

Réseaux informatiques

RXI

Chapitre 6

Routage IP

2^{ème} partie



Objectifs d'apprentissage

- Savoir expliquer le processus d'acheminement d'un paquet sur un routeur
- Savoir expliquer la différence entre remise directe et remise indirecte ainsi que l'utilisation des adresses IP et MAC dans les deux cas
- Pour un réseau donné, savoir identifier les routes nécessaires sur chacun des routeurs
- Savoir utiliser l'algorithme de Dijkstra pour calculer le plus court chemin
- Pour un réseau donné, savoir appliquer manuellement la méthode de RIP pour calculer les routes
- Connaître la distinction entre les deux niveaux de routage



Routage dynamique

- Nécessite un protocole de routage sur le routeur qui
 - communique avec les autres routeurs
 - remplit la table de routage du routeur
 - S'adapte aux changements (panne d'un lien, nouveau lien)
- Objectif des protocoles de routage
 - Trouver « le meilleur chemin » vers chaque destinataire
 - Différents métriques sont possibles
 - Nombre de sauts
 - Capacité des liens, délai, trafic, disponibilité ...



Le plus court chemin

Algorithme de Dijkstra

0. Préparation:

- Représenter le réseau par un graphe
- Pondérer chaque arête k par un coût p_k

1. Marquer chaque nœud par un doublet (C_i, N_x)

C_i : Distance totale de la source

N_x : Nœud précédent (pour reconstruire le chemin)

Doublet de chaque nœud initialisé à $(\infty, -)$

à l'exception du nœud d'origine initialisé à $(0, -)$

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'

3. Calculer les coûts des chemins de tous les voisins N_j du nœud N_i : $C_j = C_i + p_k$

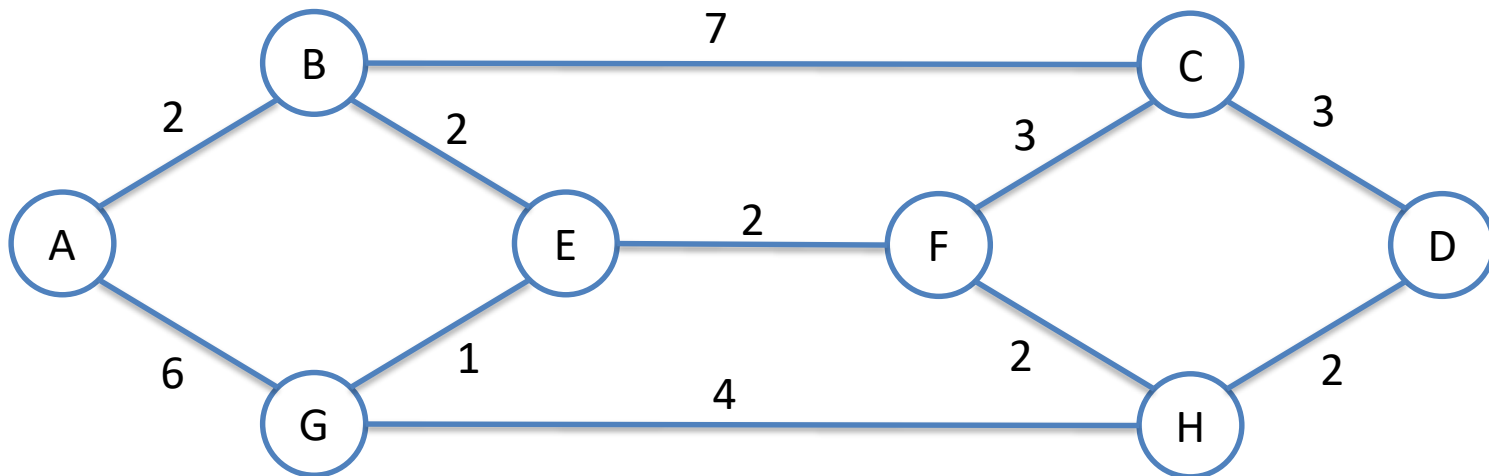
4. Si la nouvelle valeur C_j est plus petite que l'ancienne,
--> actualiser le doublet de N_j : (C_j, N_i)

5. Répéter 2 à 4 jusqu'à ce que la destination soit marquée 'permanent'



Algorithme de Dijkstra

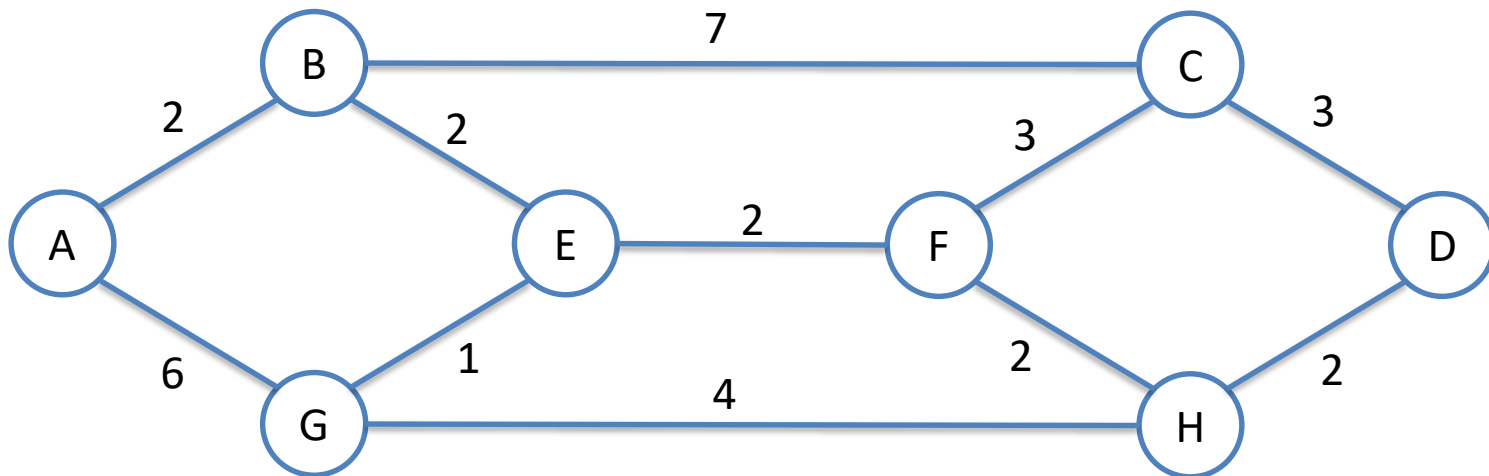
- Exemple : calcul de tous les chemins à partir de A



Algorithme de Dijkstra

0. Préparation:

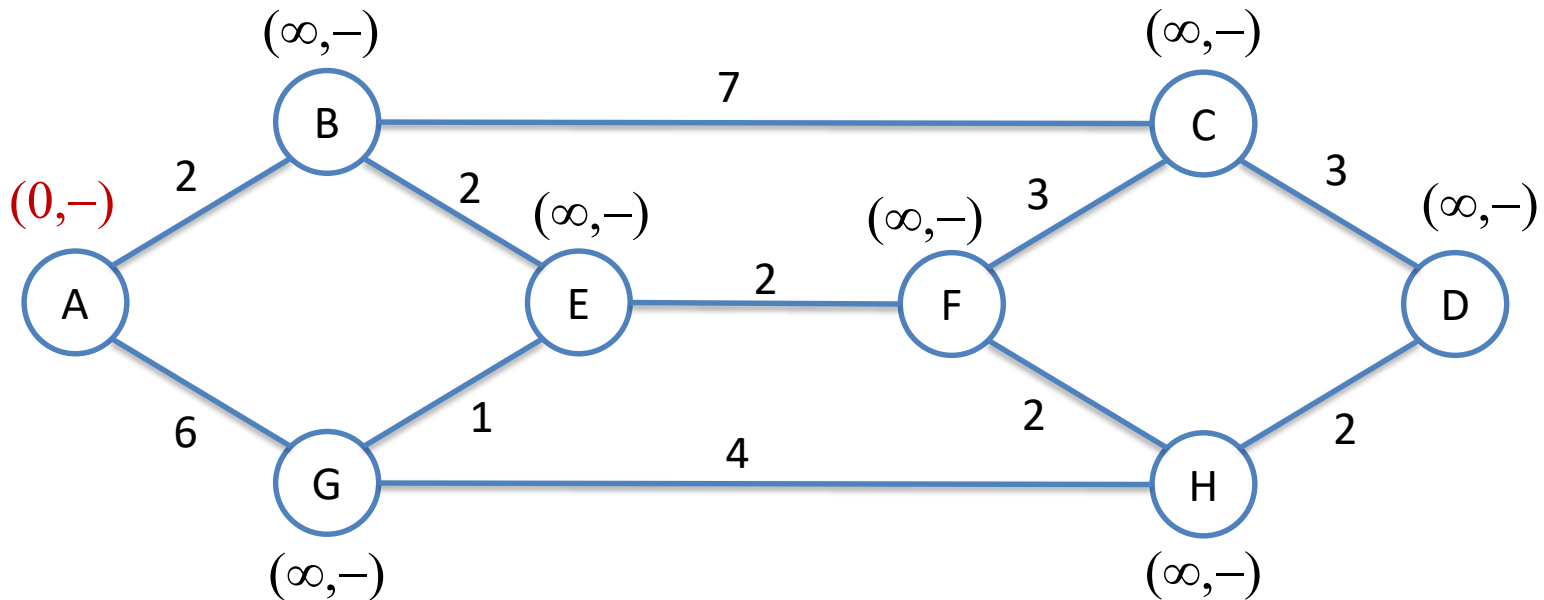
- représenter le réseau par un graphe
- pondérer chaque arête k par un coût p_k



Algorithme de Dijkstra

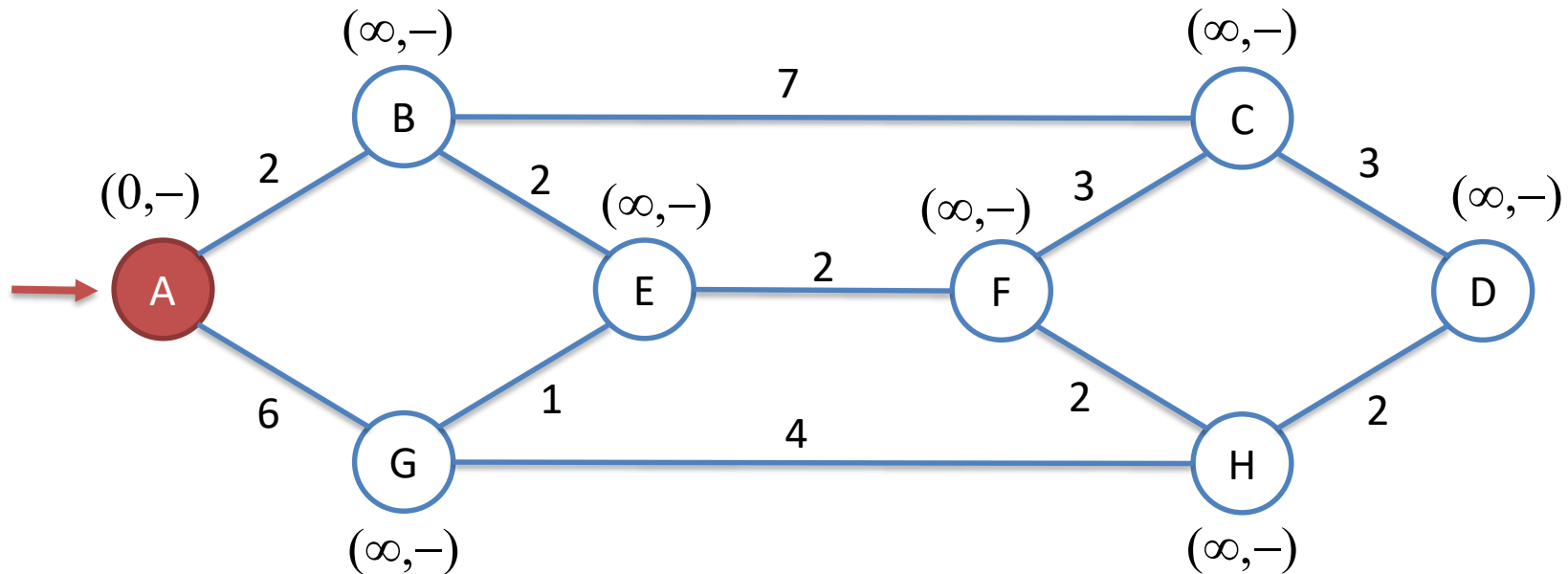
1. Initialisation :

Initialiser le doublet de chaque nœud à $(\infty, -)$
à l'exception du nœud d'origine initialisé à $(0, -)$



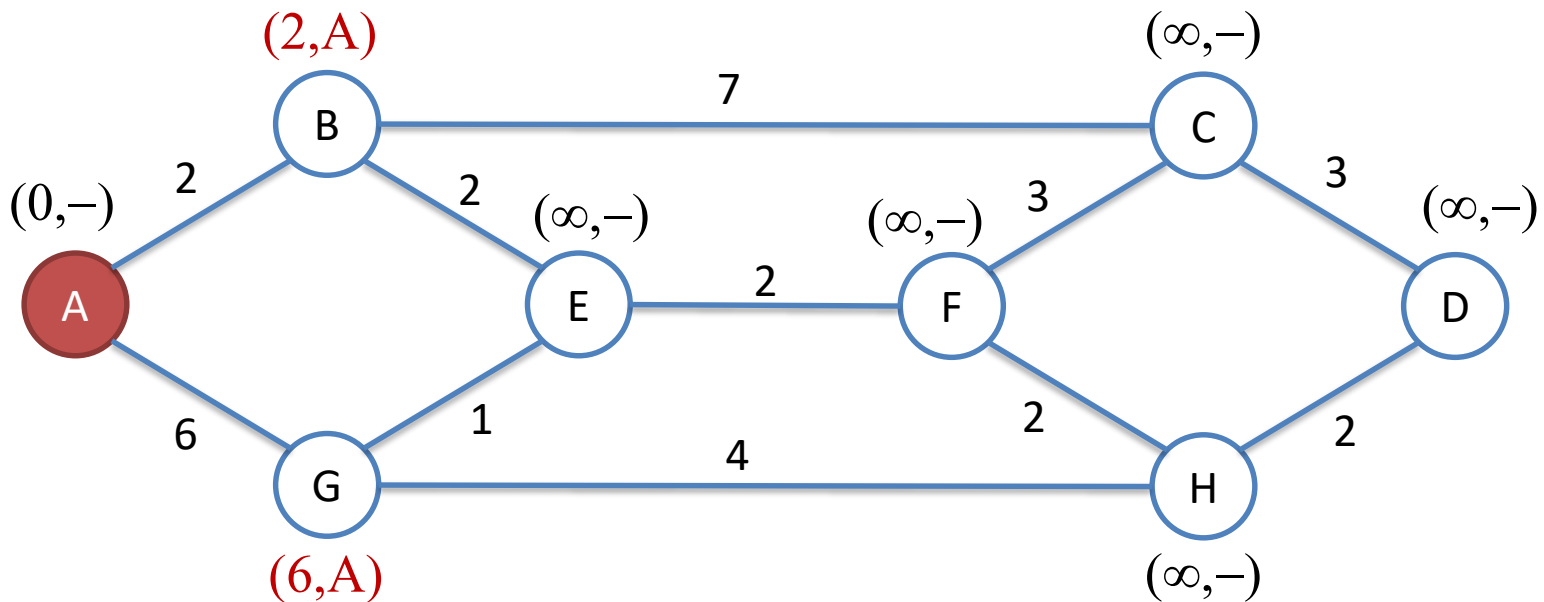
Algorithme de Dijkstra

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'



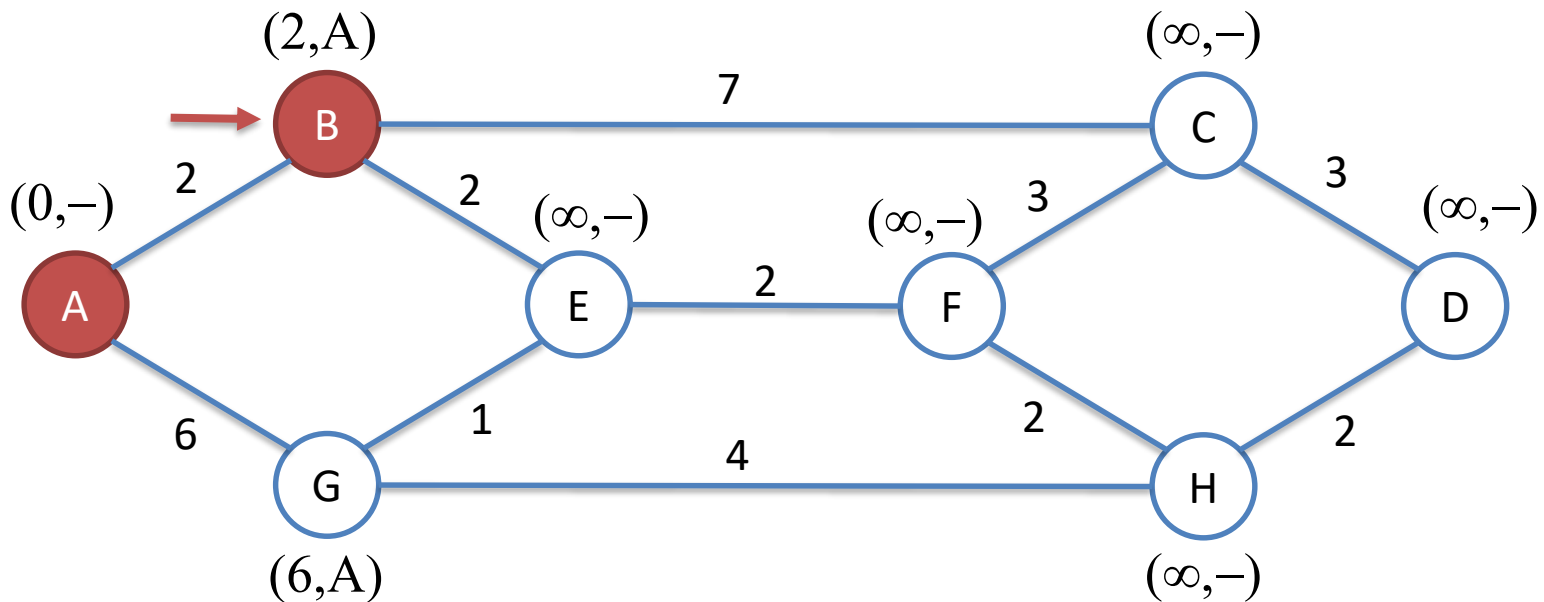
Algorithme de Dijkstra

3. Calculer les coûts des chemins de tous les voisins N_j du nœud N_i : $C_j = C_i + p_{ik}$.
4. Si la nouvelle valeur C_j est plus petite que l'ancienne, --> actualiser le doublet de N_j : (C_j, N_i)



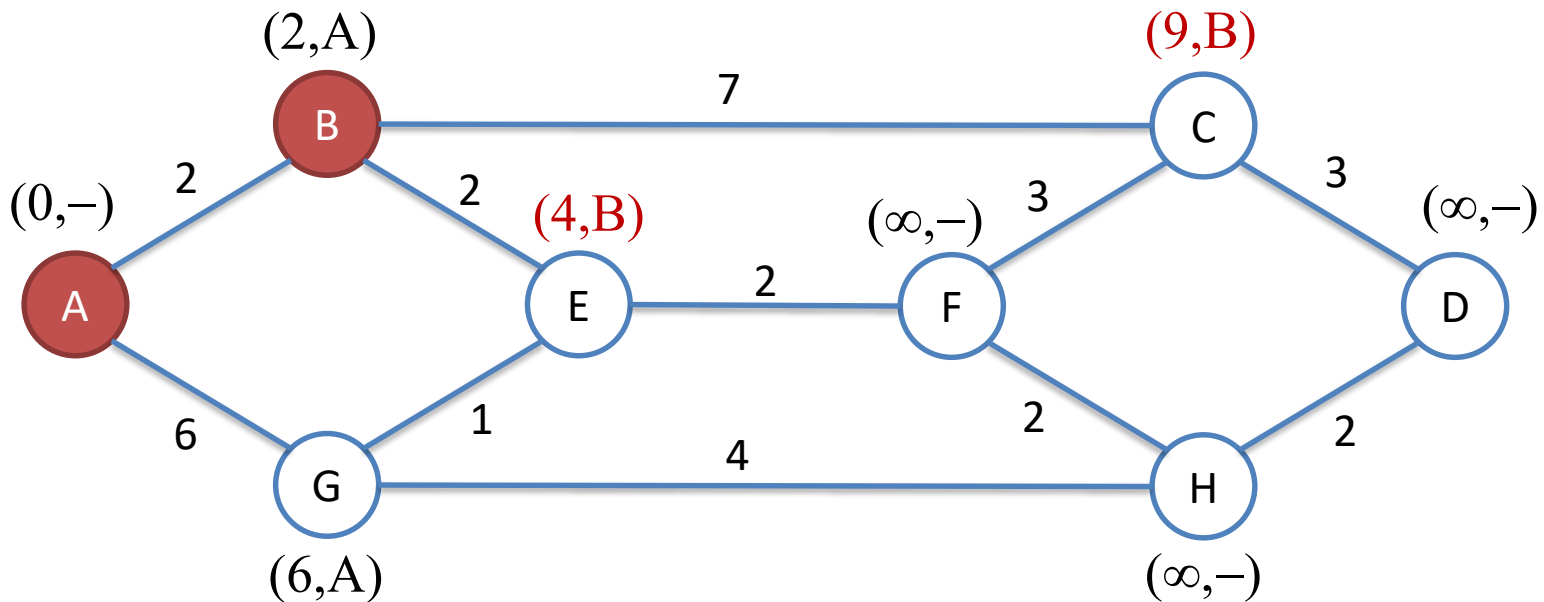
Algorithme de Dijkstra

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'



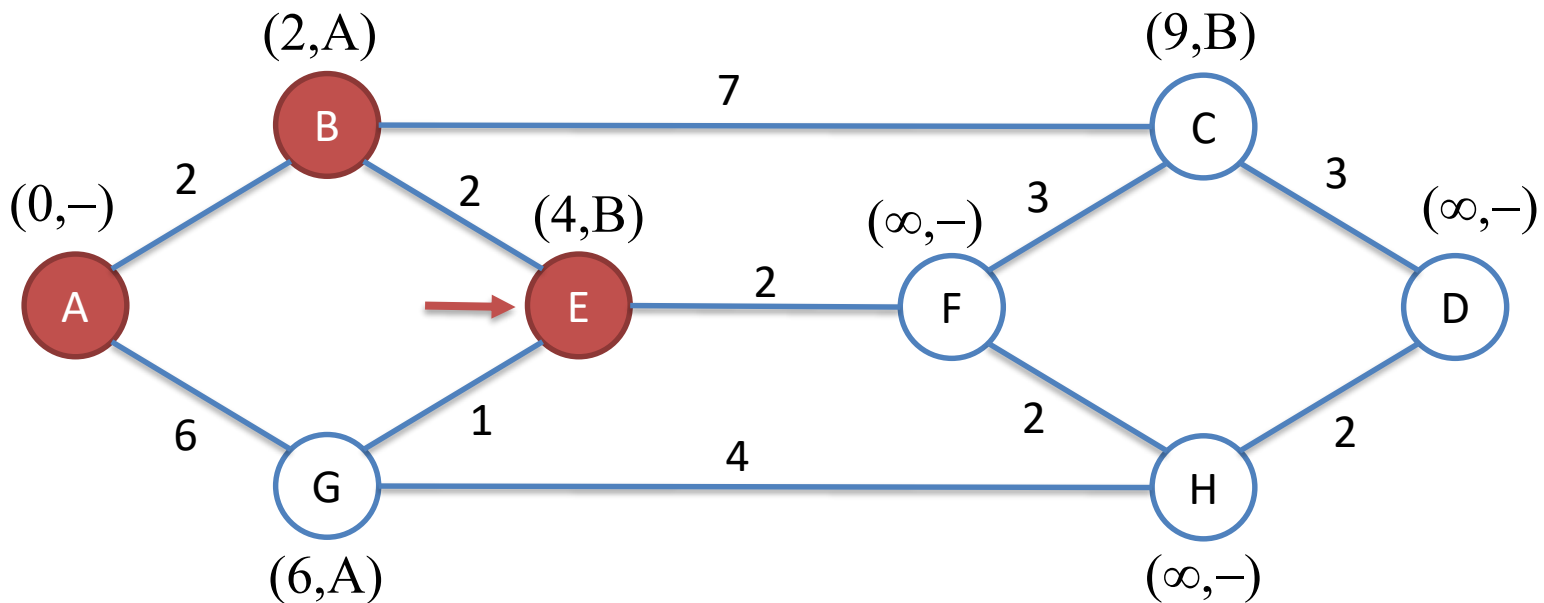
Algorithme de Dijkstra

3. Calculer les coûts des chemins de tous les voisins N_j du nœud N_i : $C_j = C_i + p_{ik}$.
4. Si la nouvelle valeur C_j est plus petite que l'ancienne, --> actualiser le doublet de N_j : (C_j, N_j)



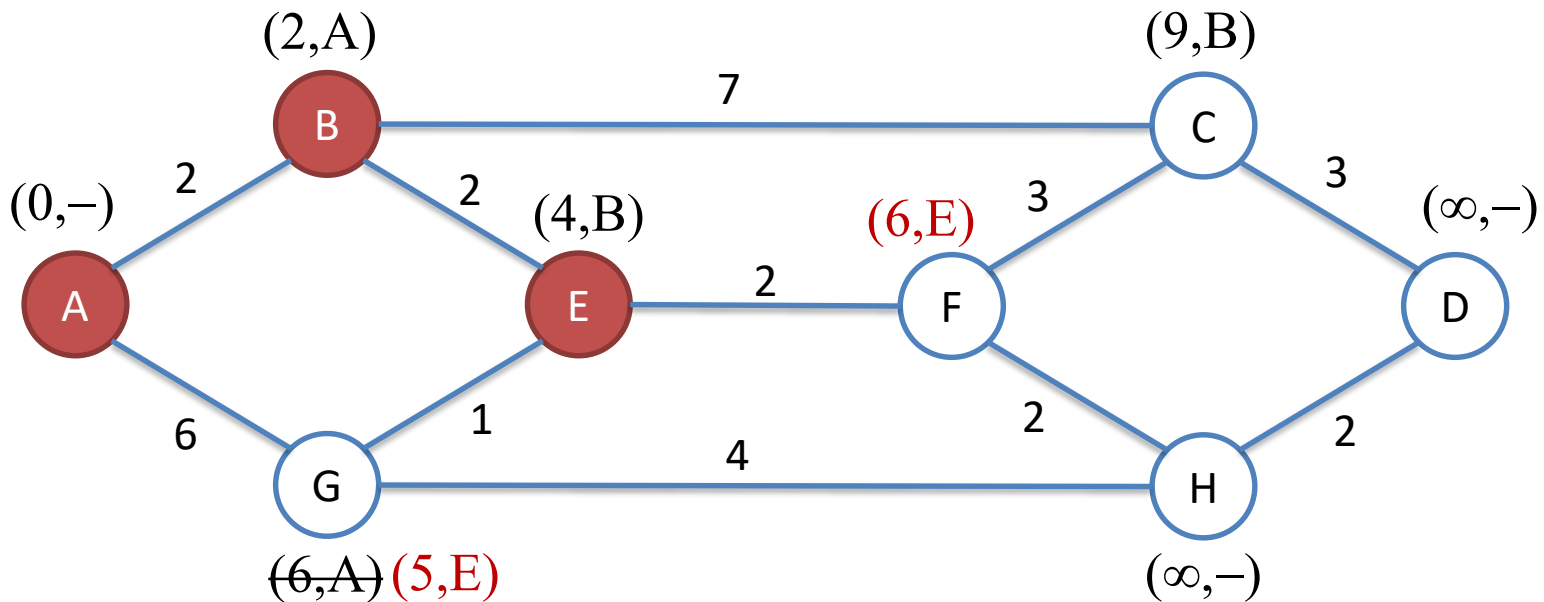
Algorithme de Dijkstra

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'



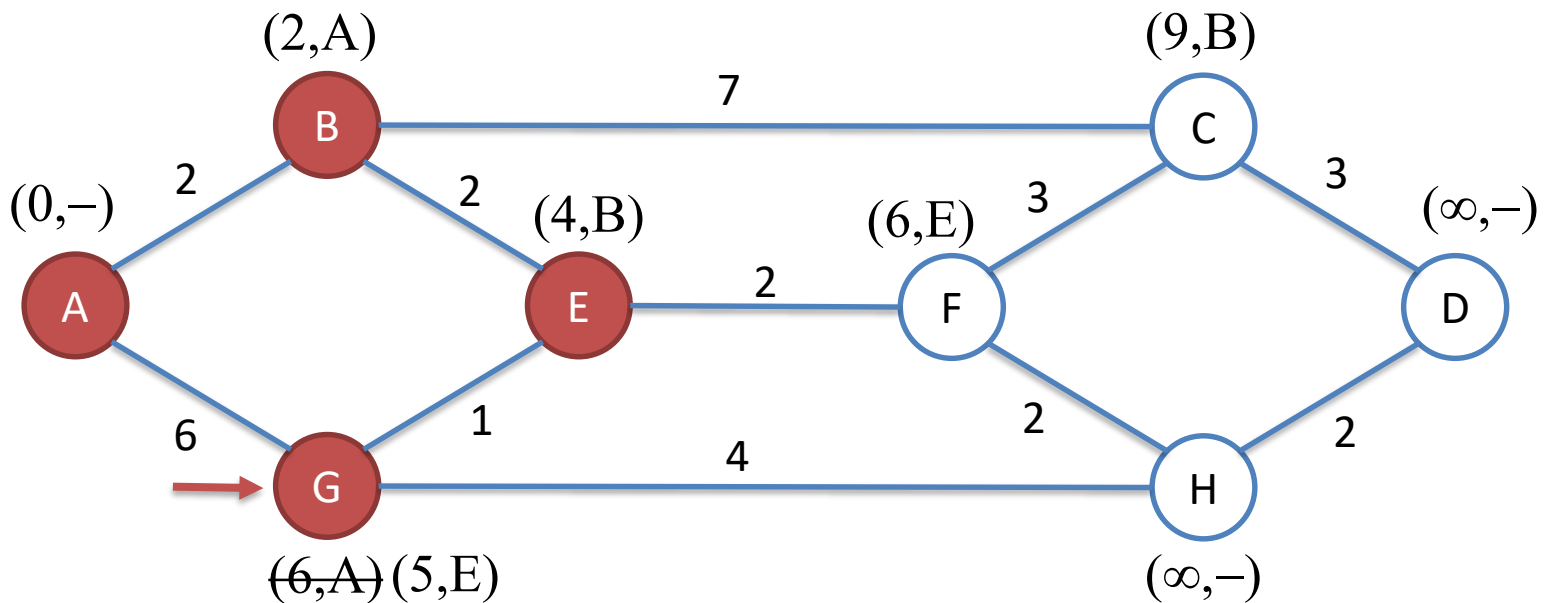
Algorithme de Dijkstra

3. Calculer les coûts des chemins de tous les voisins N_j du nœud N_i : $C_j = C_i + p_{ik}$.
4. Si la nouvelle valeur C_j est plus petite que l'ancienne,
--> actualiser le doublet de N_j : (C_j, N_j)



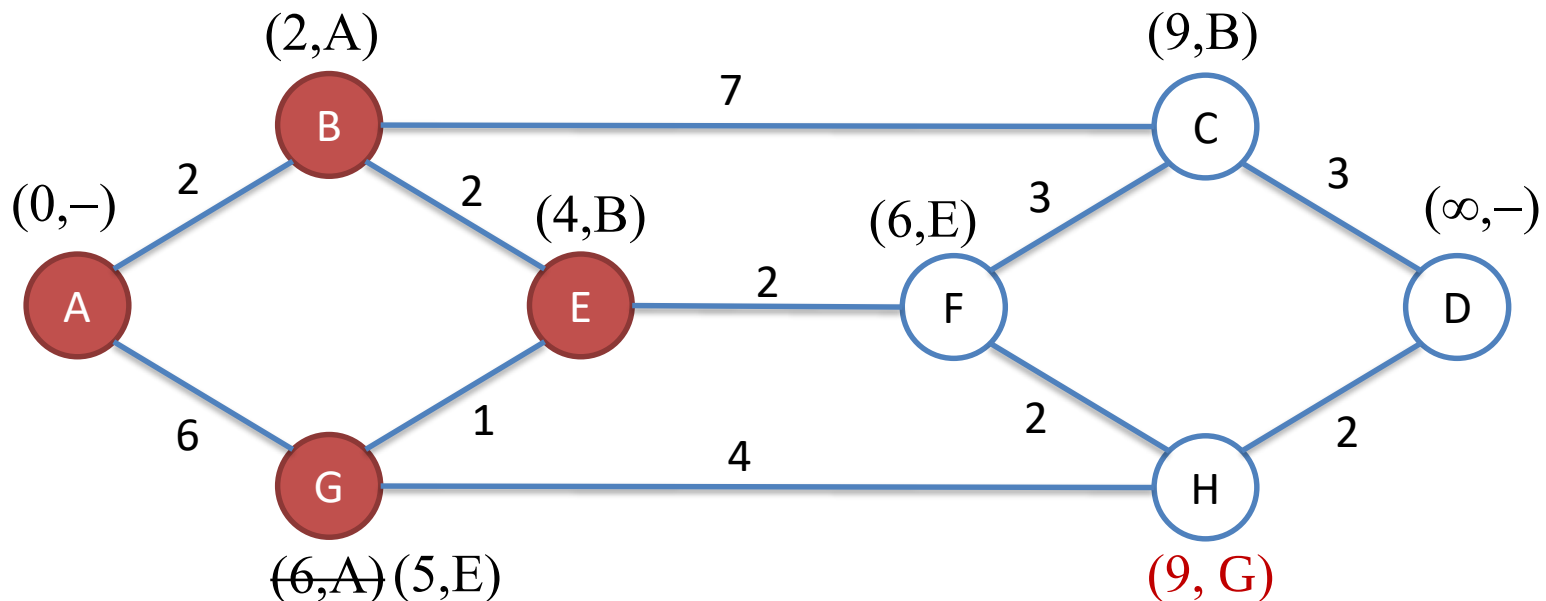
Algorithme de Dijkstra

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'



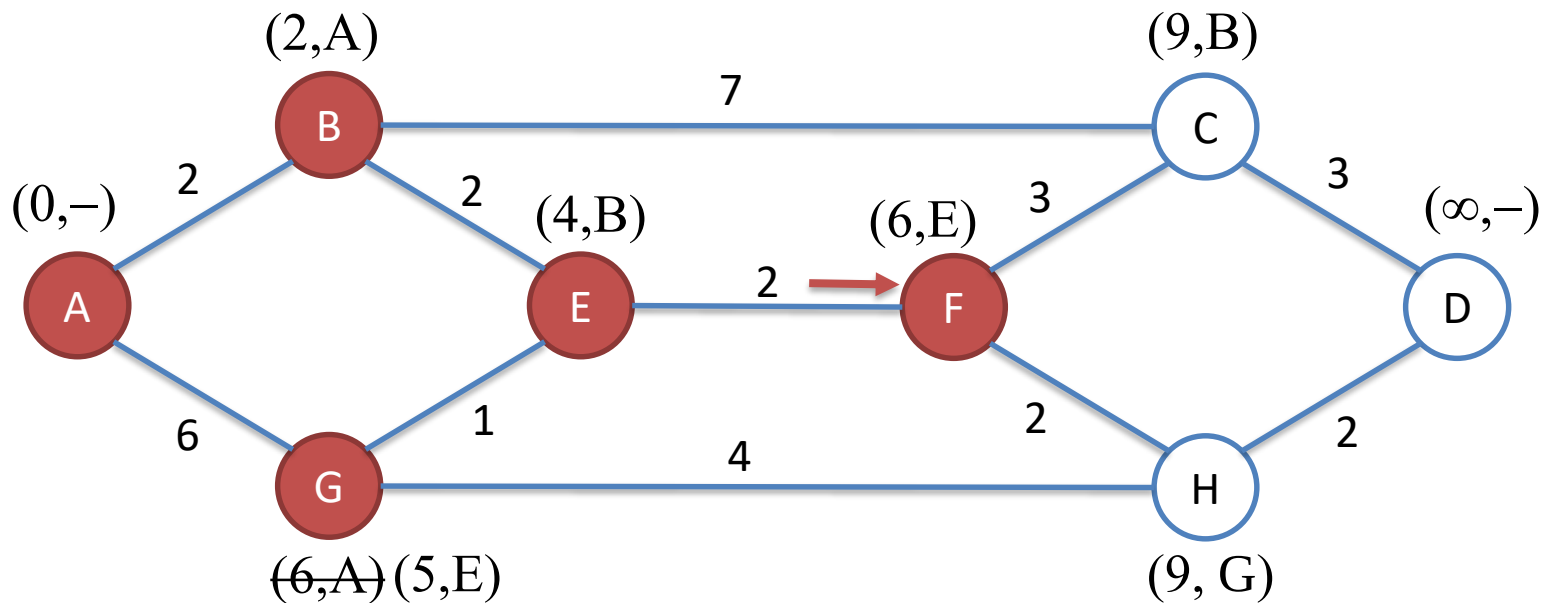
Algorithme de Dijkstra

3. Calculer les coûts des chemins de tous les voisins N_j du nœud N_i : $C_j = C_i + p_{ik}$.
4. Si la nouvelle valeur C_j est plus petite que l'ancienne, --> actualiser le doublet de N_j : (C_j, N_i)



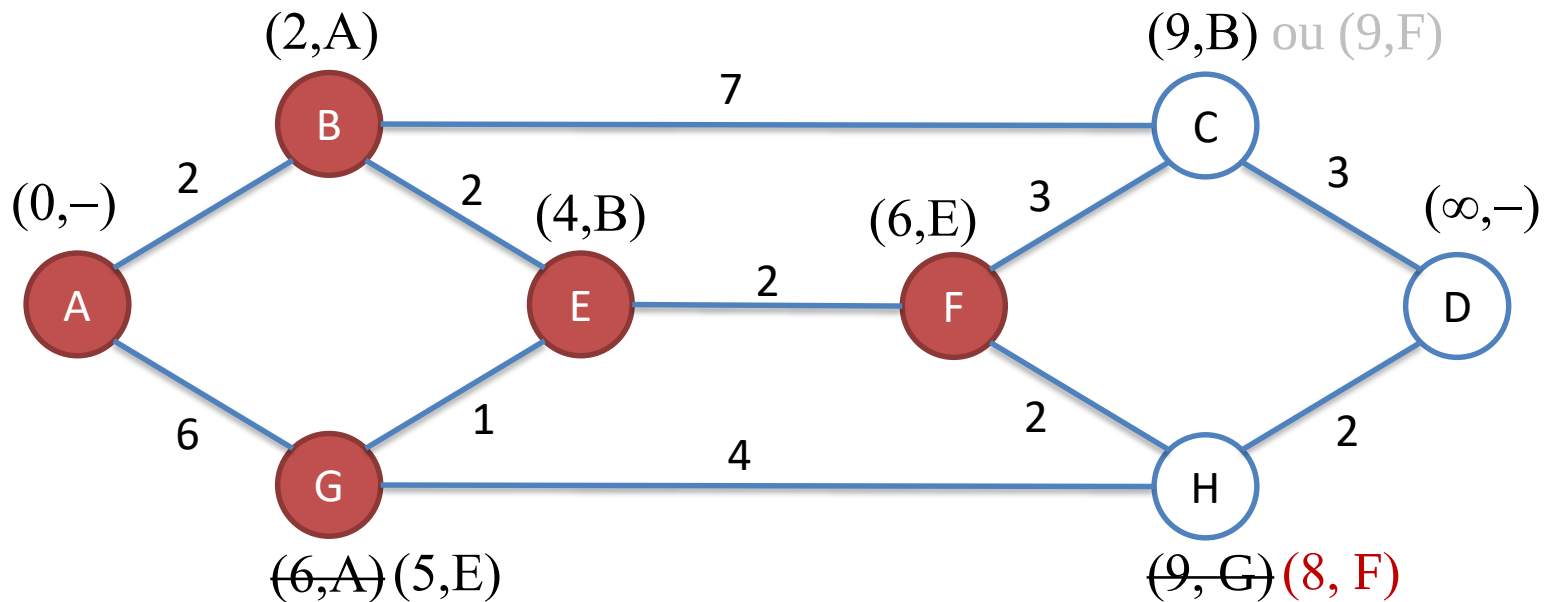
Algorithme de Dijkstra

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'



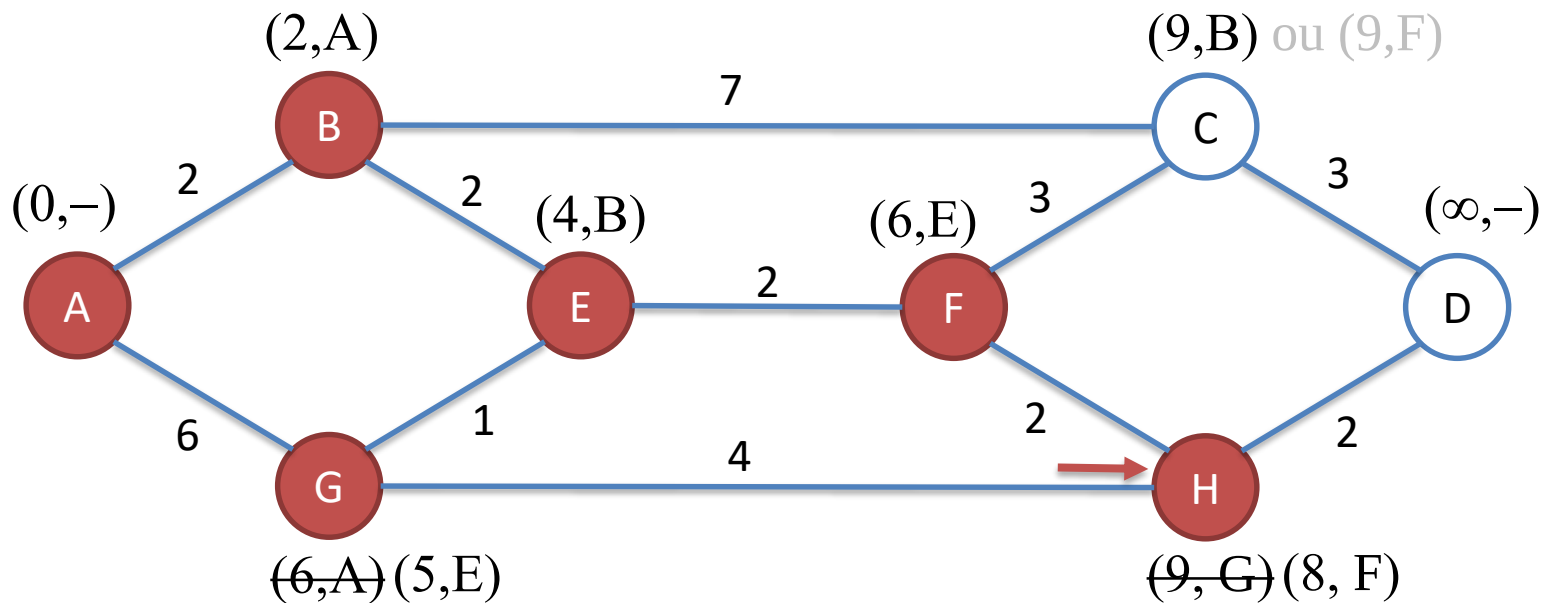
Algorithme de Dijkstra

3. Calculer les coûts des chemins de tous les voisins N_j du nœud N_i : $C_j = C_i + p_{ik}$.
4. Si la nouvelle valeur C_j est plus petite que l'ancienne,
--> actualiser le doublet de N_j : (C_j, N_j)



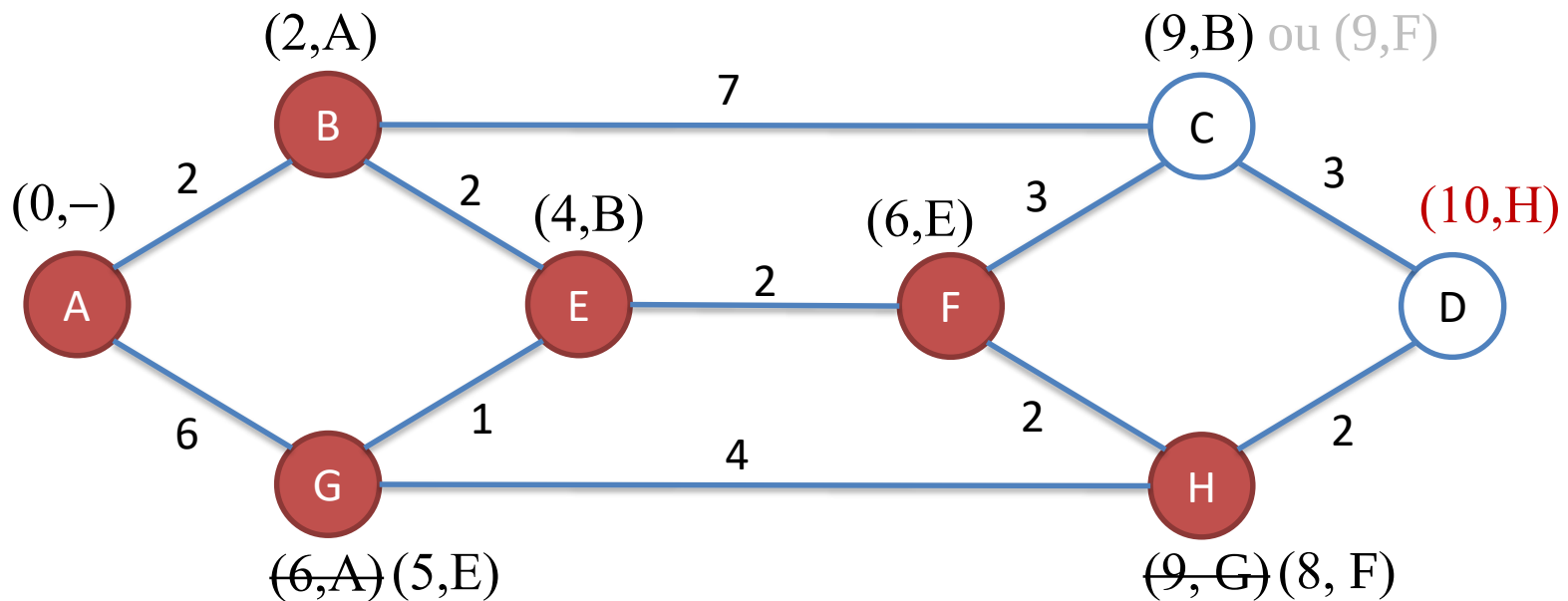
Algorithme de Dijkstra

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'



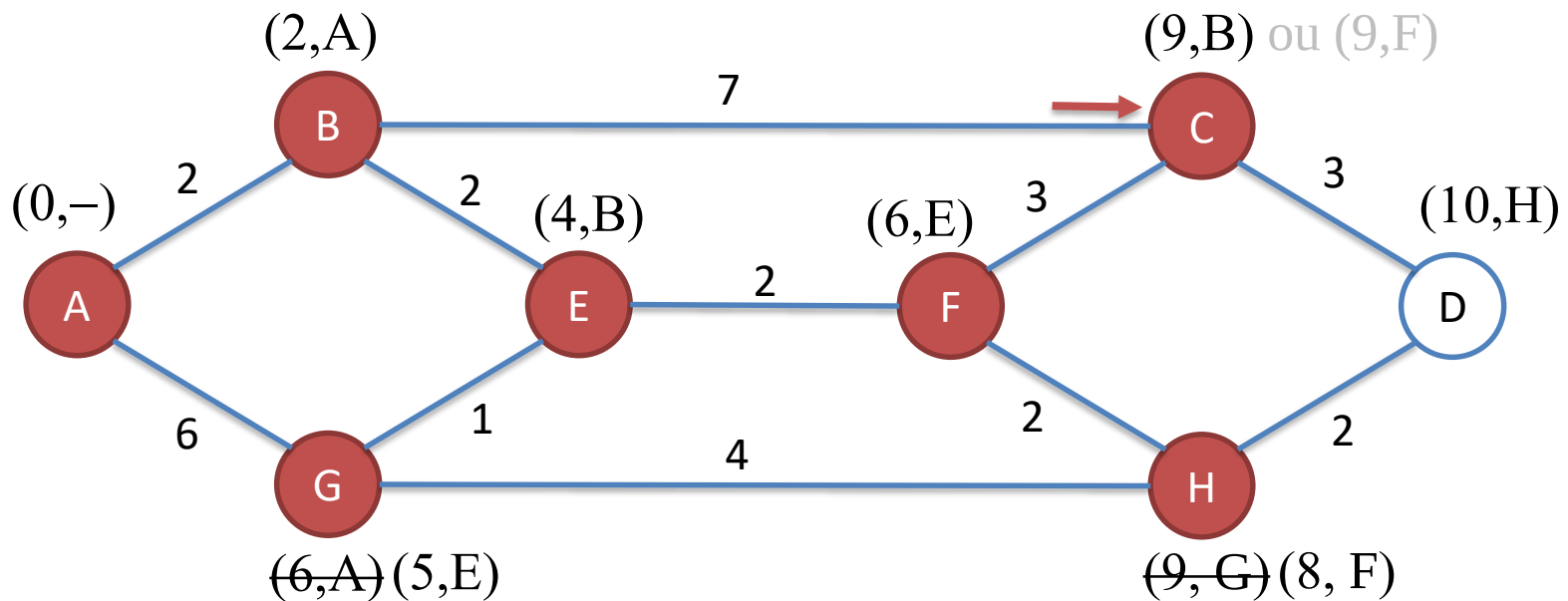
Algorithme de Dijkstra

3. Calculer les coûts des chemins de tous les voisins N_j du nœud N_i : $C_j = C_i + p_{ik}$.
4. Si la nouvelle valeur C_j est plus petite que l'ancienne, --> actualiser le doublet de N_j : (C_j, N_j)



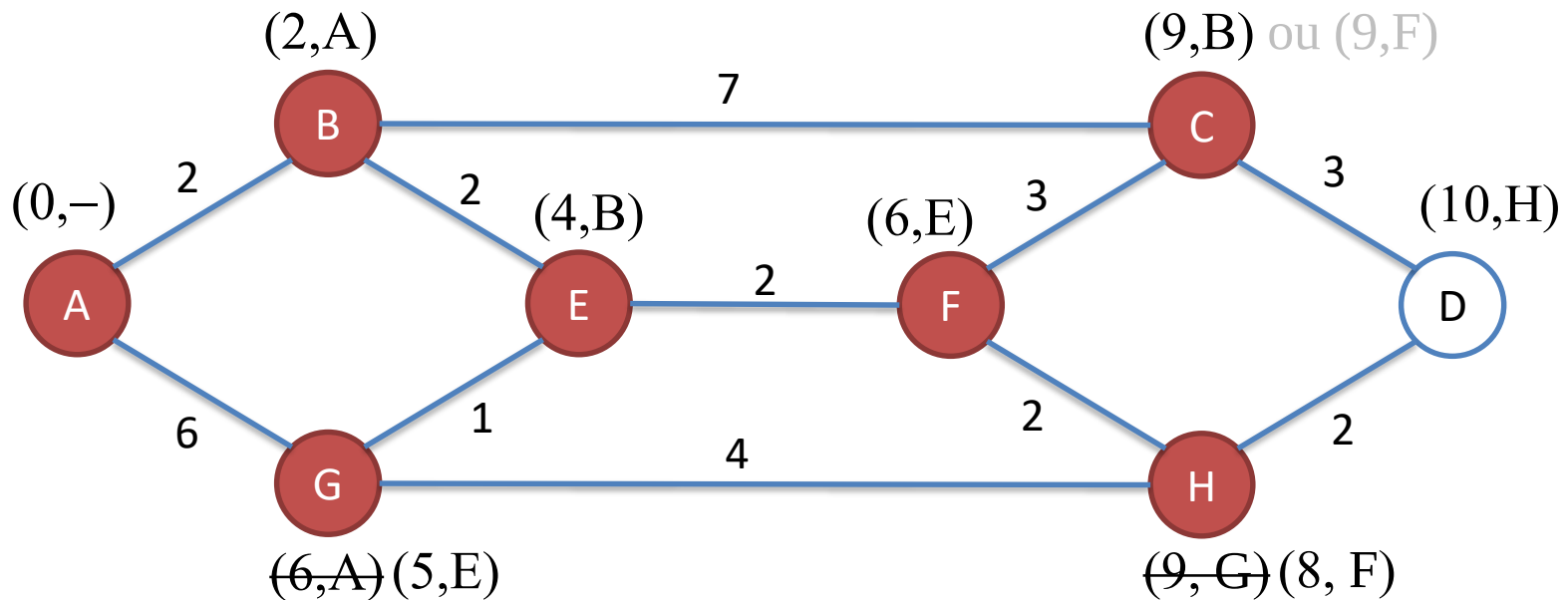
Algorithme de Dijkstra

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'



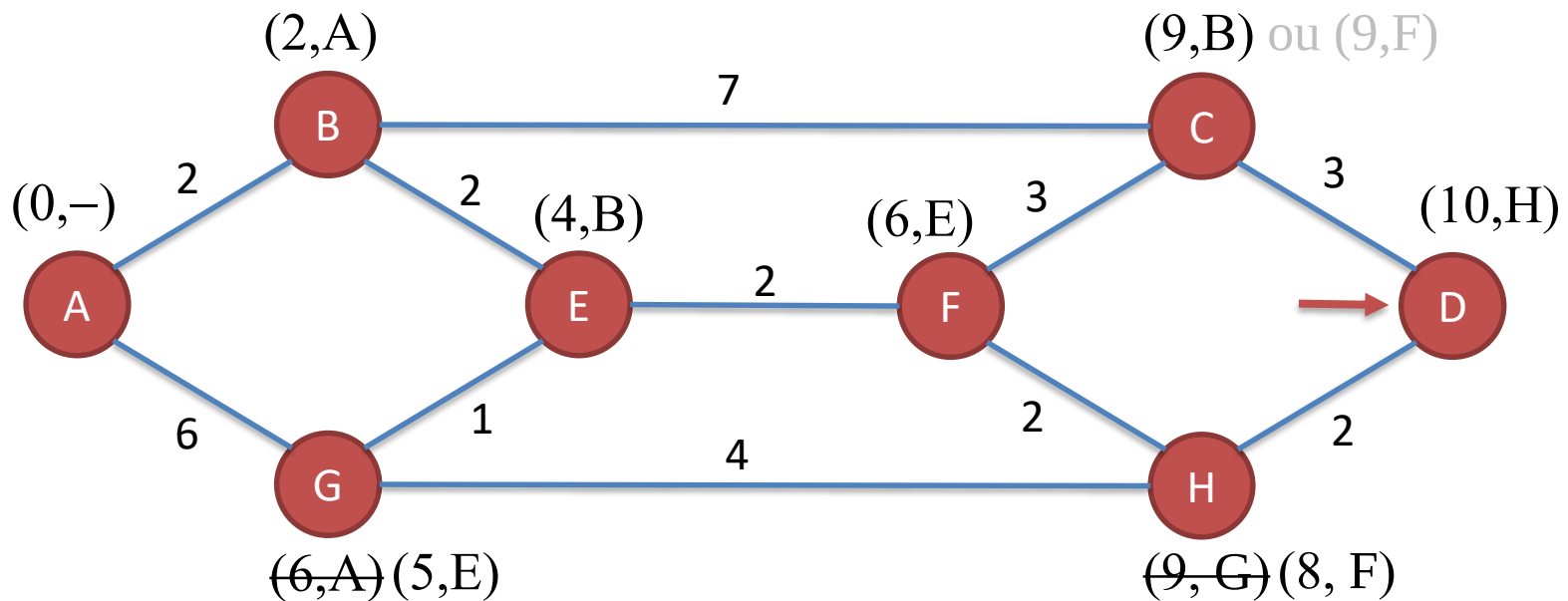
Algorithme de Dijkstra

3. Calculer les coûts des chemins de tous les voisins N_j du nœud N_i : $C_j = C_i + p_{ik}$.
4. Si la nouvelle valeur C_j est plus petite que l'ancienne,
--> actualiser le doublet de N_j : (C_j, N_j)



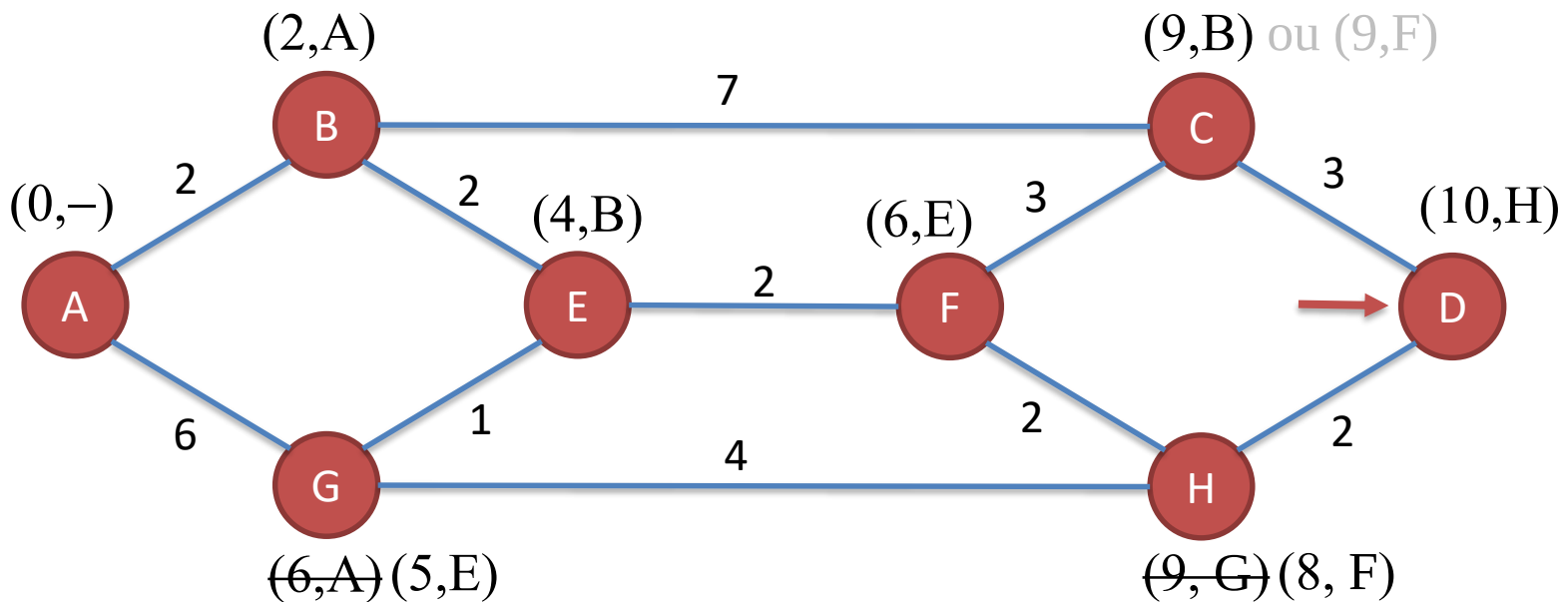
Algorithme de Dijkstra

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'



Algorithme de Dijkstra

2. Choisir le nœud N_i avec le coût C_i le plus bas et qui n'est pas marqué et le marquer comme 'permanent'



Exemple d'un protocole de routage: RIP

RIP: *Routing Information Protocol*

- Protocole de routage simple
- Utilisé dans de petits réseaux
- Utilise le nombre de sauts comme métrique pour calculer le plus court chemin
- Facile à configurer
- Fait partie de la famille de protocoles à « Vecteur de distance »
- Mais implique certaines limitations
 - Le diamètre du réseau est limité à 15 sauts
 - La mise à jour des routes peut être lente dans certaines situations



Fonctionnement de RIP

- Chaque routeur maintient une table de routage qui indique les distances vers les réseaux destinataires
- Chaque routeur envoie périodiquement sa table de routage (« vecteur de distances ») à tous ses voisins
- Chaque routeur utilise les tables de routage reçues pour calculer ses routes

Algorithme de Bellman-Ford distribué

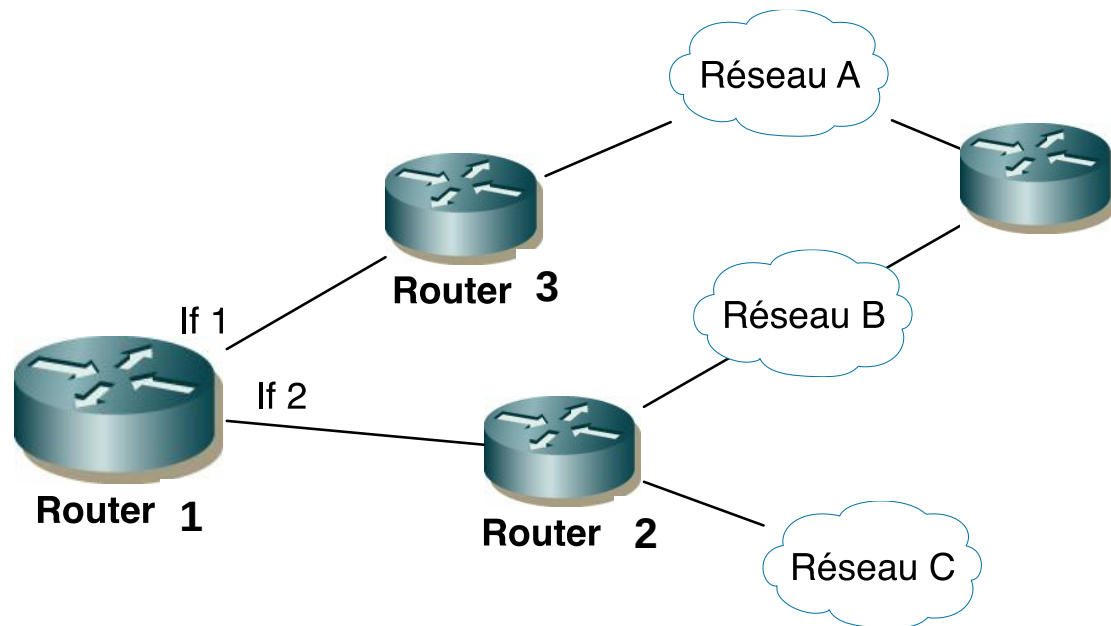
Pour calculer la meilleure route vers un réseau X

1. Le routeur choisit la route la plus courte vers X parmi celles annoncées par les voisins
2. Il incrémente la distance de la route de 1 pour tenir compte de la distance entre lui et le voisin



Exemple du calcul des routes

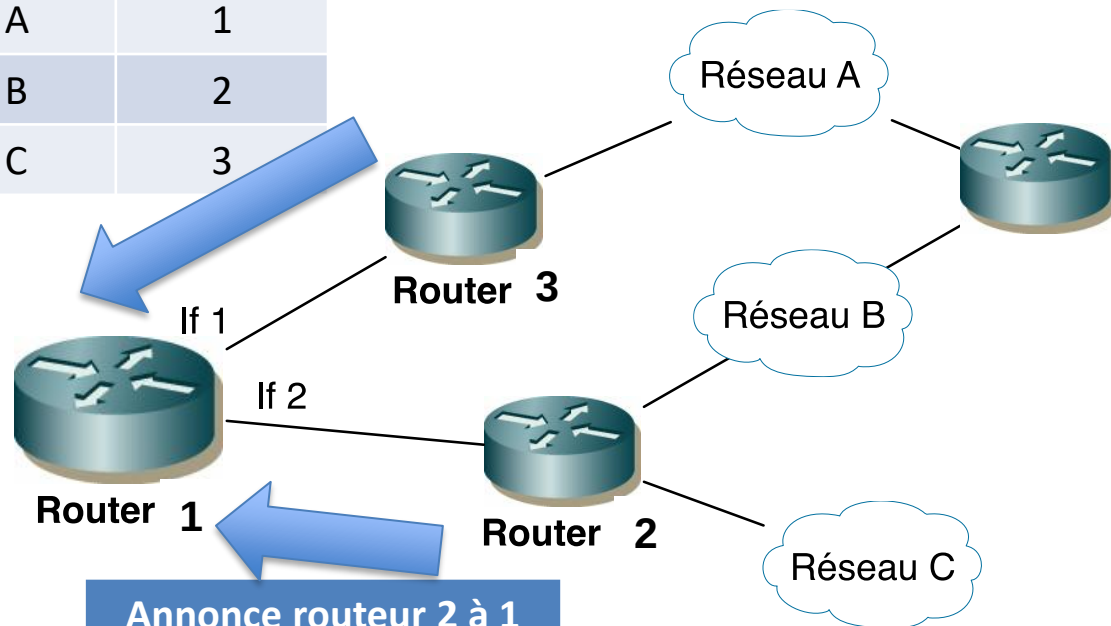
Table de routage router 1		
Destination	Sortie	Distance



Exemple du calcul des routes

Annonce routeur 3 à 1	
Destination	Distance
Réseau A	1
Réseau B	2
Réseau C	3

Table de routage router 1		
Destination	Sortie	Distance



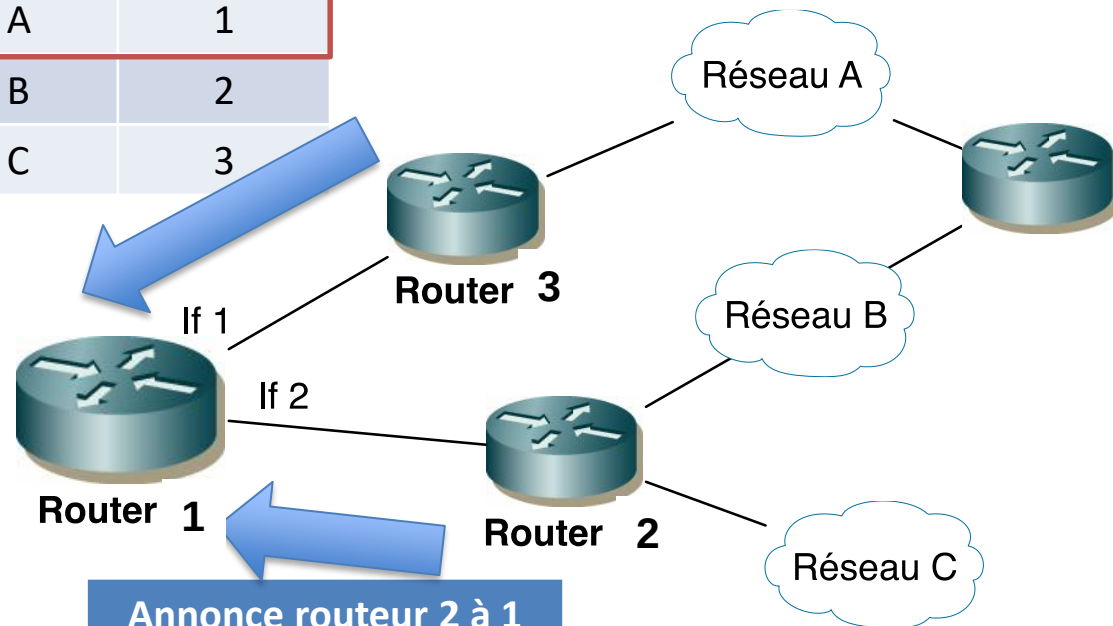
Annonce routeur 2 à 1	
Destination	Distance
Réseau A	2
Réseau B	1
Réseau C	1



Exemple du calcul des routes

Annonce routeur 3 à 1	
Destination	Distance
Réseau A	1
Réseau B	2
Réseau C	3

Table de routage router 1		
Destination	Sortie	Distance
Réseau A	If 1	2



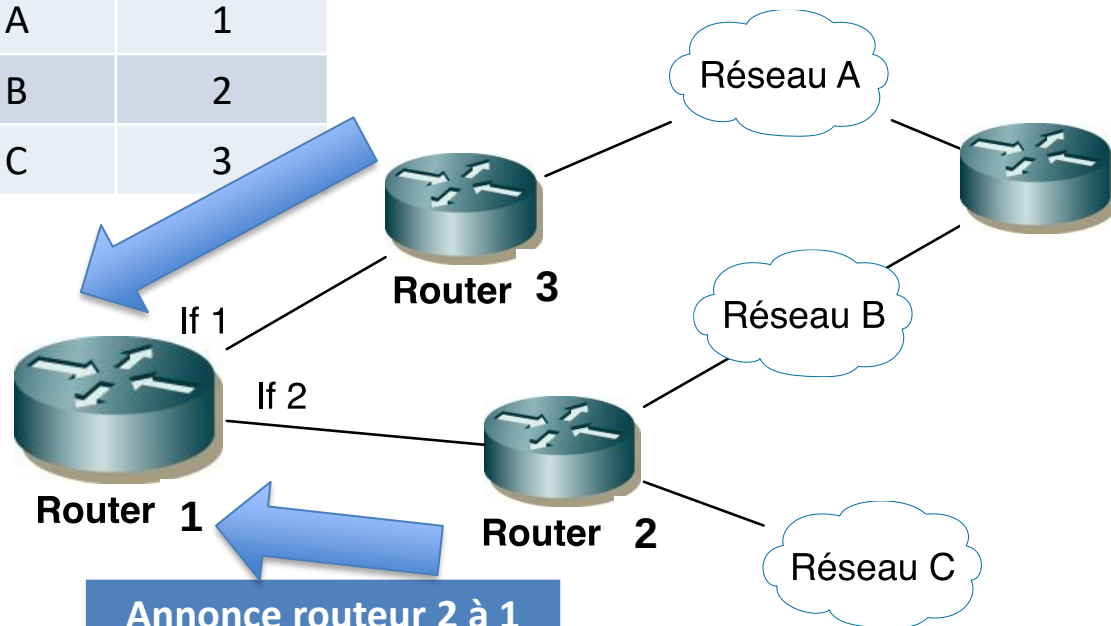
Annonce routeur 2 à 1	
Destination	Distance
Réseau A	2
Réseau B	1
Réseau C	1



Exemple du calcul des routes

Annonce routeur 3 à 1	
Destination	Distance
Réseau A	1
Réseau B	2
Réseau C	3

Table de routage router 1		
Destination	Sortie	Distance
Réseau A	If 1	2
Réseau B	If 2	2
Réseau C	If 2	2



Annonce routeur 2 à 1	
Destination	Distance
Réseau A	2
Réseau B	1
Réseau C	1



Propagation de bonnes nouvelles

- Une meilleure route se propage rapidement
- Exemple simple :
 - Réseau linéaire
 - Distance: nombre de sauts
 - Nœud A vient de démarrer

A	B	C	D	E	
●	●	●	●	●	
	∞	∞	∞	∞	État initial
	1	∞	∞	∞	Après 1 échange
	1	2	∞	∞	Après 2 échanges
	1	2	3	∞	Après 3 échanges
	1	2	3	4	Après 4 échanges



Propagation de mauvaises nouvelles

- Après une panne, le **routage converge très lentement**
- Exemple :
 - Lien entre A et B tombe en panne

A	B	C	D	E	
●	●	●	●	●	
	1	2	3	4	État initial
	3	2	3	4	Après 1 échange
	3	4	3	4	Après 2 échanges
	5	4	5	4	Après 3 échanges
	6	5	6	5	Après 4 échanges
	7	6	7	6	Après 5 échanges
	...	...	...	...	
	∞	∞	∞	∞	Après n échanges

➤ Problème de la valeur infinie



Heuristiques pour accélérer la convergence

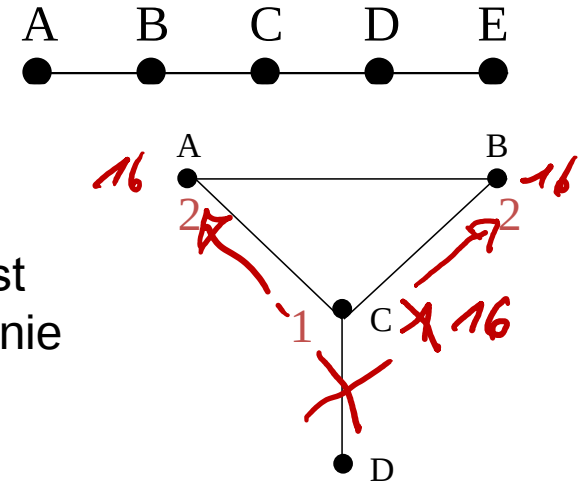
1. Définir une **distance maximale n**

- Une distance au-delà de n est infinie
- Limite la taille maximale d'un réseau

2. **Horizon éclaté (*Split horizon*)**

- La distance vers une destination n'est pas annoncée au nœud suivant dans cette direction

- Exemple
 - C n'annonce pas la route vers A à B



3. **Horizon éclaté avec retour empoisonné**

- Un routeur avise ses voisins qu'une route est devenue impraticable avec une distance infinie



Routage par état de lien

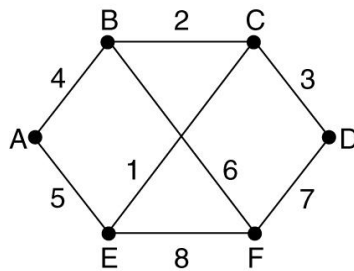
Famille de protocoles qui vise à corriger les problème des méthodes par vecteur de distance.

Principe

- Chaque routeur effectue périodiquement les opérations suivantes:
 1. Découvrir ses voisins et apprendre leurs adresses respectives
 2. Déterminer la distance vers chacun des voisins
 3. Construire un paquet contenant l'information apprise
 4. Envoyer ce paquet spécial à tous les autres routeurs du sous-réseau
 5. Calculer le plus court chemin vers tous les autres routeurs
- Un routeur apprend alors la topologie complète du réseau
- Calculer le plus court chemin en local

Envoi des états de lien aux voisins

- Chaque routeur construit des paquets contenant l'information sur l'état des **liens locaux** (LSP: *link state packet*)



		Link		State		Packets					
A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

- Les LSP d'un routeur sont diffusés dans le réseau entier
 - Inondation fiable
 - Un routeur transmet un LSP reçu sur tous les ports
 - Un numéro de séquence permet d'éliminer des LSP dupliqués
 - Les autres routeurs enregistrent le LSP le plus récent de chaque autre routeur
 - Éliminer les LSP quand sa durée de vie est terminée
 - La réception d'un LSP est confirmé par un accusé de réception



Comparaison

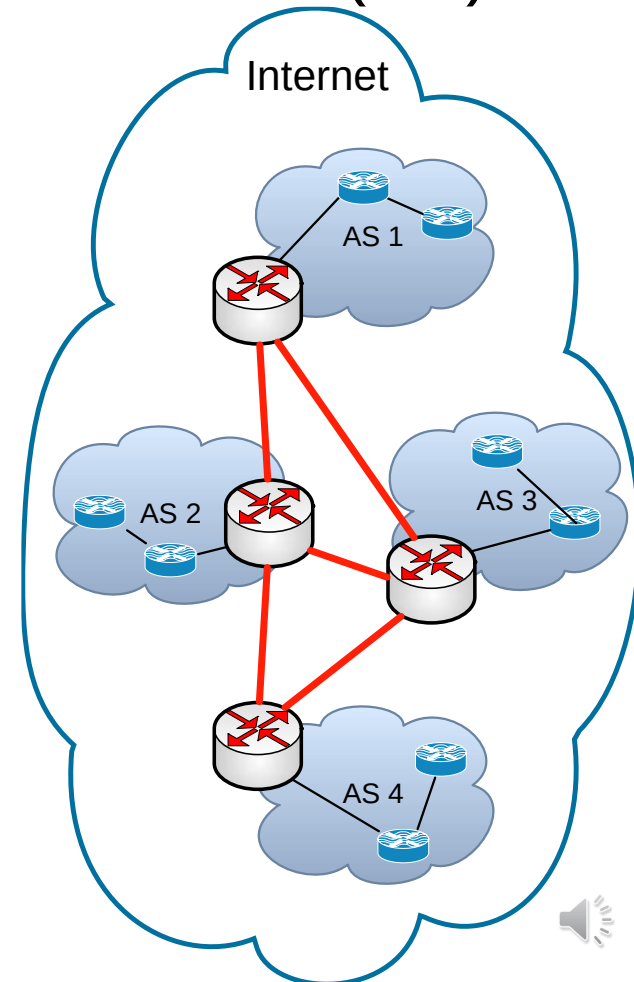
Vecteur de distance – État de lien

Caractéristique	Vecteur de distance	Etat de lien
Information envoyée	Vecteurs de distance: état du réseau global , vu par le routeur	Envoi des états de lien: état des liens locaux , vus par le routeur
Destinataires de l'information	Uniquement les voisins directs	Tous les nœuds du réseau, par inondation
Méthode de calcul	Bellman-Ford calcul partiel pour compléter le calcul effectué par les voisins	Dijkstra calcul complet sur la base des informations reçues de tous les autres nœuds
Avantages	Protocole simple	Convergence rapide et fiable
Inconvénients	Convergence peut être lente Vecteurs de distance peuvent être volumineux	Envoi des message par inondation est complexe



Niveaux de routage

- Un ou plusieurs réseaux sous le même contrôle administratif forment un Système Autonome (AS)
- Chaque AS est libre de choisir son protocole de routage
- Deux types de routage
 - Routage à l'intérieur d'un AS
 - Calcule les routes optimales
 - Par exemple RIP, OSPF, IS-IS
 - Routage entre les AS
 - Assure l'accessibilité des AS
 - Ne peut pas utiliser de métrique
 - Principalement le protocole BGP4



Fin du chapitre 6

