage.

# Routage

- Le routage IP est une réponse partielle à ces questions :
  - Réseau hiérarchique
  - Machines spécifiques : Routeur
  - Liens entre divers sous réseaux utilisant des passerelles gérées par ces routeurs.



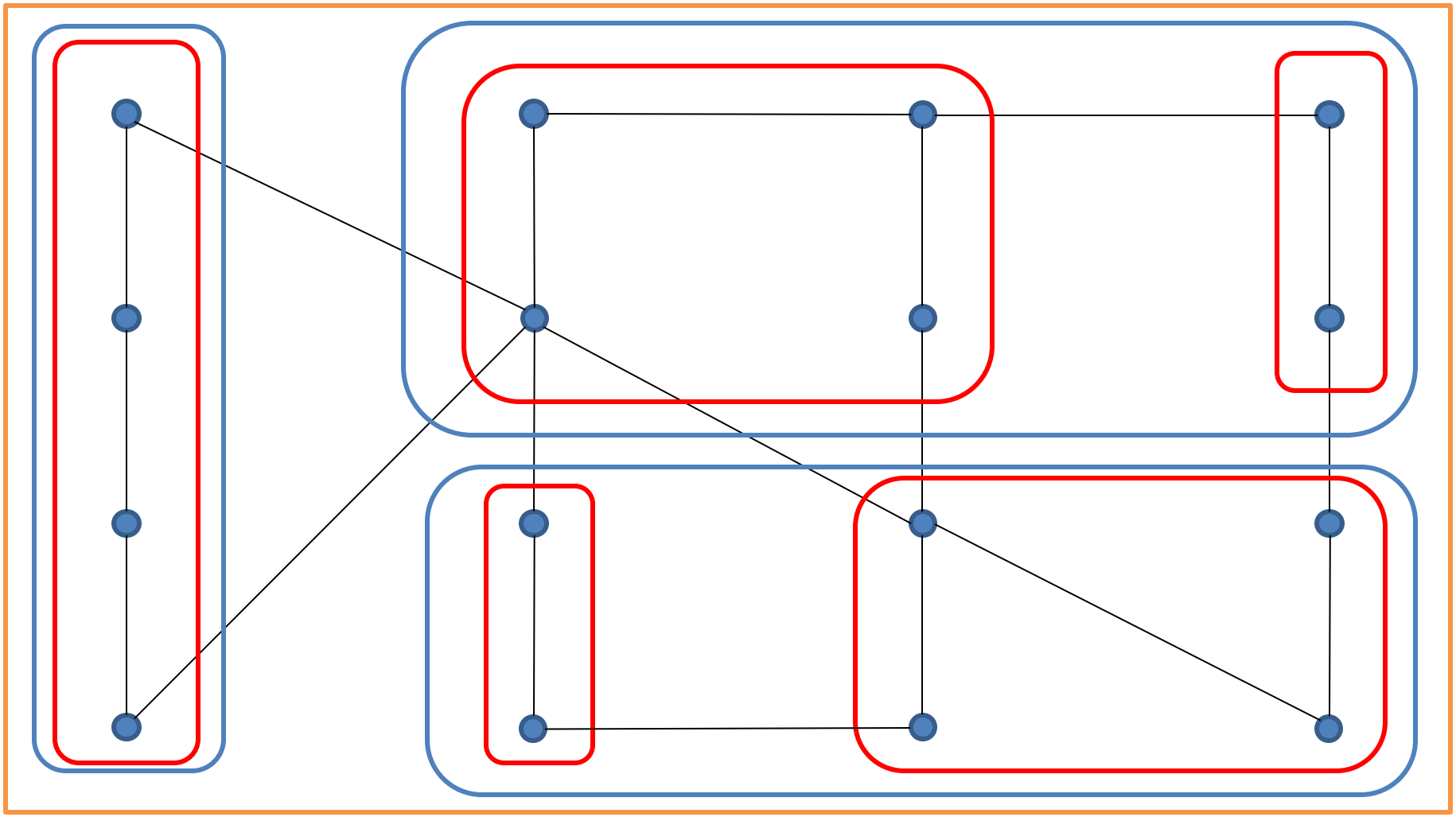
Trois domaines de diffusion distincts

# Routage

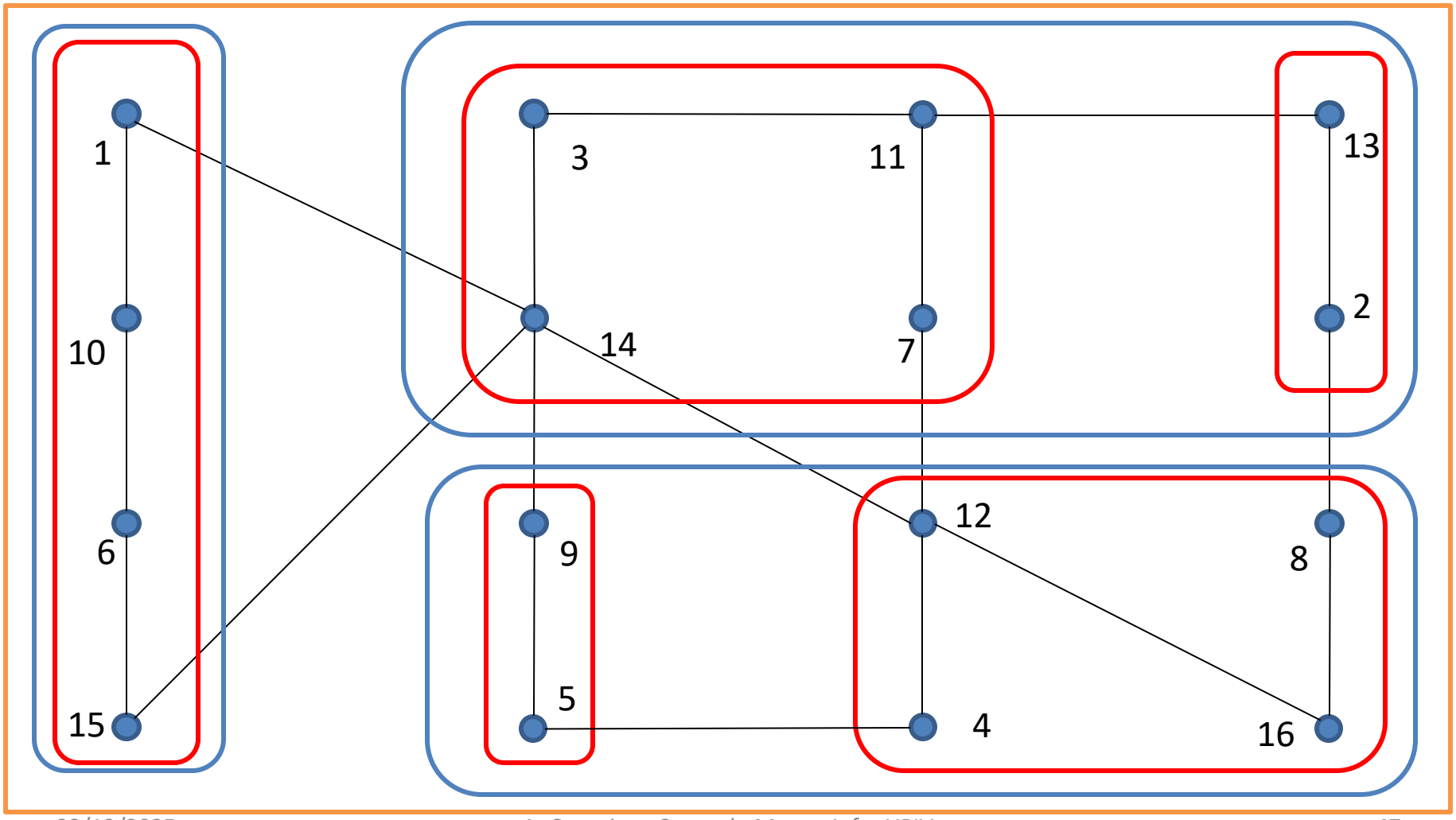
- Mais toute clustérisassion autorise un routage efficace répondant aux critères énoncés.

- Sous réserve d'un étiquetage des nœuds compatible.

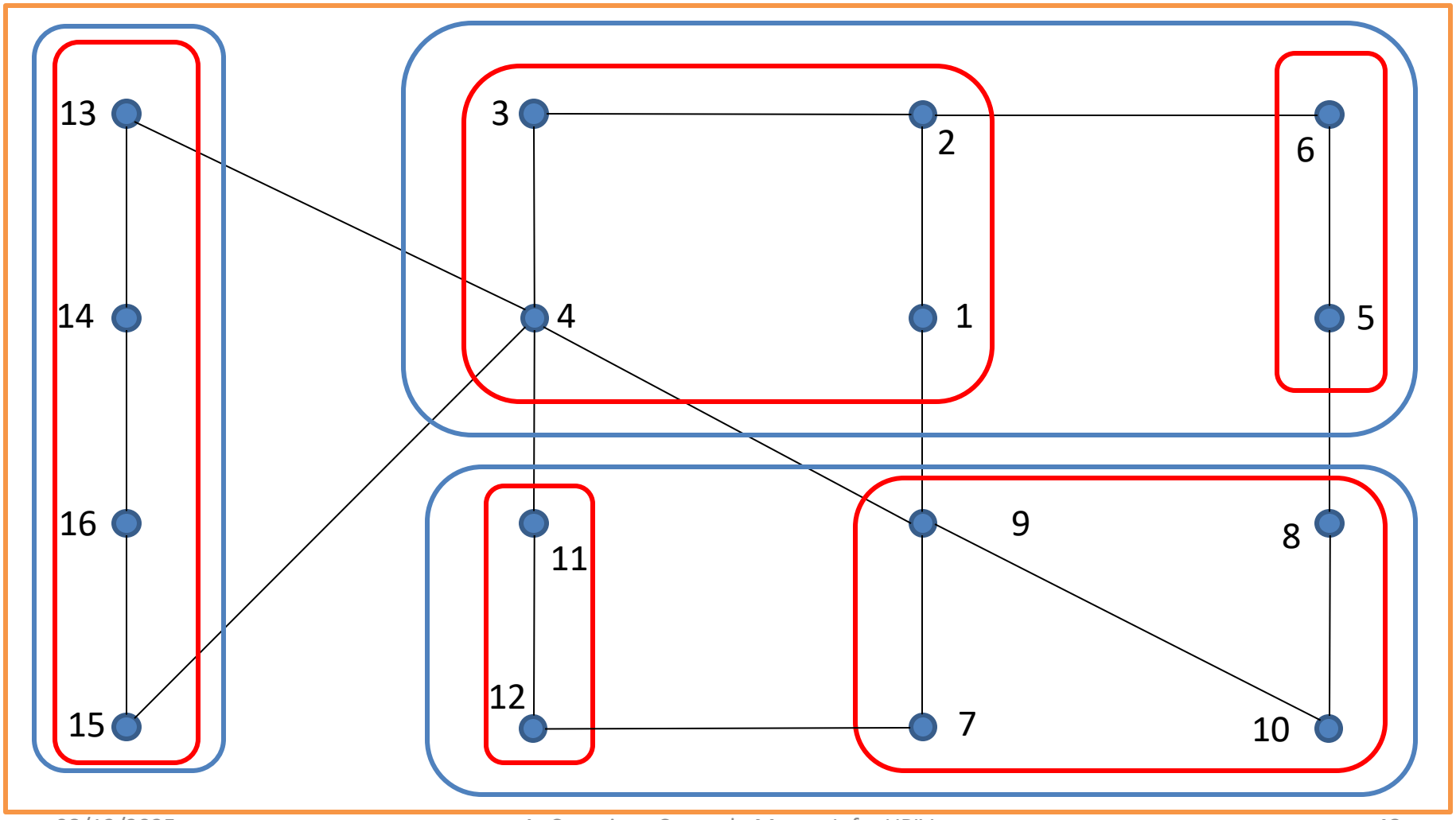
# Exemple



# Exemple



# Exemple



# Routage

- La clustérisation autorise une réduction de la taille des tables de routage

- Mais elle implique de renoncer au chemin de poids minimal (voir chemin entre 13 et 15)

# Le dilemme

- Comment trouver un compromis entre la taille de la table de routage et l'allongement de la longueur des chemins ?

Exercice : Modification des tables de routage induite par l'ajout d'un lien dans le réseau.

# Le dilemme

- Quelles sont les contraintes réelles ?

Exercice : Quelle est la taille maximale d'une table de routage selon IPv4 et IPv6 ?

# Le dilemme

Exercice : Quelle est la taille maximale d'une table de routage selon IPv4 et IPv6 ?

Réponse : Les routeurs sont configurés avec des tailles maximales pour les tables de routage… Certains de ces modèles peuvent avoir des limites de routage fixées à 1 048 576 routes pour IPv4, et à 131 072 pour IPv6

(source : mars 2021, What will happen when the routing table hits 1024k? | APNIC Blog)

# Le nom du destinataire dépend-t-il de sa position dans le réseau ?

- Vrai avec l'adresse IP par exemple
- Faux avec votre téléphone portable

- Pourtant dans les deux cas il faut acheminer l'information ?

# Routage

- Nous allons regarder le problème de la construction des tables de routage dans un réseau fixe.

- Les identités des nœuds du réseau seront considérée comme fixées.

# Le problème

- Un réseau peut être représenté par un graphe étiqueté G=(X,U,V)
- X ensemble de nœuds du réseau
- U ensemble de liens du réseau
- V une application qui à chaque lien du réseau associe un poids.

# Informations disponibles

- A-t-on une connaissance globale du réseau pour effectuer le calcul ?
- Le réseau doit-il être découvert au fur et à mesure du calcul ?
- Les valeurs permettant de calculer le meilleur chemin sont elles fixes ou variables au fil du temps ?
- L'évaluation peut-elle être multicritères ?

# Trouver le meilleur chemin : Algo de Dijkstra

- C’est un algorithme centralisé pour calculer les chemins de poids minimaux à partir d’un sommet de départ d vers tous les autres sommets du réseau.

# Algorithme de Dijkstra (hypothèse)

- Les hypothèses classiquement formulées
  - Pas de circuits de poids négatifs
  - L'ensemble des descendants de s est X
- On suppose que tous les poids sont positifs
- Nous allons nous placer dans le cas où le réseau est entièrement connu.

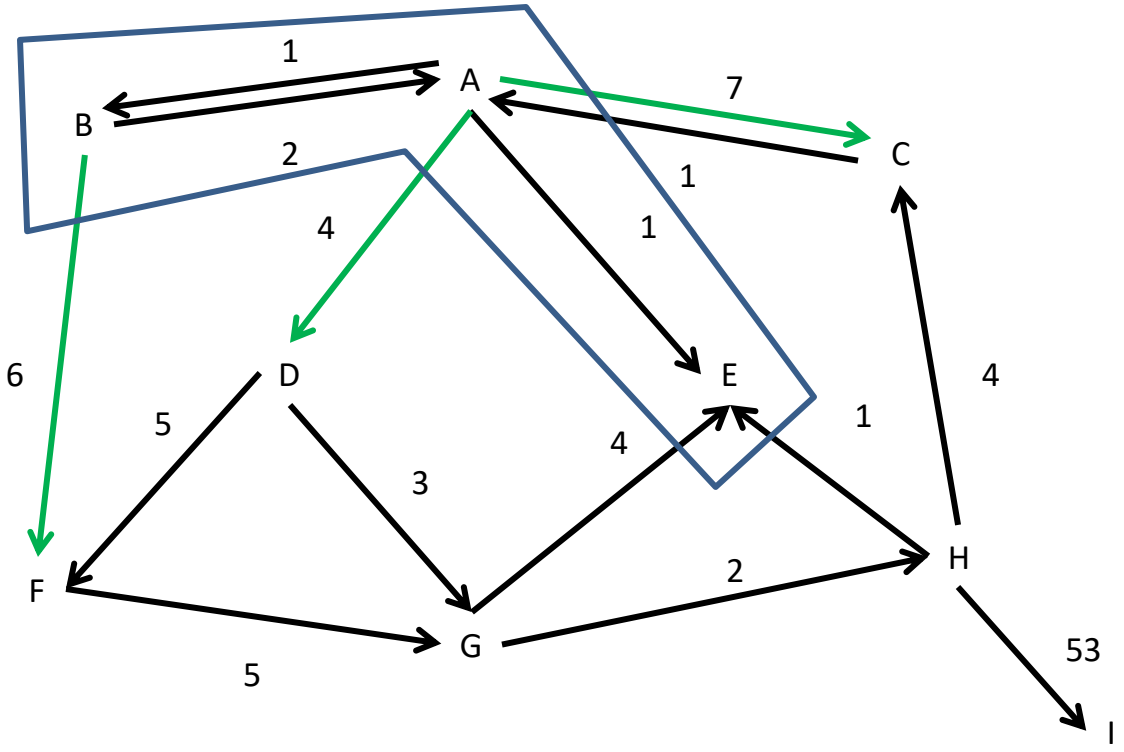
# Dijkstra (idée de l'algorithme)

- 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.

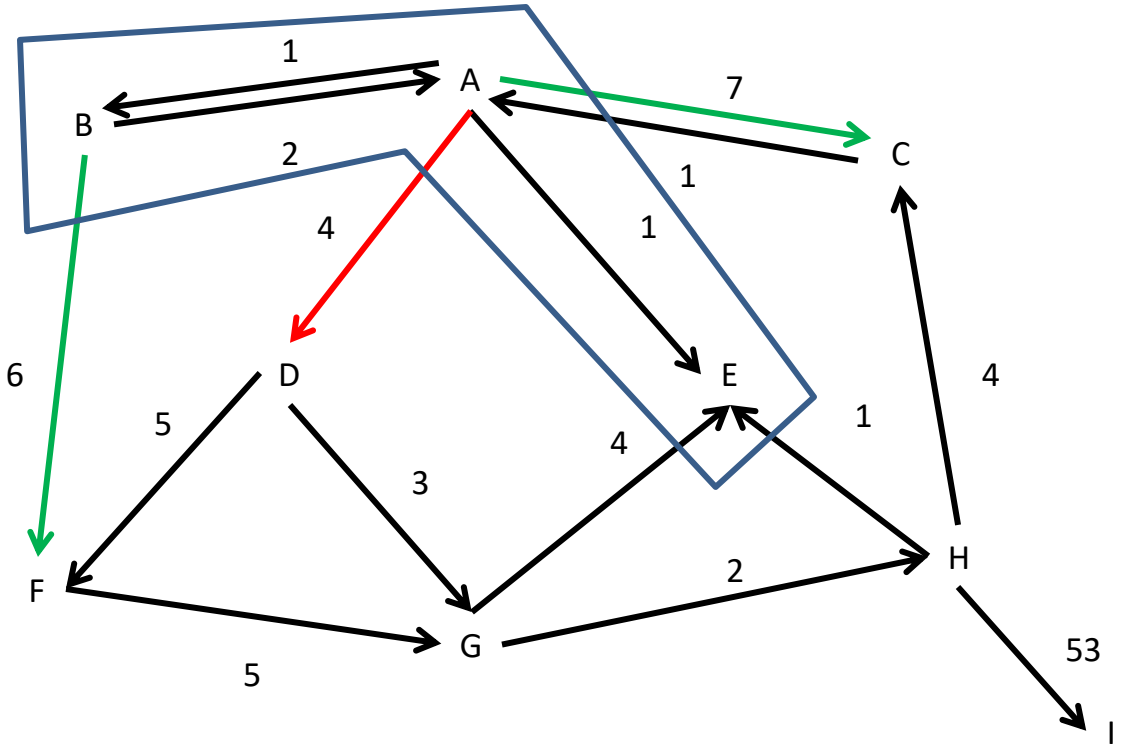
| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | ∞ | ∞ | 1 | ∞ | ∞ | ∞ | ∞ |



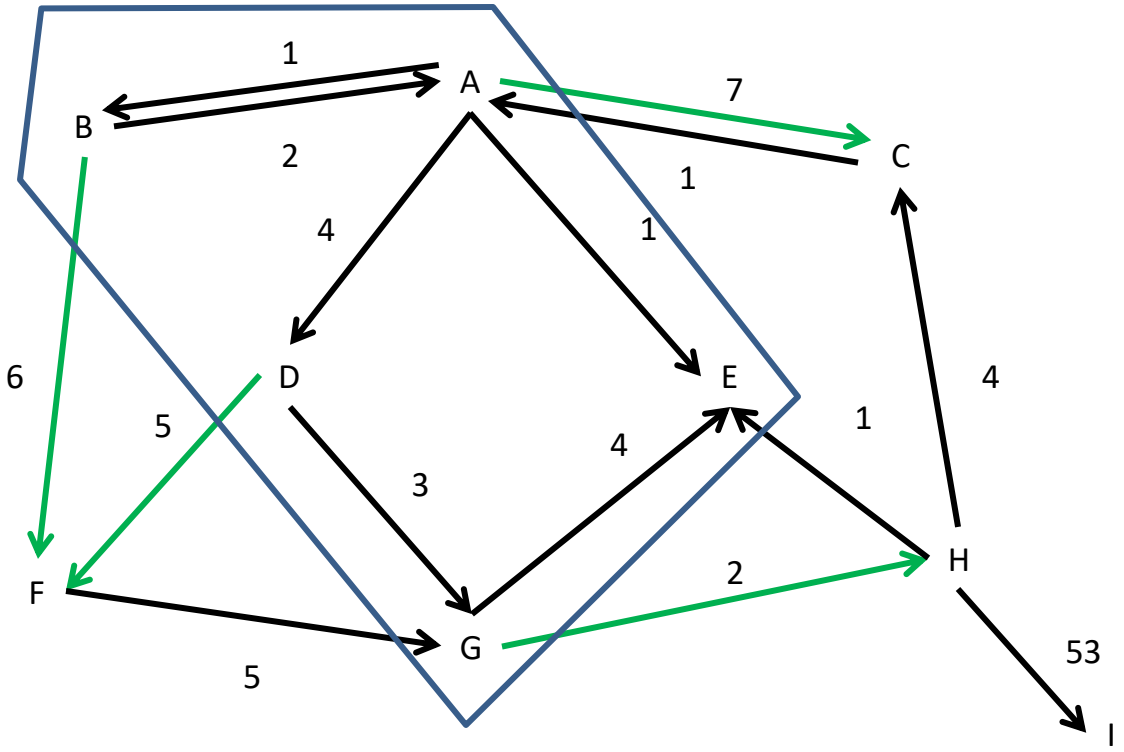
# Idée d'une étape de l'algorithme

- 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 | | |




Alain Cournier

# Valeurs initiales

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

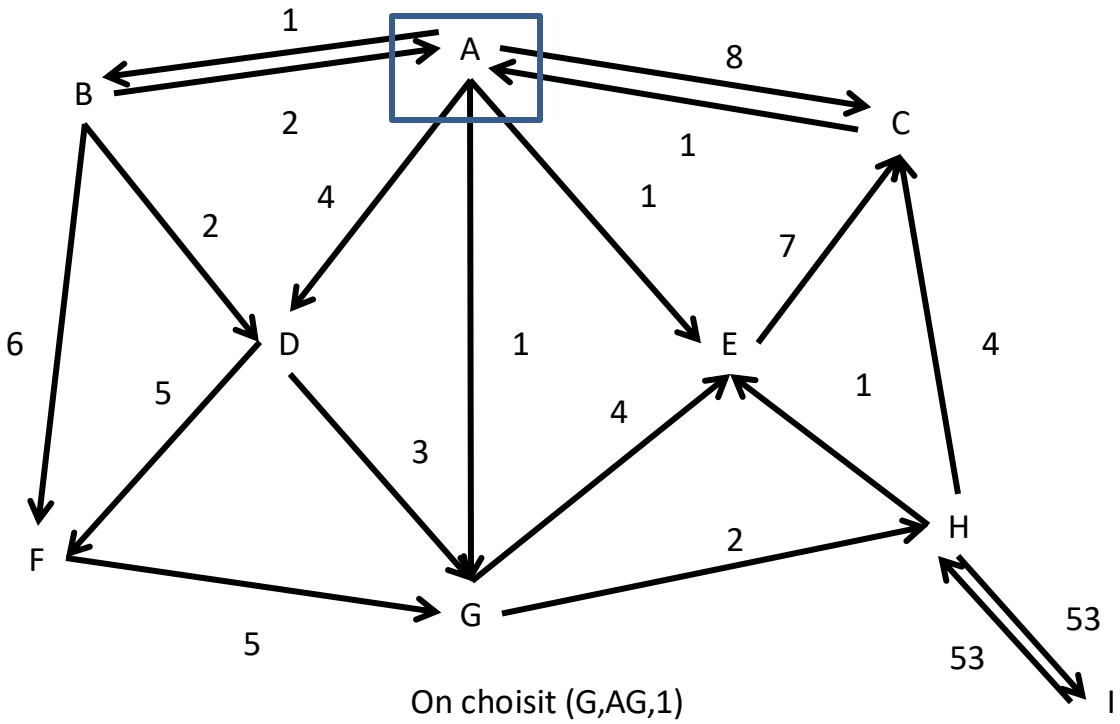
```
DébutCode
    U' ← {}; fermé ← {s}
    Pour chaque sommet t de X faire
        D[t] ← ∞;
    FinPour
    D[s] ← 0; cocycle ←{}
    Pour chaque successeur y de s faire
        Insérer (y, sy, v(sy)) dans cocycle;
    FinPour
```

# 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
```

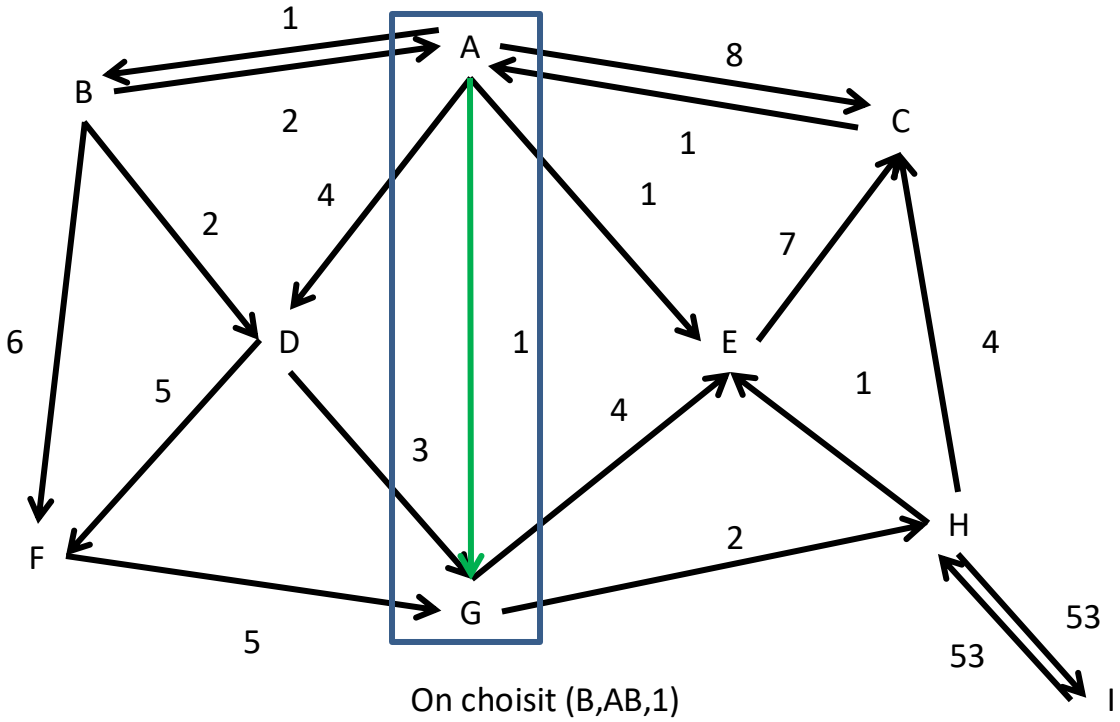
| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |

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



| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | 1 | ∞ | ∞ |

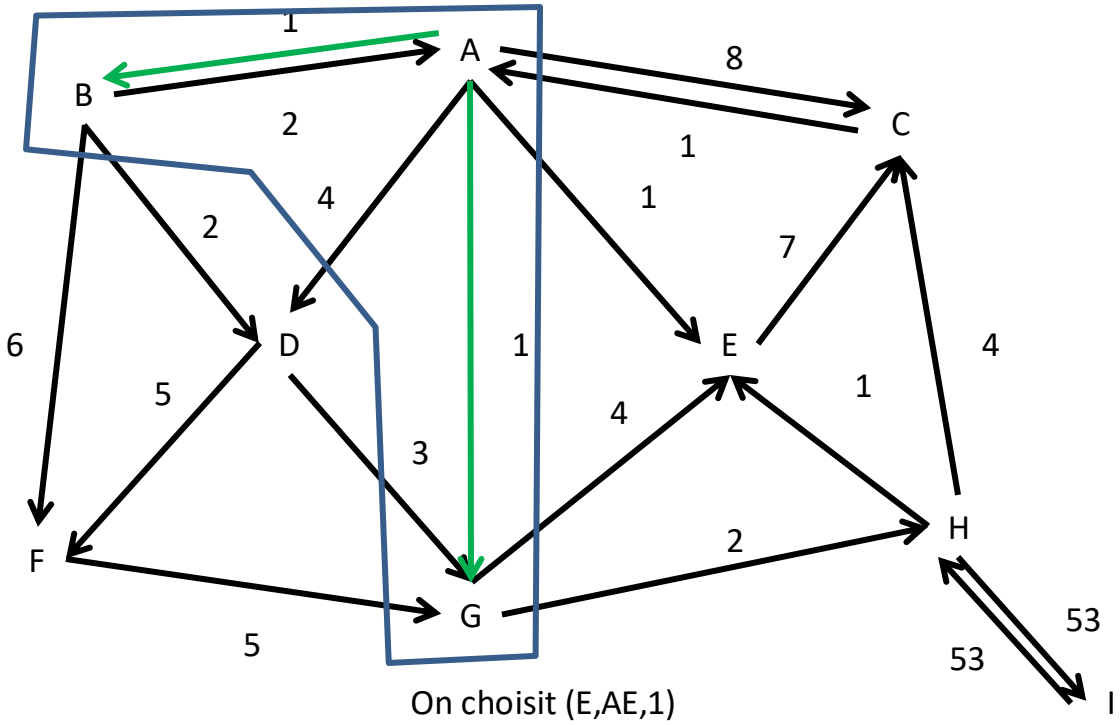
cocycle = {(B,AB,1), (C,AC,8), (D,AD,4), (E,AE,1), (E,GE,5), (H,GH,3)}




On choisit (B,AB,1)

| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | ∞ | ∞ | ∞ | ∞ | 1 | ∞ | ∞ |

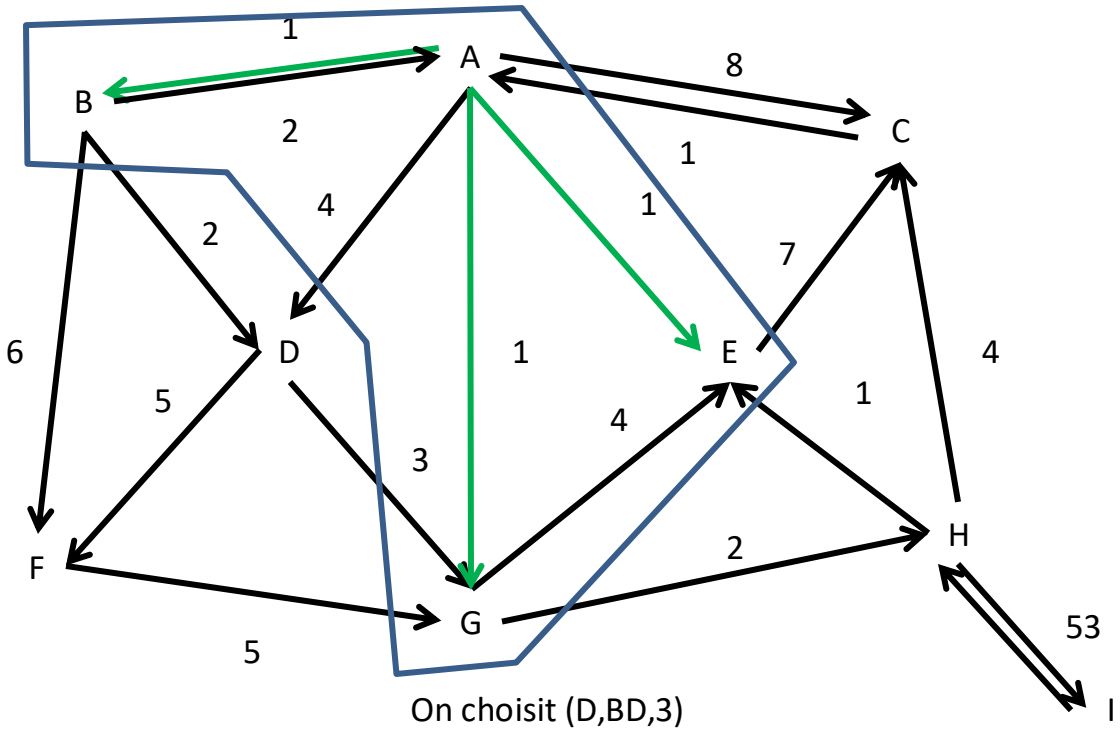
cocycle = {(C,AC,8), (D,BD,3), (E,AE,1), (H,GH,3), (F,BF,7)}




On choisit (E,AE,1)

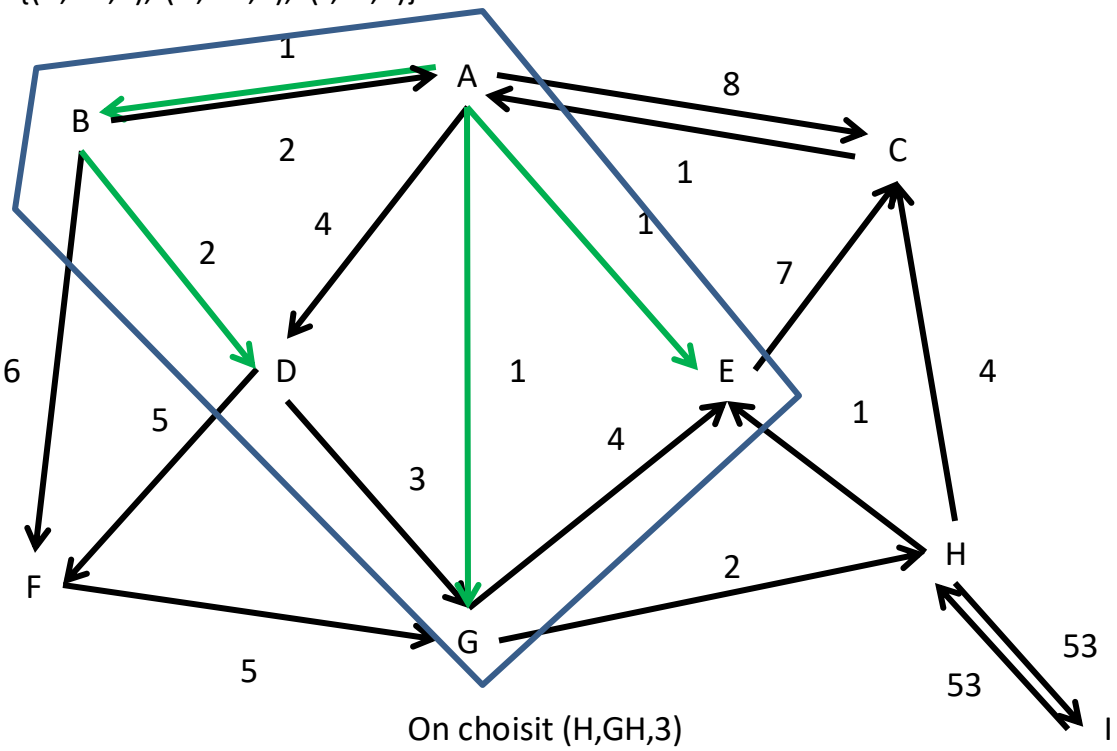
| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | ∞ | ∞ | 1 | ∞ | 1 | ∞ | ∞ |

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



| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | ∞ | 3 | 1 | ∞ | 1 | ∞ | ∞ |

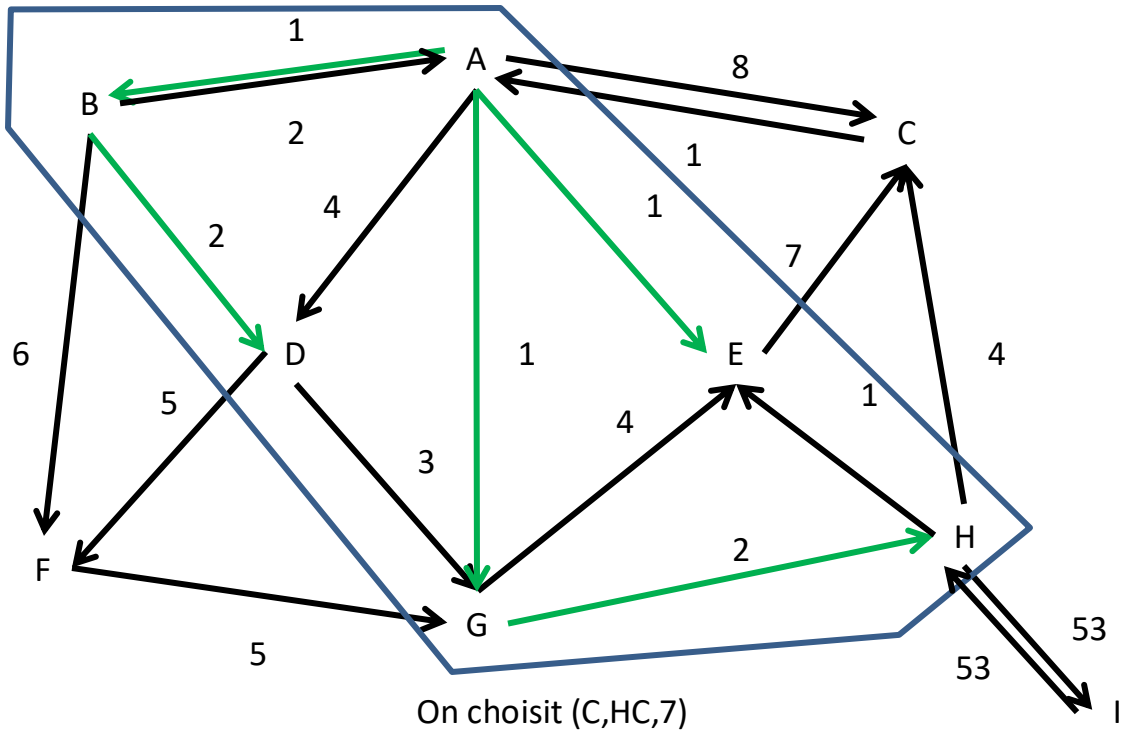
cocycle = {(C,AC,8), (H,GH,3), (F,BF,7)}




On choisit (H,GH,3)

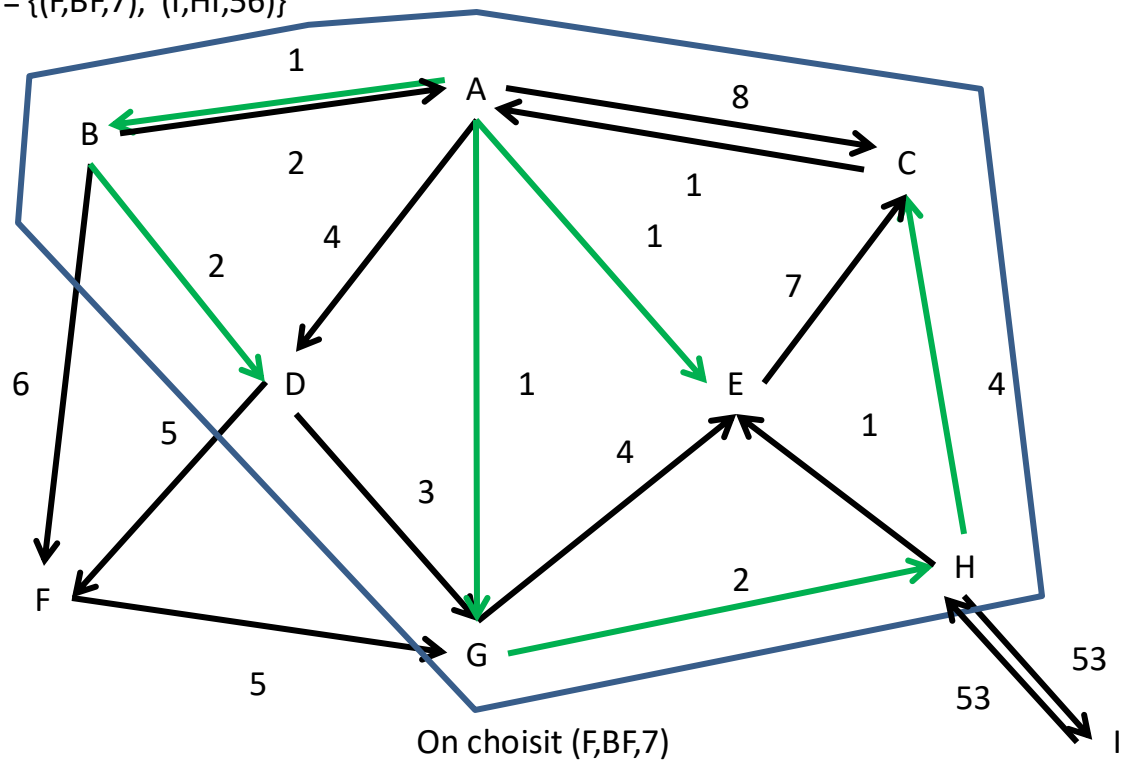
| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | ∞ | 3 | 1 | ∞ | 1 | 3 | ∞ |

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



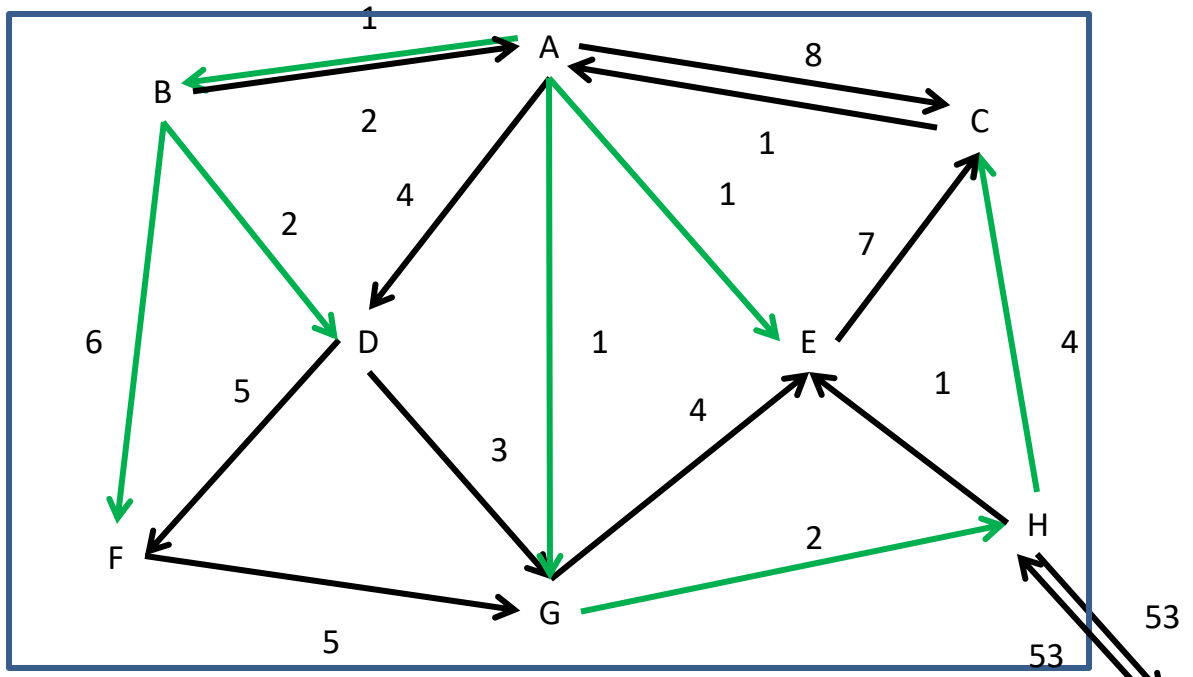
| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 7 | 3 | 1 | ∞ | 1 | 3 | ∞ |

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



| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 7 | 3 | 1 | 7 | 1 | 3 | ∞ |

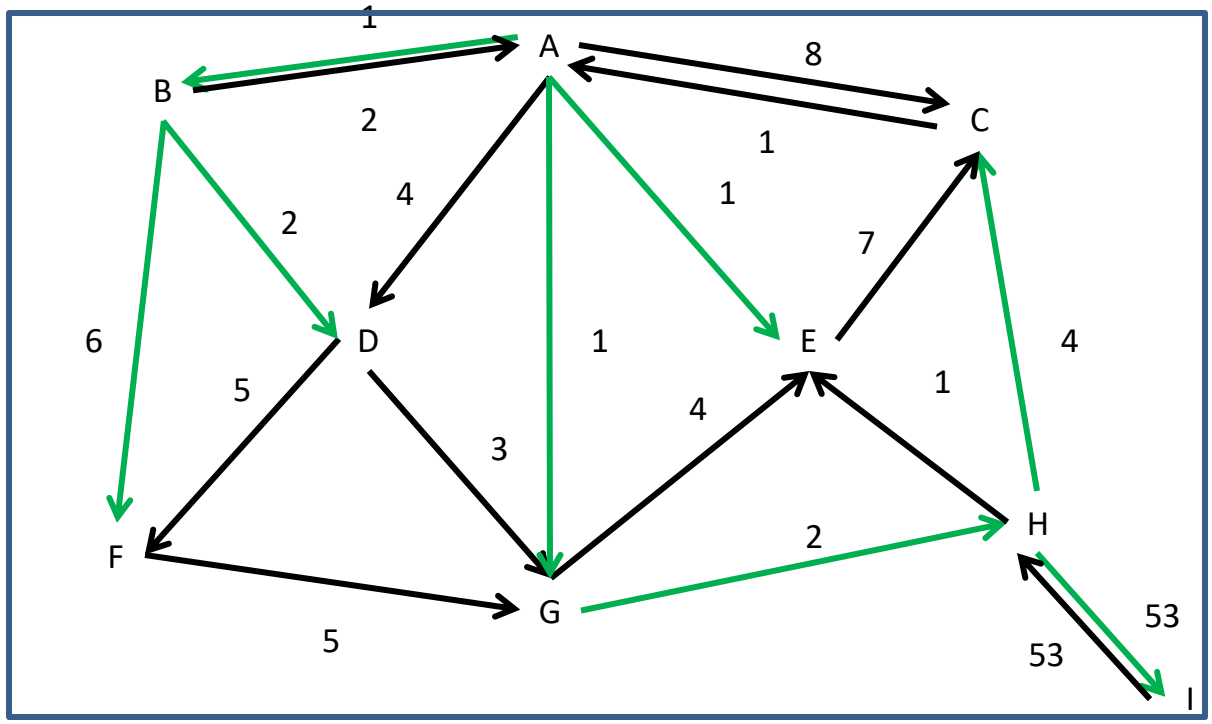
cocycle = {(I,HI,56)}




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

# Complexité

- $n^2$ pour une représentation par matrice
- m log n pour une représentation par liste d'adjacence

# C’est bien mais pour quoi faire ?

- On vient de calculer un arbre qui permet la diffusion de l’information de A (sur l’exemple) vers tous les autres nœud du réseau en suivant un chemin de poids minimal.

- Ce n’est pas ce que nous voulions

# C’est bien mais pour quoi faire ?

- Un simple parcours de l’arbre construit permet d’obtenir la table de routage du nœud A.

| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| - | B | G | B | E | B | G | G | G |

# Avantages/inconvénients

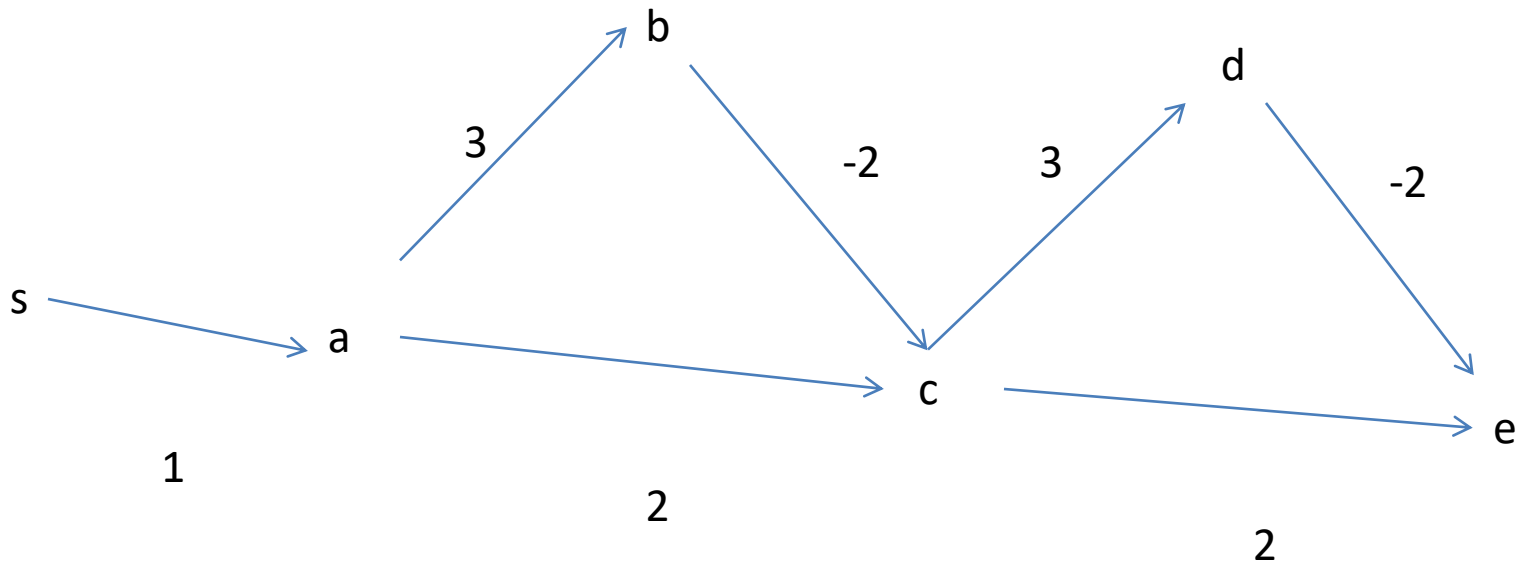
- On calcule les chemins de poids minimaux (latence, sauts) lorsque l'opérateur de calcul du poids d'un chemin est un +

- Algorithme fortement centralisé (recherche d'un minimum global) donc inapproprié pour faire des calculs à la volée

- Que se passe-t-il pour d'autres opérateurs (min, max, *, …)

# SECOND ALGORITHME : BELLMAN-FORD

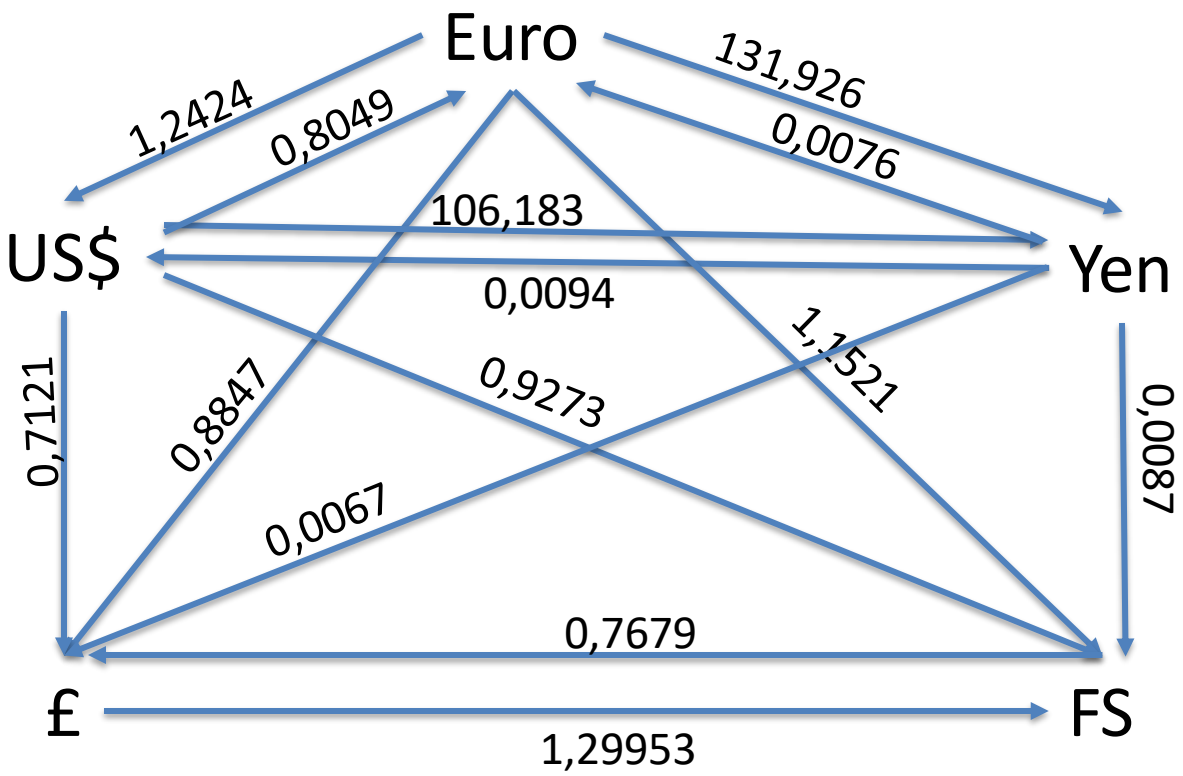
# Arcs de pondération négative ?



# Pondérations négatives ? Un petit exemple

- Problème d'arbitrage de change :
  - Données :
    - GTC : un graphe de taux de change
    - S : une monnaie de départ
    - D : Une monnaie d'arrivée
  - Question : quel chemin de S à D me permet d'optimiser la somme obtenue dans la devise D ?

# Pondérations négatives ?
# Un petit exemple



# Pondérations négatives ? Un petit exemple

- Pour 100 Euros j'aurai 115,21 francs suisses
- Pour 100 Euros j'aurai 13192,6 Yen
- Pour 13192,6 Yen j'aurai 13192,6 * 0,0094 $
- Pour 13192,6 * 0,0094 $ j'aurai

  13192,6 * 0,0094 * 0,9273 Francs suisses
- Avons-nous :

  115,21 = 13192,6 * 0,0094 * 0,9273 ?

# Pondérations négatives ? Un petit exemple

- On cherche un chemin avec un poids mais ce poids est calculé par le produit des poids des arcs (au lieu de la somme).
- On cherche un chemin de poids maximal (au lieu de minimal.

# Pondérations négatives ? Un petit exemple

- Remarque 1 : $0 < x < y \Rightarrow 0 < 1/y < 1/x$
- Remarque 2 : $c = a*b \Rightarrow 1/c = 1/a * 1/b$
- Remarque 3 : $\log(a*b*c) = \log(a)+\log(b)+\log(c)$
- Remarque 4 : log est une fonction croissante sur l'intervalle $]0, +\infty[$.

# Pondérations négatives ? Un petit exemple

- On souhaite un chemin de poids maximal.
- Pour un chemin $\mu=x_1x_2x_3\ldots x_k$ de notre graphe des taux de change le poids de ce chemin est : $V(\mu)=V(x_1x_2)*V(x_2x_3)*\ldots*V(x_{k-1}x_k)$
- Remarque si touts les poids sont strictement positif $V(x_1x_2)*V(x_2x_3)*\ldots*V(x_{k-1}x_k)$ est maximal si et ssi $(1/V(x_1x_2))*(1/V(x_2x_3))*\ldots*(1/V(x_{k-1}x_k))$ est minimal.

# Pondérations négatives ? Un petit exemple

Nous créons une nouvelle pondération de nos arcs :

- V’(xy) = 1/V(xy)

Cette nouvelle pondération nous permet de transformer la recherche d’un chemin de poids maximal (en utilisant V) en la recherche d’un chemin de poids minimal (en utilisant V’)

# Pondérations négatives ? Un petit exemple

Reste le problème de la multiplication que nous allons résoudre grâce à la remarque 3 : log(a*b*c) = log(a)+log(b)+log(c)

Posons V’’(xy) = log (V’(xy))

Cette nouvelle pondération nous permet de transformer la recherche d’un chemin de poids minimal (en utilisant *) en la recherche d’un chemin de poids minimal (en utilisant +)

# Pondérations négatives ? Un petit exemple

- On transforme donc toutes les pondérations :
  - V’’(uv) = log (1/V(uv)) = - log (V(uv))

- On cherche alors un chemin de poids minimal (au sens que nous avons donné dans la première partie) qui contiendra des arcs de pondérations négatives. Car - log (V(uv)) < 0 dès que V(uv) > 1

# Algorithme de Bellman-Ford

- Cet algorithme construit les chemins de poids minimaux en acceptant les arcs de pondération négative.

- Il faut donc accepter de reconsidérer des sommets pour lesquels un chemin à été calculé.

# Principes de Bellman-Ford

- Construire les chemins à partir du sommet s par ordre de longueur croissante.
- Retenir pour chaque sommet un chemin qui réalise la valeur minimale
- Initialement on sait construire un chemin de longueur 0 allant de s à s de poids nul.

# Principes de Bellman-Ford

- Considérons que nous avons un tableau D (indicé par les sommets) de valeurs

- A l'étape i : $D_i[x]$ contient le poids minimal d'un chemin de longueur au plus i ayant pour origine s et pour extrémité x. Si un tel chemin n'existe pas $D[x] = \infty$
- Construisons $D_{i+1}$ (D à l'étape i+1)

# Principes de Bellman-Ford

- Un chemin de poids minimal de longueur au plus i+1 se décompose en un chemin de poids minimal de longueur au plus i (dans $D_i$) et un arc qui prolonge ce chemin.
- $D_{i+1}[x] = \text{Min}(D_i[x], \text{Min}_{y \in Pred(x)}(D_i[y]+V(yx)))$

# Condition finale

- Avoir construit tous les chemins élémentaires de poids potentiellement minimal. Or S'il n'y a pas de circuits de poids négatifs un chemin élémentaire est au plus de longueur |X| - 1.

# Algo BF (entête)

- Algo BF
- Données :
  - G = (X,U,V) un graphe pondéré
  - s un sommet de G
- Résultats :
  - GCPM = (X, U', V) un graphe pondéré
  - D : tableau de nombres indicé par les sommets
- Variables :
  - Origine : tableau de sommets indicé par les sommets

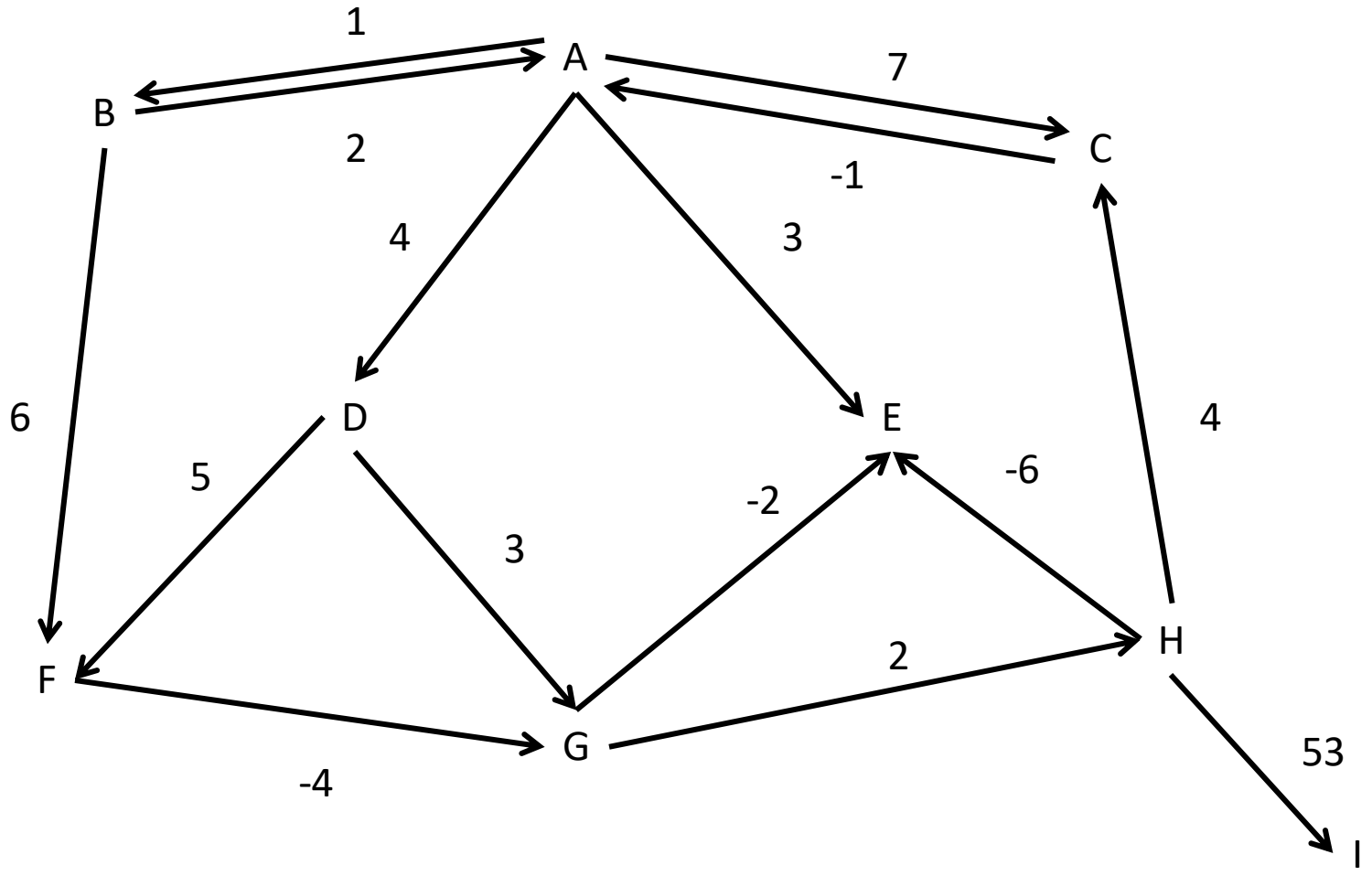
# Algo BF (code 1)

- Début
  - Pour chaque sommet t de X faire
    - D[t] ← ∞; Origine [t] ← t
  - FinPour
  - D[s] ← 0; U' ← {};

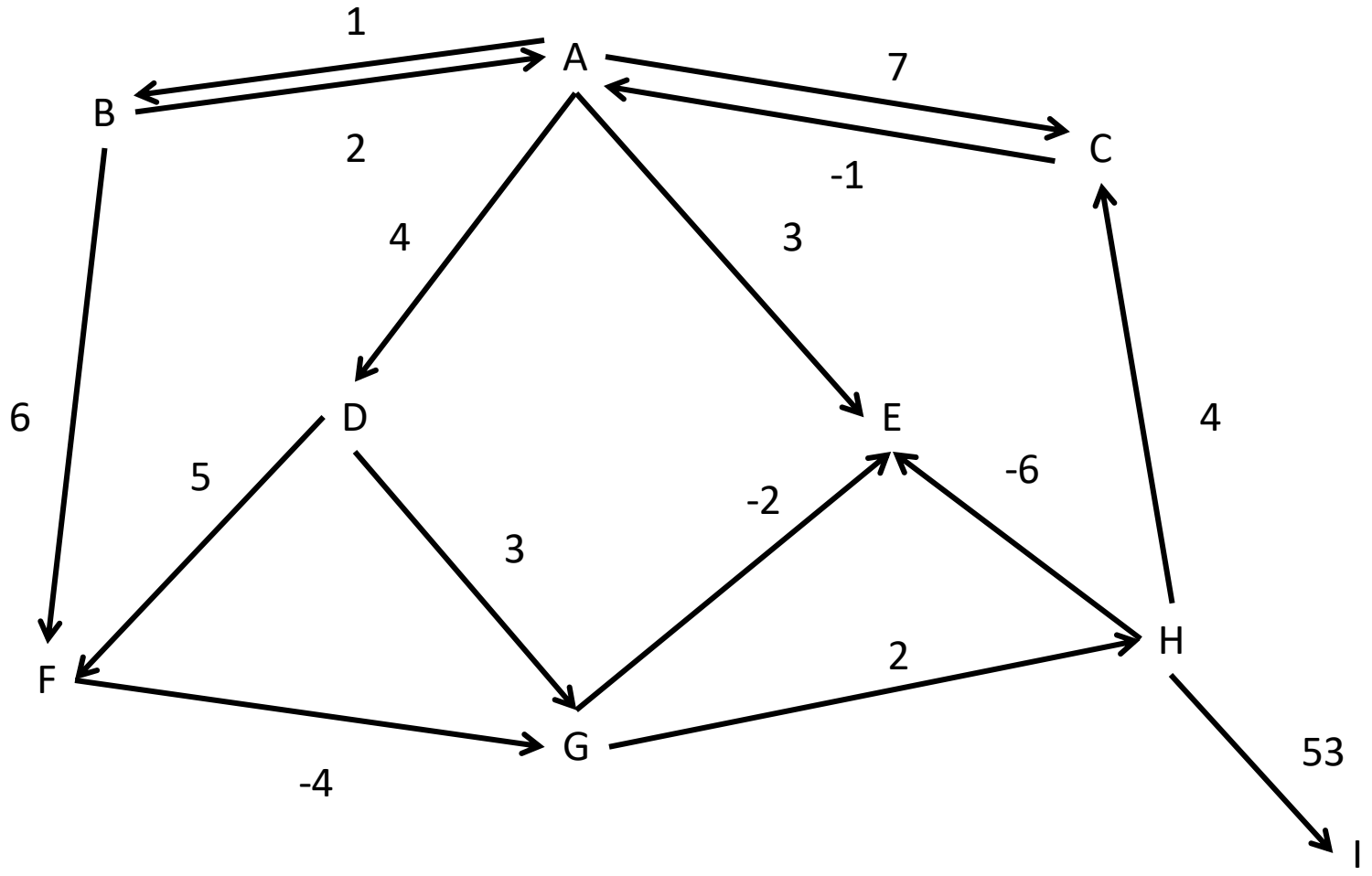
# Algo BF (code 2)

- Pour i allant de 1 à |X|- 1 faire
  - Pour chaque sommet u ϵ X faire
    - Si D[u] ≠ ∞ alors
      - » Pour chaque y ϵ Succ(u) faire
        - Si D[y] > D[u] + v(uy) alors
          - D[y] ← D[u] + v(uy); Origine [y] ← u;
        - FinSi
      - » FinPour
    - FinSi
  - FinPour
- FinPour
- Pour chaque sommet u ϵ X-{s} faire
  - Insérer l'arc (Origine[u], u) dans U'
- FinPour
- FinCode

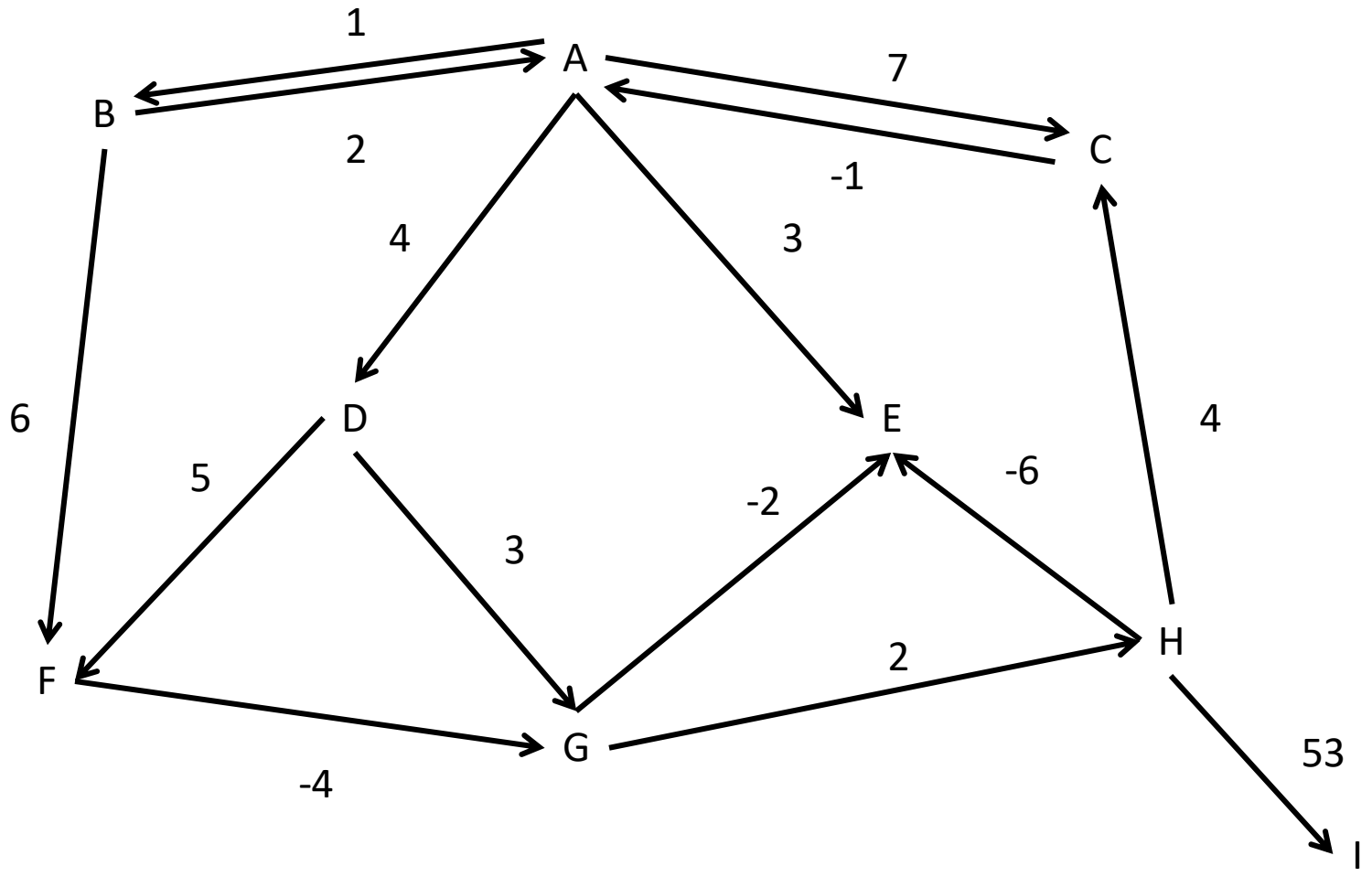
# Exemple



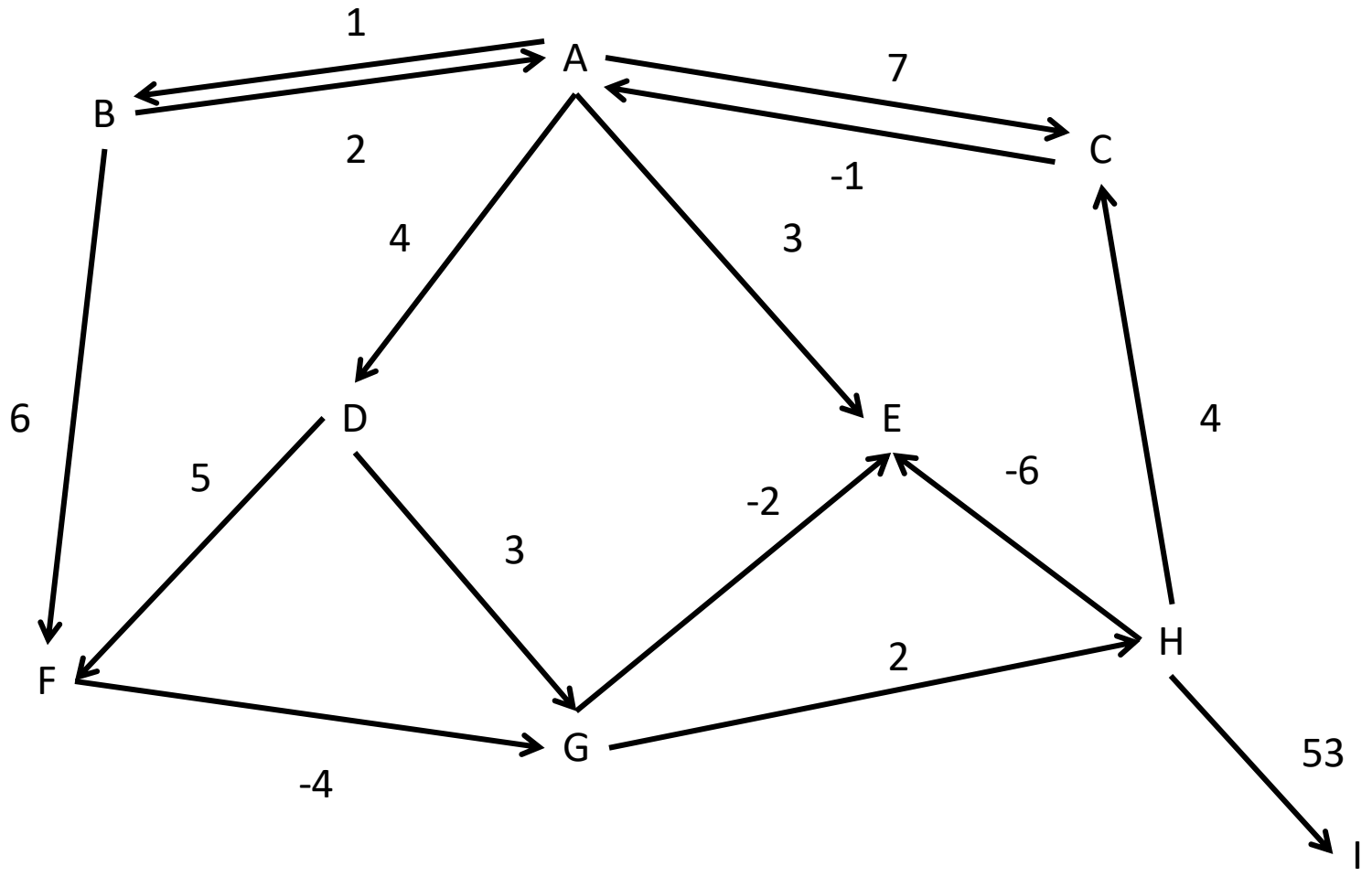
| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| A | B | C | D | E | F | G | H | I |



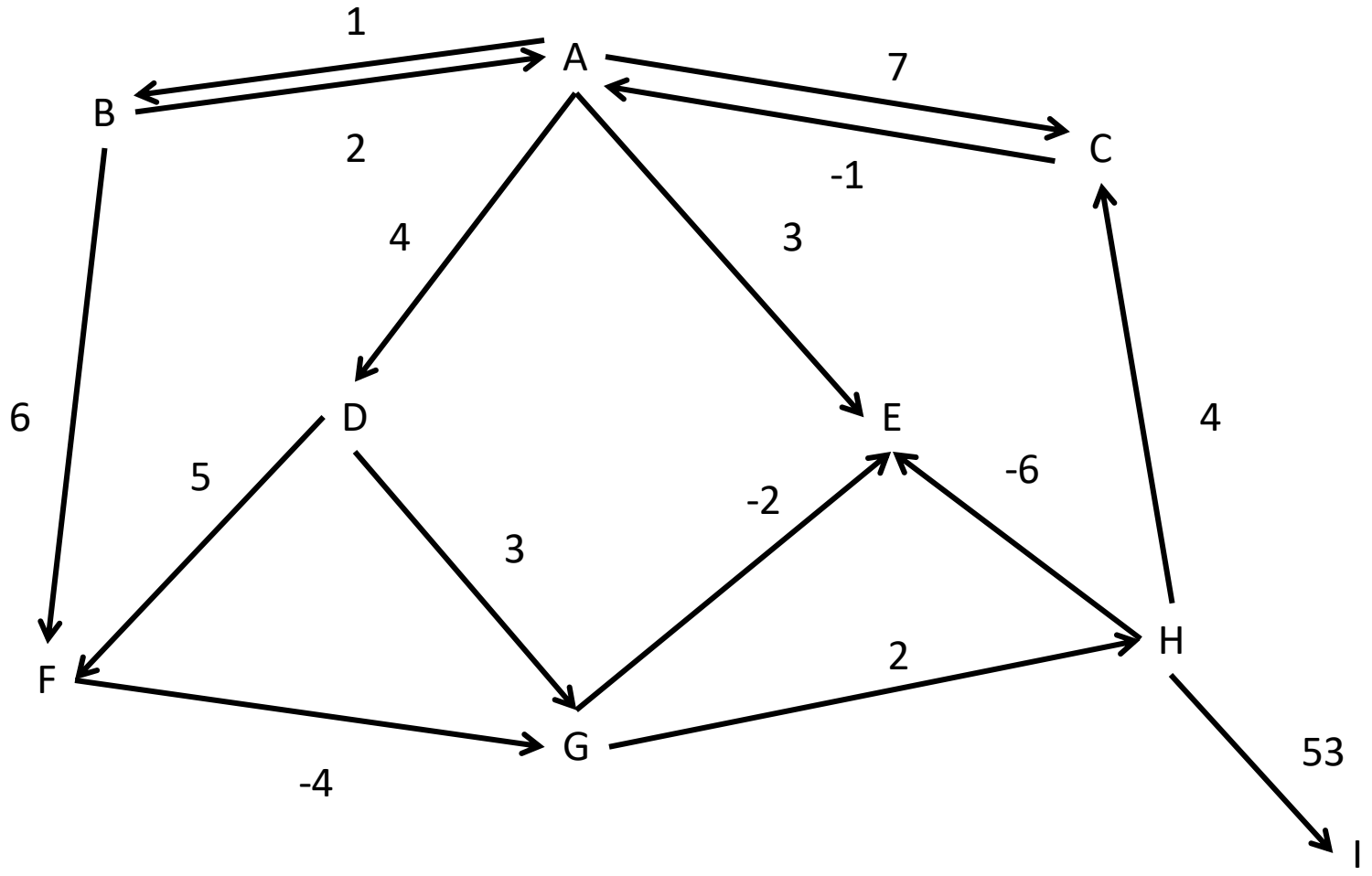
| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 7 | 4 | 3 | ∞ | ∞ | ∞ | ∞ |
| A | A | A | A | A | F | G | H | I |



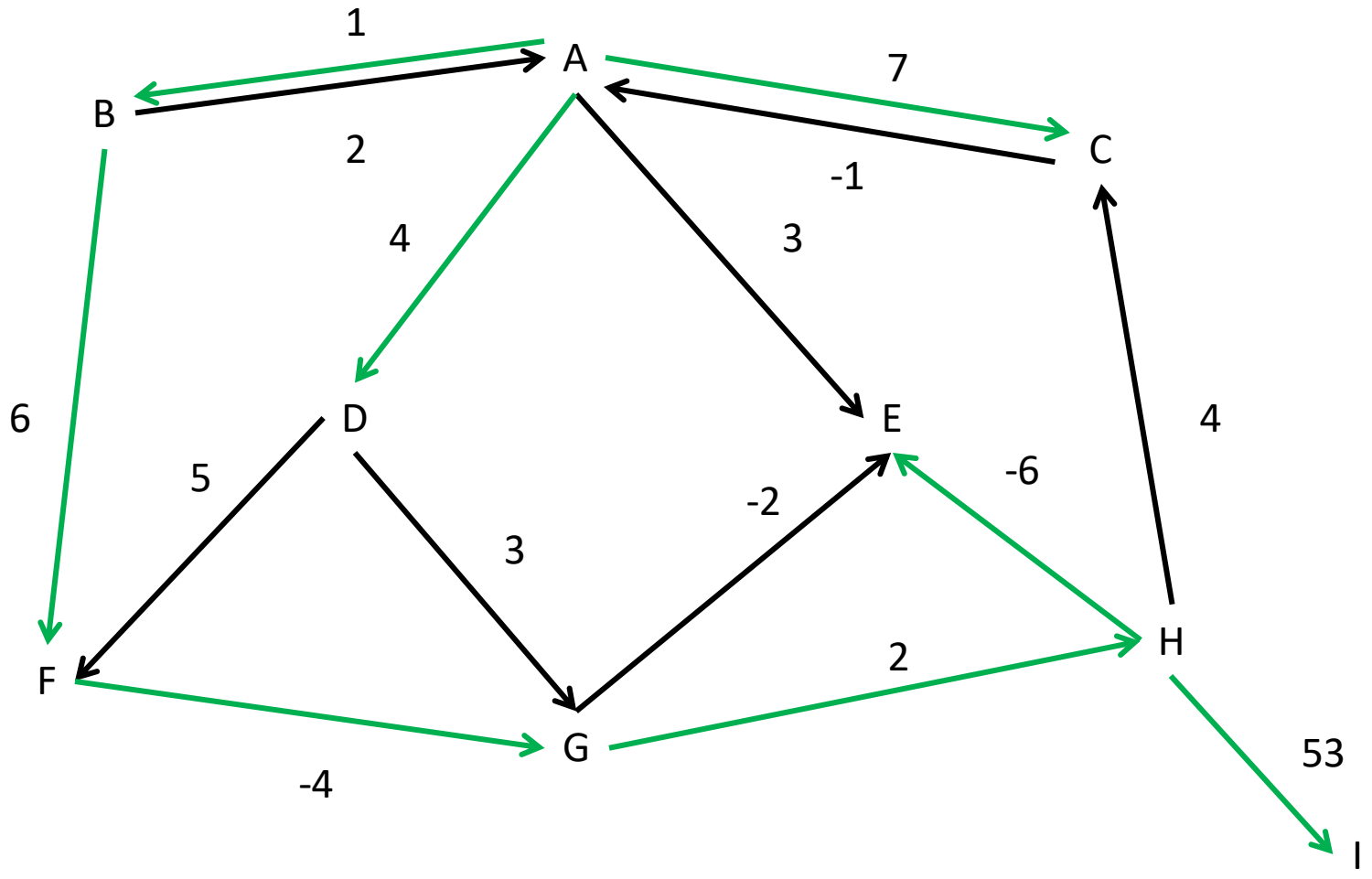
| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 7 | 4 | 3 | 7 | 7 | ∞ | ∞ |
| A | A | A | A | A | B | D | H | I |



| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 7 | 4 | -1 | 7 | 3 | 5 | 58 |
| A | A | A | A | H | B | F | G | H |



| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 7 | 4 | -1 | 7 | 3 | 5 | 58 |
| A | A | A | A | H | B | F | G | H |



# Propriété

- 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 $x_{k-1}$.

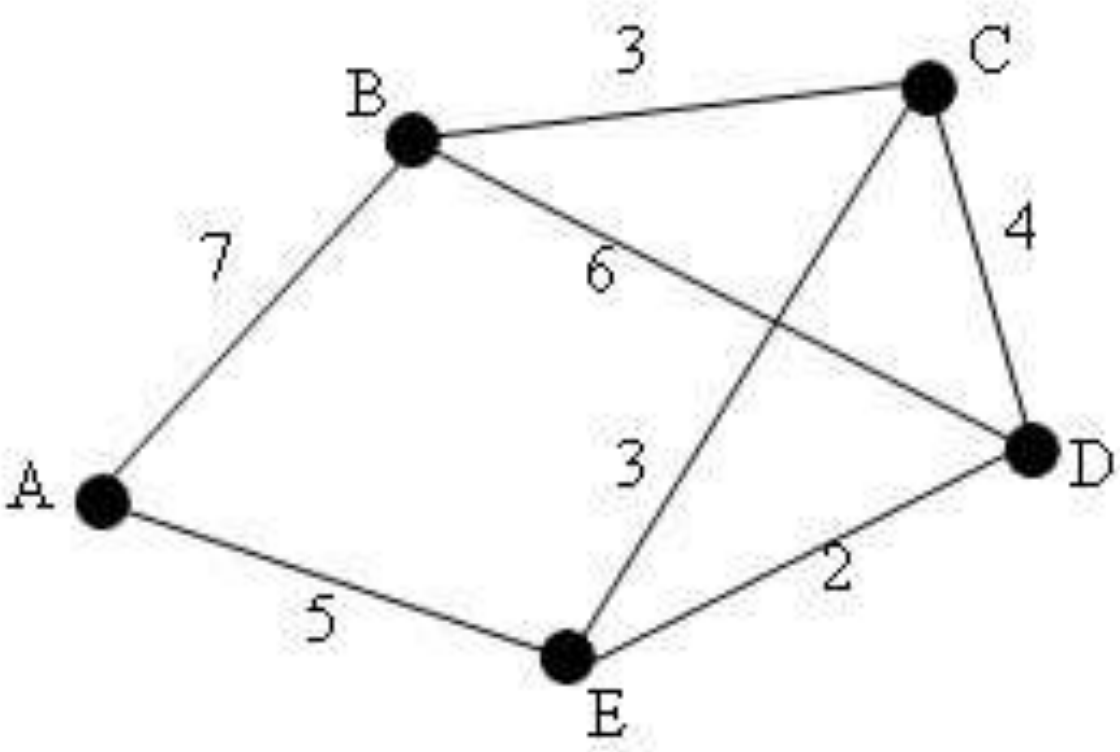
- Bellman-Ford exploite cette propriété en effet à l'étape k cet algorithme construit les chemins de poids minimaux de longueur inférieure ou égale à k.

# Complexité

- La complexité de cet algorithme est :
  - $O(n^3)$ pour une représentation par matrice
  - O(nm) pour une représentation par listes d'adjacence

# ALGORITHME DE PLUS COURT CHEMIN ENTRE TOUT COUPLE DE SOMMETS

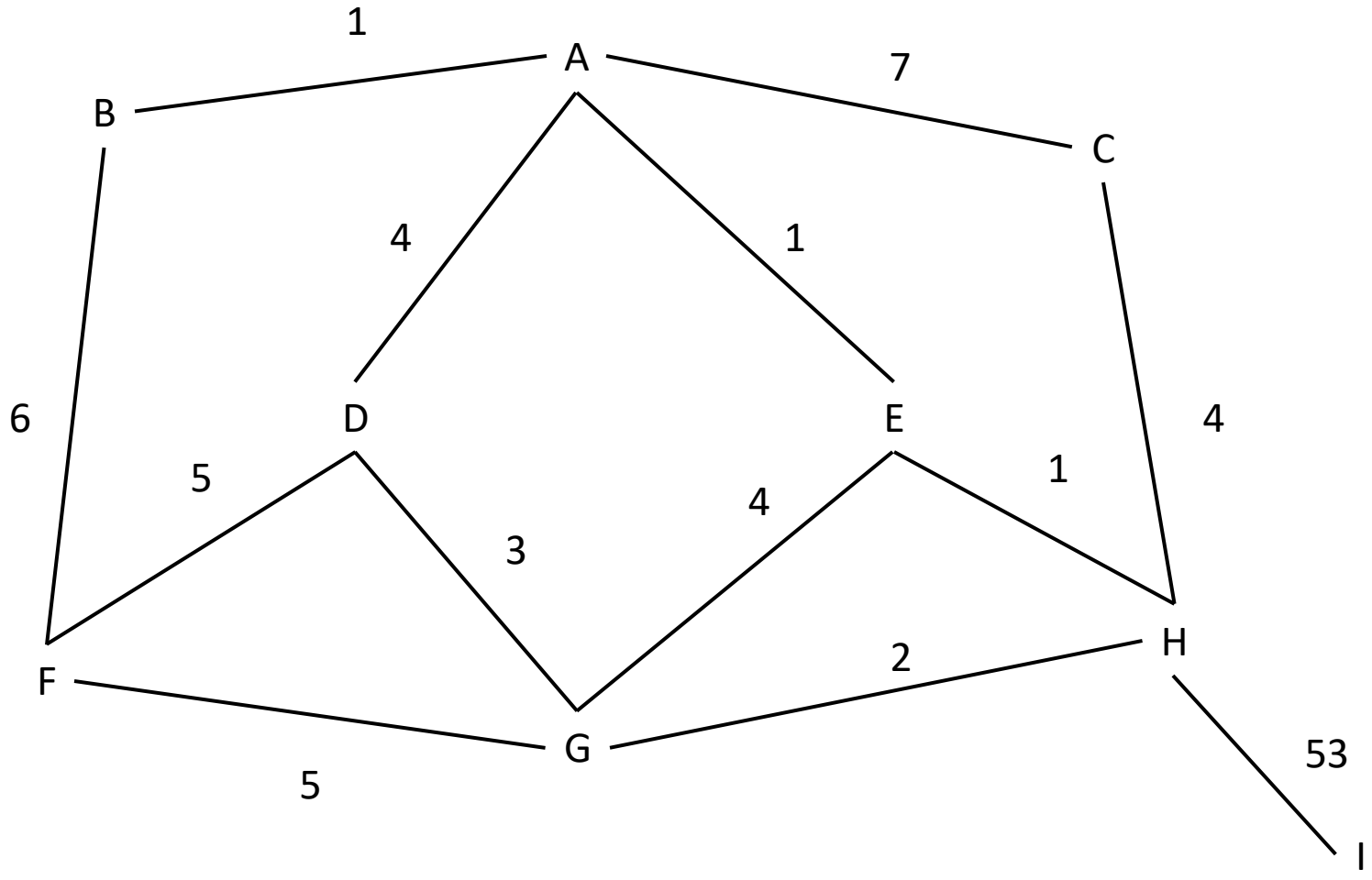
# Exemple



# Représentation

- La représentation en algorithmique centralisée se fait par une matrice M carrée indicée par les éléments de X
  - M(x,x) = 0
  - M(x,y) = V(x,y) si (x,y) ∈ U
  - M(x,y) = ∞ si l'arête (x,y) n'est pas dans U

# Exemple



# Représentation par matrice

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

# But

- On souhaite connaitre le plus court chemin entre tous les couples de sommets du graphe.

- Utile pour l'acheminement d'information d'une source vers une destination dans un réseau

# Représentation Table

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| A | (0,_) | (1,B) | (7,C) | (4,D) | (1,E) | (∞,_) | (∞,_) | (∞,_) | (∞,_) |
| B | (1,A) | (0,_) | (∞,_) | (∞,_) | (∞,_) | (6,F) | (∞,_) | (∞,_) | (∞,_) |
| C | (7,A) | (∞,_) | (0,_) | (∞,_) | (∞,_) | (∞,_) | (∞,_) | (4,H) | (∞,_) |
| D | (4,A) | (∞,_) | (∞,_) | (0,_) | (∞,_) | (5,F) | (3,G) | (∞,_) | (∞,_) |
| E | (1,A) | (∞,_) | (∞,_) | (∞,_) | (0,_) | (∞,_) | (4,G) | (1,H) | (∞,_) |
| F | (∞,_) | (6,B) | (∞,_) | (5,D) | (∞,_) | (0,_) | (5,G) | (∞,_) | (∞,_) |
| G | (∞,_) | (∞,_) | (∞,_) | (3,D) | (4,E) | (5,F) | (0,_) | (2,H) | (∞,_) |
| H | (∞,_) | (∞,_) | (4,C) | (∞,_) | (1,E) | (∞,_) | (2,G) | (0,_) | 53 |
| I | (∞,_) | (∞,_) | (∞,_) | (∞,_) | (∞,_) | (∞,_) | (∞,_) | (53,H) | (0,_) |

# Par opérations sur les matrices

- Pour chaque couple de sommet (x,y) on calcule la valeur

$$v2(xy) = Min_{z \in X}(v(xz)+v(zy))$$

- On obtient ainsi le chemin de poids minimal de x à y dont la longueur est au plus 2. Si on souhaite les chemins de longueur au plus 3 on calcule la valeur :

$$v3(xy) = Min_{z \in X}(v2(xz)+v(zy))$$

# Opération de Base (Entête)

- Algorithme OpMat
- Données :
  - M, N : deux matrices d'entiers indicées par les sommets
- Résultat :
  - Res : Une matrice d'entier indicées par les sommets

# Opération de Base (code)

- DébutCode
  - Pour tout x ϵ X faire
    - Pour tout y ϵ X faire
      - Res[x,y] ← M[x,y] +N[y,y]
      - Pour tout z ϵ X faire
        - » Res[x,y] ← Min(Res[x,y], M[x,z] + N[z,y])
      - FinPour
    - FinPour
  - FinPour
- FinCode

# Complexité

- $O(n^3)$ opérations

# Algorithme des plus courts chemins : opération matrice (entête)

- Algorithme PCCOMOM
  - Donnée :
    - M la matrice d'un graphe pondéré
  - Résultat :
    - Res une matrice
  - Variables
    - i entier
    - Inter, Inter2 : deux matrices

# Algorithme des plus courts chemins : opération matrice (code)

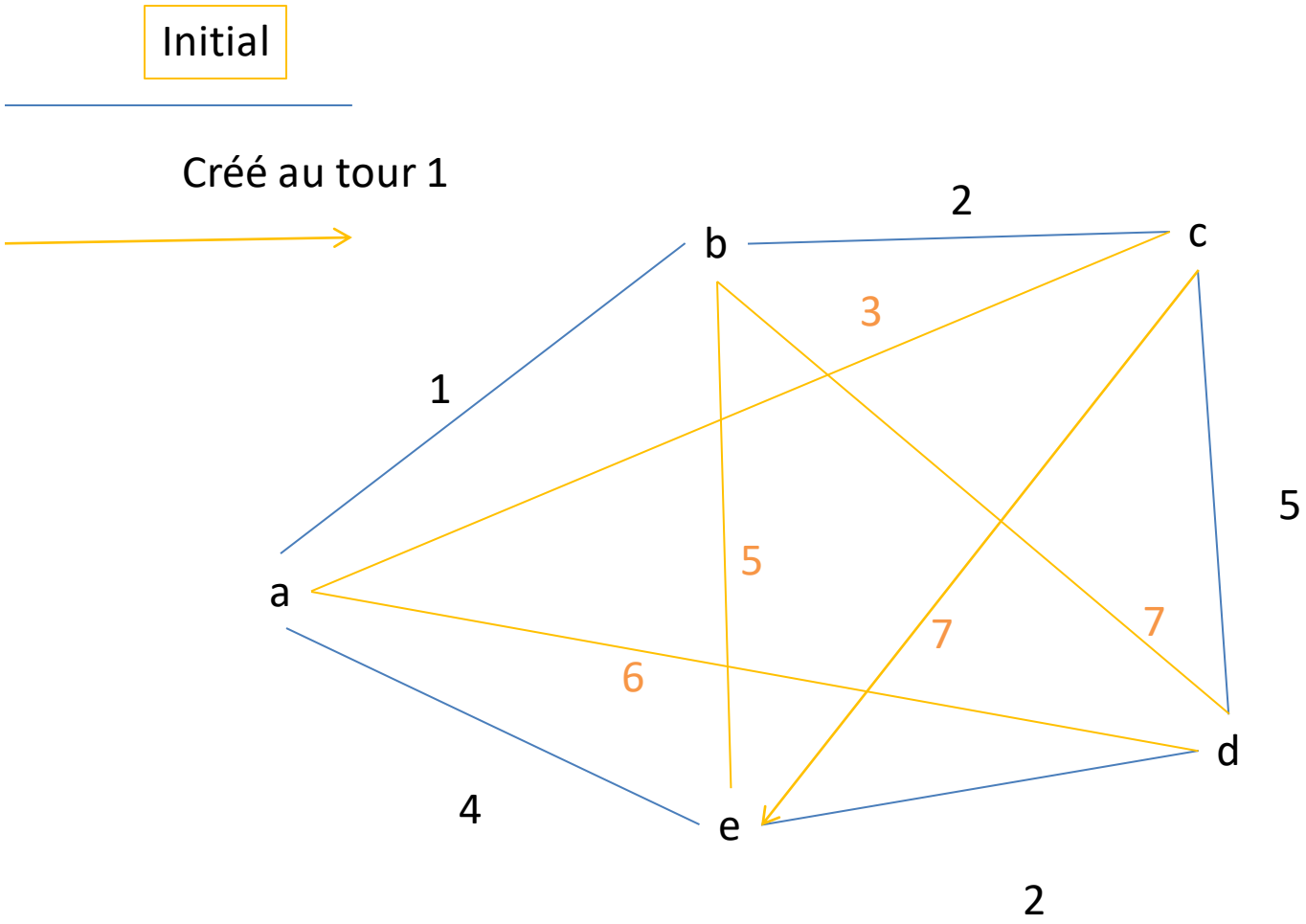
- DébutCode
  - Inter ← M; Inter2← M;// par duplication
  - Pour tout x ϵ X faire
    - Inter[x,x]← 0; Inter2[x,x] ← 0;
  - FinPour
  - Pour i ← 2 à n faire
    - OpMat(Inter, Inter2, Res); Inter ← Res;//Duplication
  - FinPour
- FinCode

# Propriétés

- A la fin du kème tour de boucle, Inter[u,v] contient le poids d'un chemin de u à v de poids minimal et de longueur au plus k+1. C'est l'invariant de l'algorithme

- Complexité $O(n^4)$

# Exemple



# Question

- Comment adapter cet algorithme en système distribué ?

# Répartition des données

- Chaque nœud contiendra une ligne de la matrice le nœud i contiendra la ligne i de la matrice.
- Ci-dessous la table du nœud A Les liens sont étiqueté par le nom du nœud extrémité

| TR | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| A | (0,_) | (1,B) | (7,C) | (4,D) | (1,E) | (∞,_) | (∞,_) | (∞,_) | (∞,_) |

# Hypothèses

- On considère que tous les nœuds connaissent les identités de tous les nœuds du réseau.
- C l'ensemble de tous les canaux d'un nœud
- Ces canaux sont numérotés de 0 à δ-1
- Pour chaque canal i : Coût(i) donne le poids du canal dans le chemin. Les poids sont tous positifs.

# Constante et Variables

- Constante : IdLoc identité du nœud
- Variables
- TR un tableau de couple (poids,canal) indicé par les identités du réseau :
  - TR[IdLoc] ← (0,_)
  - TR[Id] ← (∞,_) quand Id ≠ IdLoc
- Mess(Identité,Poids) un message

# Action1

Spontanée (dois être effectuer par tous avant toute réception de message)

Envoyer Mess(IdLoc, 0) sur C

FinSpontanée

# Action2

A la réception de Mess(Id,p) sur le canal i

(ap,ac) ← TR[Id]

Si ap > p + Coût(i) alors

TR[Id] ← (p+Coût(i),i)

Envoyer Mess(TR[Id]) sur C // tous les voisins

FinSi

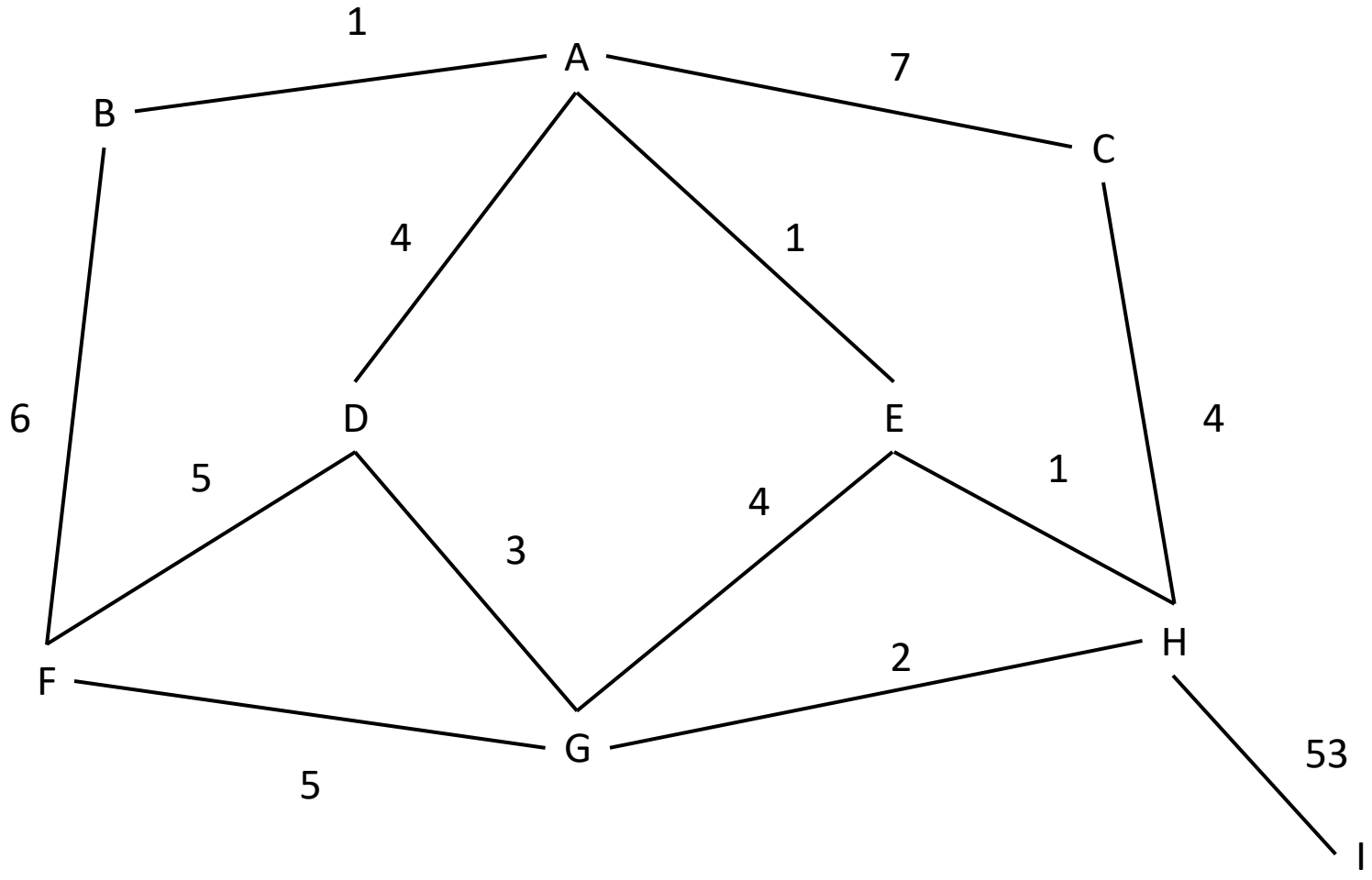
FinAction

# Exercice

- Donnez une exécution de cet algorithme sur le graphe donné page suivante

- Calculez la complexité Chandy-Misra : « Distributed computation on graphs: shortest path algorithms »

# Exemple



# Sujets de réflexion

- Comment faire en sorte que les chemins soient construits à la manière de BF, c'est-à-dire les chemins de longueur 0 puis 1 puis 2…

- Comment détecté la terminaison ?

# Réduire la taille de la table de routage

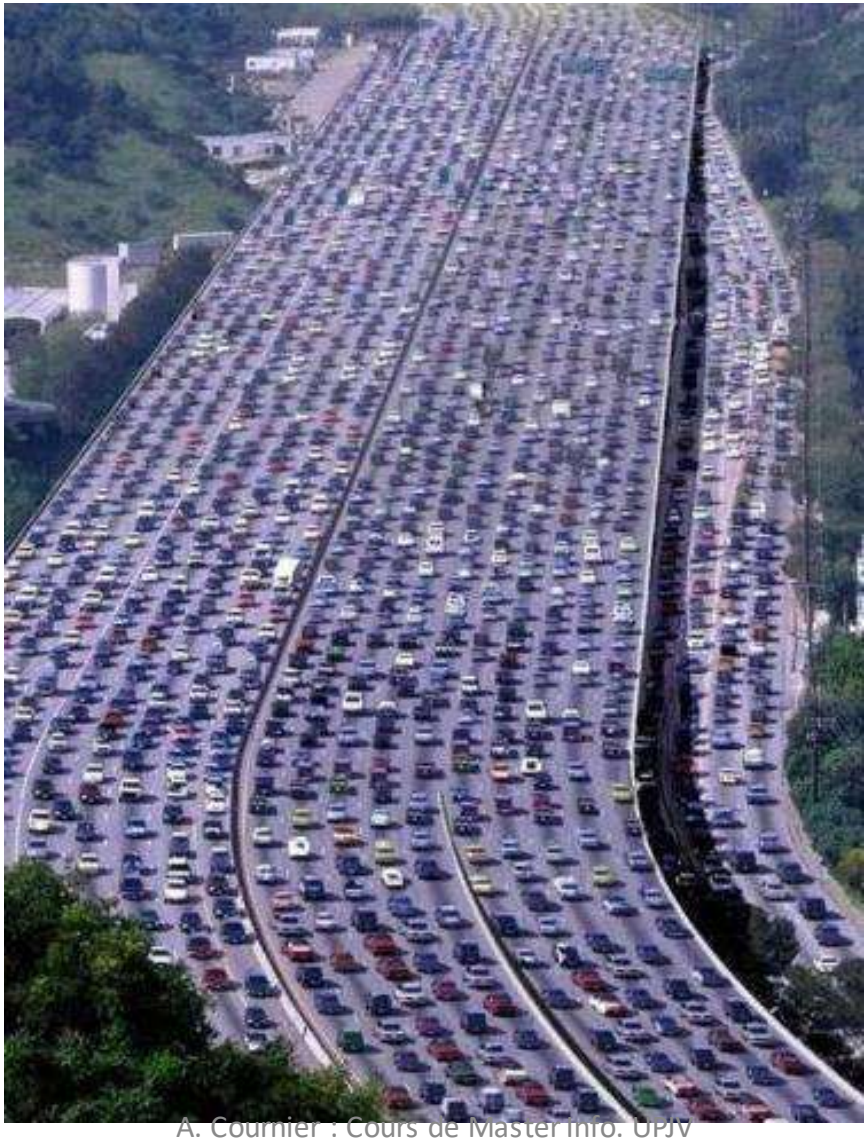
- Jusqu'à présent un table de routage code sur chaque nœud tous les couples (Id, canal) utiles pour acheminer les messages. Nous avons donc une taille de la table de routage égale à : Θ(nlog(w)) avec
  - n le nombre de nœud
  - w la valeur maximale d'une identité

# Réduire la taille de la table de routage

- Une solution consiste à utiliser le routage par intervalles.
  - [i,j] = {i, i+1, i+2, …, j-1, j} quand i < j
  - [i,j] = {i, i+1, …, n-1, 0, 1, …, j} dans le cas contraire.

# RECHERCHE DE CHEMINS ALTERNATIFS

# Et pour les embouteillages ?



# Les solutions radicales ?

# Le routage alternatif

# Définition

- Soit a et b deux nœuds du réseau. Soit μ un chemin de a vers b. On dis que μ' est un chemin alternatif ssi :
  - μ' est un chemin de a vers b.
  - Le chemin μ' est indépendant de μ.

- L'indépendance peut se juger par une indépendance des arêtes utilisées par ces deux chemins ou par l' indépendance des sommets utilisées par ces deux chemins.

# Utilité

- Résistance aux pannes du réseau :
  - Coupure d'un lien
  - Indisponibilité d'un nœud

- Sécurité :
  - Résistance à l'écoute
  - Augmenter la bande passante disponible entre les deux nœuds

# Des solutions plus douce

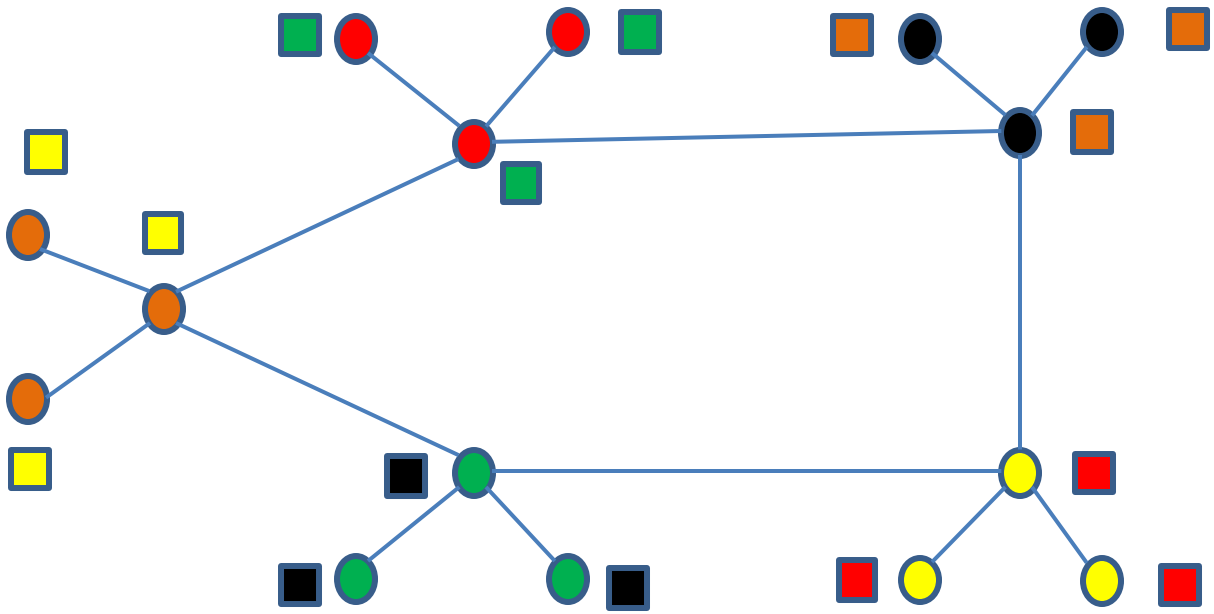
- Peut-on garantir que cela n’arrivera pas ?

# Interblocage

- Est-il possible qu'une pénurie de resources rende l'acheminement des messages impossibles ?

- Peut-on prévenir ces difficultés ?

# Interblocage



# Que faire si le réseau est dynamique ?