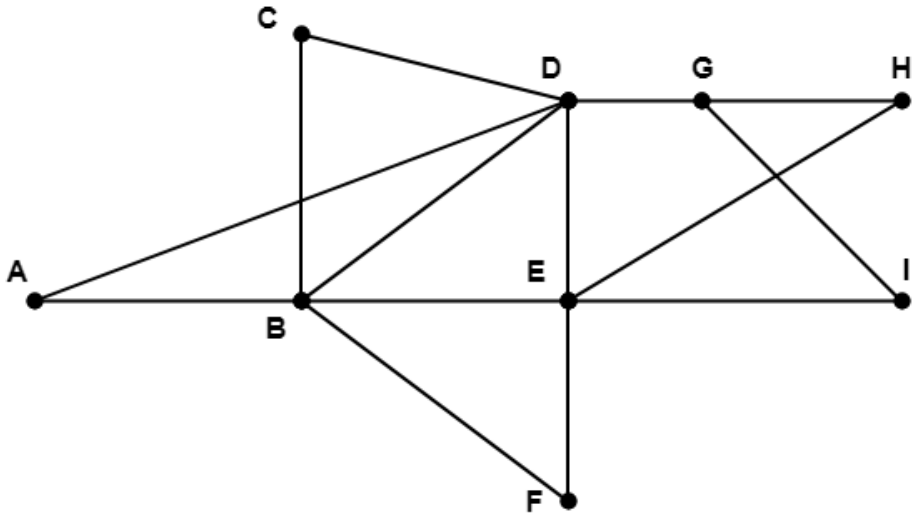
ACTIVITE 3 - LE ROUTAGE : AUTOMATISER LA RECHERCHE DU CHEMIN LE PLUS COURT

L'Internet étant un réseau très vaste et très variable, il est difficile d'y trouver son chemin. C'est d'autant plus difficile qu'il s'agit d'un réseau de réseaux et que différents types de réseaux sont interconnectés, pouvant fonctionner très différemment. Pour être transmis vers des machines qui sont sur un autre réseau, un message doit transiter par des **passerelles** (qu'on appelle aussi des **routeurs**) qui assurent l'interface entre différents réseaux. L'efficacité du routage malgré la complexité et l'évolution permanente de l'Internet est un point essentiel de son succès. Ce processus prend place dans le cadre du **protocole IP**, associé à des protocoles spécialisés dans le routage. Sans rentrer dans les difficultés techniques, nous allons essayer d'en comprendre les principes. En travaillant sur les schémas ci-dessous, il faudra bien garder en tête que chaque point (ou **nœud**) correspond en fait à une passerelle, un point d'entrée vers un sous-réseau qui n'est pas représenté.

3.1

Sur le réseau 1 ci-dessous, recherchez le plus court chemin entre C et I (en termes de nombre de sauts)



Il n'y a pas UN unique plus court chemin, mais trois : CBEI, CDEI et CDGI

3.2

Observez la manière dont le graphe en arbre permet d'identifier, dans le réseau 1, tous les plus courts chemins à partir du nœud A, vers tous les autres nœuds du réseau. Appliquez la même méthode à partir du nœud B et remplissez le tableau ci-dessous.

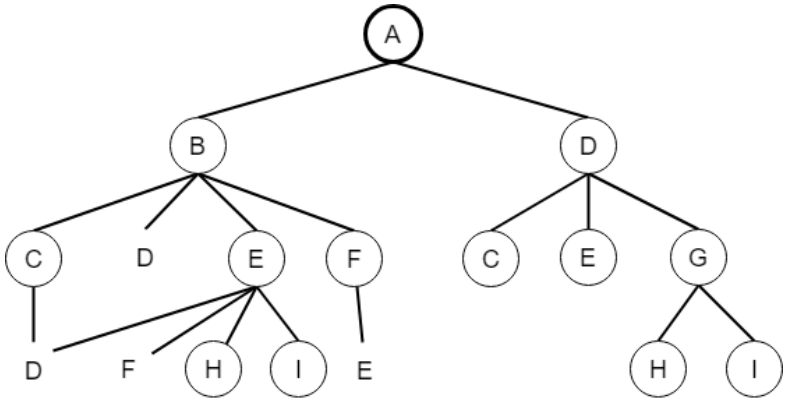


Tableau des plus courts chemins depuis A

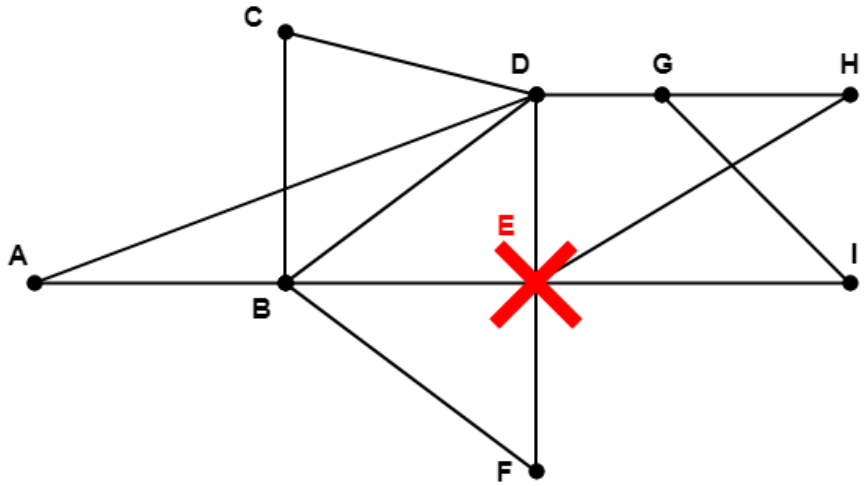
Destination	Plus court(s) chemin(s)
B	direct
C	BC, DC
D	direct
E	BE, DE
F	BF
G	DG
H	BEH, DGH
I	BEI, DGI

Tableau des plus courts chemins depuis B

Destination	Plus court(s) chemin(s)
A	Direct
C	Direct
D	Direct
E	Direct
F	Direct
G	DG
H	EH
I	EI

3.3

Imaginez que le routeur E tombe en panne ou soit saturé. Quel est alors le meilleur chemin entre B et I ? Combien de « meilleurs chemins » sont affectés et doivent être recalculés ?



Le meilleur chemin de B vers I devient BDGI
Seulement 2 si on parle des chemins depuis B, à savoir vers H et I, mais si on parle de tous les chemins possibles depuis tous les nœuds, davantage sont affectés et on imagine bien qu'avec un réseau (beaucoup) plus complexe comme Internet, cela deviendrait vite très difficile de recalculer tous les chemins affectés avec la méthode des graphes, c'est pourquoi on va envisager une autre solution.

3.4

Le réseau Internet est beaucoup plus complexe et sa structure change très fréquemment (routeurs en panne ou momentanément saturés, liaisons coupées, nouveaux routeurs, nouveaux réseaux locaux, etc.). La recherche du chemin le plus court doit être automatisée et optimisée pour que la plupart des messages soient acheminés en quelques dizaines de millisecondes à l'autre bout de la Terre. Le tableau suivant présente différentes stratégies envisageables mais qui n'auraient jamais permis à l'Internet de se développer jusqu'à son échelle actuelle. Chercher les inconvénients que présentent ces stratégies.

Stratégies d'acheminement	Inconvénients majeurs
Chaque message est **diffusé** dans toutes les directions en chaque point, il finira nécessairement par arriver à destination.	Solution envisageable sur un petit réseau local, le message se diffuse et n'est reconnu que par le destinataire mais dans un réseau de réseaux cela produit énormément de redondance inutile et chaque routeur est sollicité pour renvoyer dans tous les sens
Chaque routeur possède un **tableau des chemins les plus courts** vers tous les autres nœuds, qu'il doit mettre à jour à chaque changement.	Chaque changement réclame de recalculer tous les chemins concernés, donc retester toutes les possibilités à chaque nœud, ce qui peut être très lourd, d'autant que cela doit être fait de même par tous les routeurs du réseau qui a au moins un chemin qui passe par le nœud en panne
Chaque routeur possède une **carte complète du réseau**, mise à jour régulièrement, consultée chaque fois qu'un message rencontre une difficulté et doit trouver un chemin alternatif.	Une carte totale est très volumineuse quand il y a plusieurs milliers de nœuds, c'est un document lourd à manipuler, il faut un ordinateur très puissant, et si elle est dupliquée sur chaque routeur, c'est une redondance très couteuse (et il est aussi coûteux de s'assurer que chaque carte est à jour) Grâce à la carte, on pourrait croire qu'il est plus simple de modifier un chemin en cas de panne, mais un programme ne regarde pas une carte, il faut bien qu'il fasse un calcul du même type que le précédent !
Une carte complète du réseau est disponible sur un **serveur central** accessible à tous les routeurs et régulièrement mise à jour.	Cela supprime le problème de la redondance, mais si tous les routeurs doivent sans cesse interroger le serveur central ou l'informer de changements, cela risque de générer des congestions et ne semble pas compatible avec le fait que les paquets sont routés en quelque 10aines de ms sur l'Internet.

Pensez-vous à une autre solution ?

En fait, personne n'a de carte complète et précise d'Internet et c'est inutile pour le routage : il suffit que chaque routeur connaisse l'état de ses voisins immédiats et qu'il puisse indiquer, à chaque message, la direction à prendre (le saut à faire vers le prochain routeur), sans connaître tout le chemin qu'il devra parcourir. Mais comment être sûr qu'il indique la bonne direction ? Grâce à un **algorithme**, par lequel chaque routeur, dès qu'il est informé d'un changement d'état d'un de ses voisins immédiats, informe ses autres voisins de ce changement (qui modifieront éventuellement la direction qu'ils indiqueront aux futurs messages) et ainsi de suite : l'information sur le changement du réseau se propage **de proche en proche** et, en temps réel, chaque routeur adapte sa table de routage.

3.5

Reprenez le réseau 1 et complétez le tableau 2 qui indique seulement les directions.
Tout est-il utile dans ce tableau ? Comment le réorganiser pour qu'il soit plus compact ?

Destination	Prochain saut	Nb de sauts
B	direct	1
C	B ou D	2
D	direct	1
E	B ou D	2
F	B	2
G	D	2
H	B ou D	3
I	B ou D	3

On peut à la limite se contenter de 4 lignes :

Destination	Prochain saut	Nb de sauts
B, D	direct	1
G	D	2
H, I	D	3
autres	B (par défaut)	2

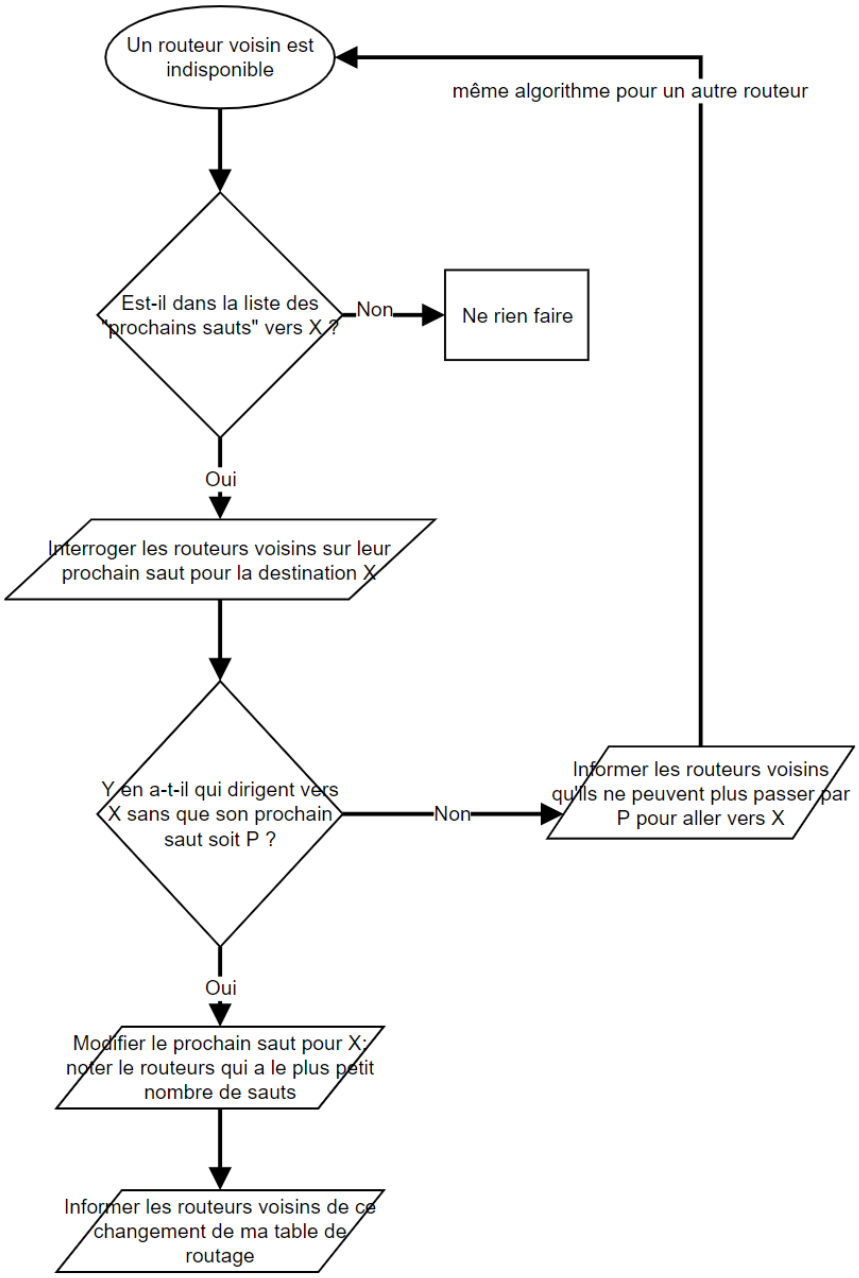
On aurait pu décider de mettre par défaut (vers B ou D, peu importe) toute destination supérieure à 2 sauts… On voit qu'on n'a pas besoin de tout écrire dans un table de routage, il y a en général un prochain saut par défaut pour toutes les destinations lointaines ou qu'ignore le routeur (ce qui peut, bien sur engendrer quelques problèmes ; il ne faut pas, par exemple, que deux routeurs reliés soient considérés l'un par l'autre comme prochain saut par défaut…)

3.6

Considérez l'algorithme suivant, qui schématise (très grossièrement par rapport à la réalité) ce que fait chaque routeur pour adapter sa table de routage lorsqu'il est informé qu'un routeur voisin est tombé en panne ou a été supprimé (on le nommera ici le routeur P). Pour simplifier, on ne considérera que la procédure mise en œuvre pour une destination donnée, X.

Appliquez cet algorithme au réseau ci-dessus, en imaginant que le routeur E est en panne : comment le routeur B va-t-il modifier, dans sa table de routage, sa route vers H ? Décrivez ce qui va se passer par ailleurs.

Que faudrait-il ajouter pour appliquer cette procédure à toutes les destinations qui passent par P ?



B va identifier D comme un routeur alternatif vers H, il va modifier sa table de routage en conséquence et indiquer 3 en nombre de sauts. Puis il va informer A, C et F qu'il a changé sa table de routage. Il aura également informé F que le routeur E est en panne, ce qui va enclencher chez lui le même algorithme d'adaptation à la panne.
Pour traiter toutes les destinations susceptibles d'être concernées, il faudrait ajouter une boucle qui répète pour toutes les directions (Pour…), la procédure à partir de l'interrogation des routeurs voisins, en testant si chacune des destinations (Si…) est concernant par la panne de P.

3.7

Imaginez maintenant que A est informé du changement que B a effectué dans sa table de routage. Quel est désormais le chemin le plus court de A vers I ?
Construisez, sur le modèle du précédent, un algorithme qui s'activerait dès qu'un routeur voisin informerait d'un changement de sa table de routage et qui aurait pour but de corriger si nécessaire sa propre table de routage (pour simplifier, vers une seule destination X) afin de garantir le chemin le plus court.

Le chemin le plus court de A vers I, qui pouvait être ABEI en 4 sauts, est désormais ADGI en 4 sauts également.

L'algorithme pourrait être de cet ordre :
1 – Le changement concerne-t-il une destination que je pointe ?
2 – Si oui, le nb de sauts qu'il indique est-il inférieur au nb de sauts que j'indique ?
- 3 – Si oui, changer la route pour passer par ce routeur vers X et aller au point 7
- 3' – Si non, est-ce le prochain saut que j'indique pour X ?
- 4 – Si oui, interroger mes autres voisins sur leur nombre de sauts vers X
  - 5 – Y a-t-il un voisin plus proche de X ?
  - 6 – Si oui, changer la route en retenant la plus courte
  - 6' – Si non modifier le nb de sauts vers X sans changer le prochain saut
  - 7 – Informer mes voisins de ce changement
- 4' – Si non ne rien faire

2' – Si non, ne rien faire

3.8

Ce système ne permet pas de garantir le temps de transmission, ni qu'un message ne s'égare pas dans des renvois sans fin (par ex. si un chemin crucial est HS). D'après vous, que faudrait-il faire pour éviter qu'un message n'ère indéfiniment et n'encombre le réseau ?

Il faut prévoir un mécanisme de suppression. Vu que personne ne contrôle de bout en bout le chemin d'un message, il faut qu'il porte avec lui une sorte de compte à rebours, au-delà duquel on estime qu'il n'arrivera pas à destination sans coût excessif : il est détruit lorsqu'il arrive à zéro. Il y a en effet un tel compteur dans l'entête IP (TTL : « Time To Live », que correspond au nombre de saut maximum : au passage de chaque routeur, il est réduit d'une unité. Lorsqu'un retour reçoit le paquet avec TTL=0, il le détruit. C'est ce qu'on appelle le « principe du meilleur effort ».

A retenir : **routage dynamique** = architecture **décentralisée**, très flexible = très robuste – qui repose essentiellement sur des algorithmes dont la charge est répartie sur tous les routeurs, plus que des machines puissantes capables de gérer de grandes quantités de données, ou une structure matérielle ultra-performante et organisée. C'est grâce à cela que l'Internet a pu se développer à moindre coût
Par contre, **le temps requis pour délivrer un message n'est pas garanti**, mais le système de contrôle qui serait nécessaire pour le garantir alourdirait énormément le système (au point qu'Internet ne se serait sans doute pas développé). Le fait qu'aucun routeur ne connaisse à l'avance le chemin exact des routeurs qu'il achemine n'est donc pas un souci (mais un gain de place et de temps de calcul).

Observer le routage en pratique

3.9

Utiliser la commande **`tracert`** [@IP] pour suivre le cheminement vers différents points du globe (cherchez l'adresse IP de différents sites internet, grâce à la commande **`ping`** par exemple)

- Une adresse en Europe
- Une adresse sur un autre continent ; puis testez d'autres adresses sur ce continent, que constatez-vous ?
- Enfin, cherchez une adresse aussi loin que possible, avec le maximum de sauts. Que constatez-vous au sujet du temps de latence des derniers sauts ?

Visualisez l'un de ce cheminement sur une carte, grâce à https://www.ip2location.com/free/traceroute ou grâce au logiciel libre **Open Visual Traceroute** . Le site https://stefansundin.github.io/traceroute-mapper/ permet également de visualiser sur une carte le trajet fourni par une commande **`tracert`**