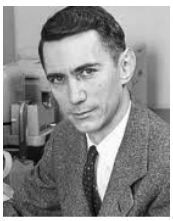
HE2B ESI

printemps '17 des sciences
les sciences à portée de main

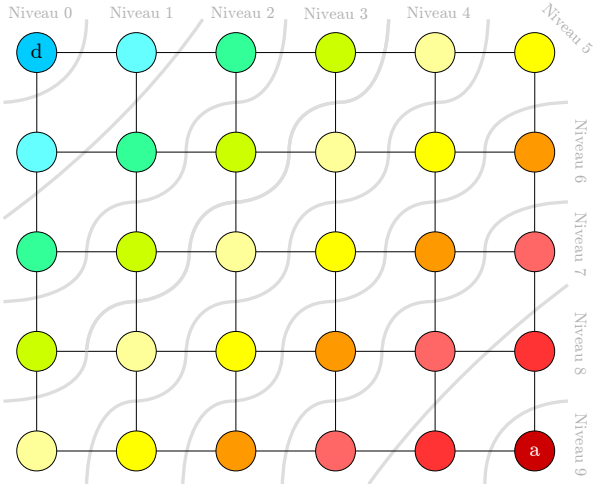
Comment demander son chemin quand on est un email ?

Un cas simple : le parcours en largeur



- À l'évidence, un chemin le plus court n'est pas « la ligne droite »
  - — On doit suivre les liens
- Supposons, pour simplifier, que le temps mis pour un paquet de voyager entre deux nœuds connectés soit le même pour tout le réseau
- En pratique, certains liens seront congestionnés
  - — Le trafic sera lent sur ces liens
  - — On ne voudra pas y passer
- Utilisation de l'algorithme du *parcours en largeur*
- Élaboré par E. F. Moore en 1959

Exemple d'exécution



Essayez avec les exemples papier !

- Utilisez la boîte à tuiles pour la file
- ... et les post-it pour les distances
- Trouvez la longueur d'un plus court chemin de d à a

Principe de l'algorithme de Dijkstra

- Élaboré par E. W. Dijkstra en 1956

Propriété d'un plus court chemin

- Si $x \rightsquigarrow y \rightsquigarrow z$ est un plus court chemin de x à z, alors
  1. $x \rightsquigarrow y$ est un plus court chemin de x à y
  2. $y \rightsquigarrow z$ est un plus court chemin de y à z
- Si ce n'était pas le cas, on aurait fait un détour dans le chemin $x \rightsquigarrow y \rightsquigarrow z$

Principe

- On va calculer les plus courts chemins de proche en proche
- On établit une frontière
  - — à l'intérieur de la frontière, les plus courts chemin ont été calculés
  - — à l'extérieur, ils ne le sont pas
- À chaque étape, on ajoute le nœud le plus proche de la frontière
  - — Si on ajoute la destination à l'intérieur de la frontière, on a terminé

Algorithme

1: $d_{dist} \leftarrow 0$
2: **pour chaque** nœud n différent de d **faire**
3: $\quad n_{dist} \leftarrow \infty$
4: Seul d est à l'intérieur de la frontière
5: **tant que** tous les nœuds ne sont pas dans la frontière **faire**
6: $\quad$ **pour chaque** connexion (p, q) traversant la frontière **faire**
7: $\quad\quad$ **si** $p_{dist} + \text{Poids}(p, q) < q_{dist}$ **alors**
8: $\quad\quad\quad q_{dist} \leftarrow p_{dist} + \text{Poids}(p, q)$
9: $\quad\quad\quad$ ajouter q dans la frontière

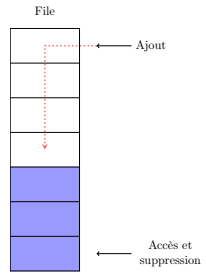
- La ligne 7 cherche à savoir s'il est plus intéressant de passer par p pour aller vers q à partir de d que ce qui était précédemment calculé
- Si tel est le cas, on met à jour la longueur du plus court chemin sous-jacent

Principe du parcours en largeur

- Algorithme d'exploration de proche en proche
- On se déplace « par niveaux » dans le graphe
  1. On visite un nœud ;
  2. On visite chacun des voisins du nœud ;
  3. On visite chacun des voisins des voisins du nœud, etc.

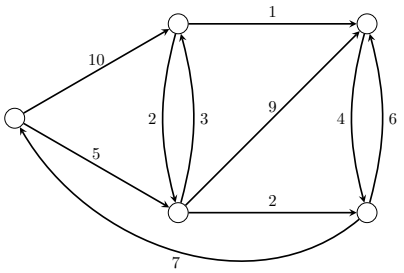
Algorithme

1: $F \leftarrow$ file vide
2: Mettre le départ d dans F
3: $d_{dist} \leftarrow 0$ (d est visité)
4: **tant que** F n'est pas vide **faire**
5: $\quad n \leftarrow$ nœud en tête de file
6: $\quad$ Retirer n de la file
7: $\quad$ **si** n n'est l'arrivée **alors**
8: $\quad\quad$ **pour chaque** voisin v non visité de n **faire**
9: $\quad\quad\quad$ Mettre v dans F, à la fin (v est visité)
10: $\quad\quad\quad v_{dist} \leftarrow n_{dist} + 1$
11: $\quad$ **sinon**
12: $\quad\quad$ **retourner** n_{dist}



Un cas plus complexe : l'algorithme de Dijkstra

- Rappel : en pratique, certains liens sont congestionnés
  - — « Embouteillages » : on veut éviter !
- Il faut adapter le modèle de graphe



- Flèches : certains liens ne peuvent être empruntés que dans un sens

Illustration