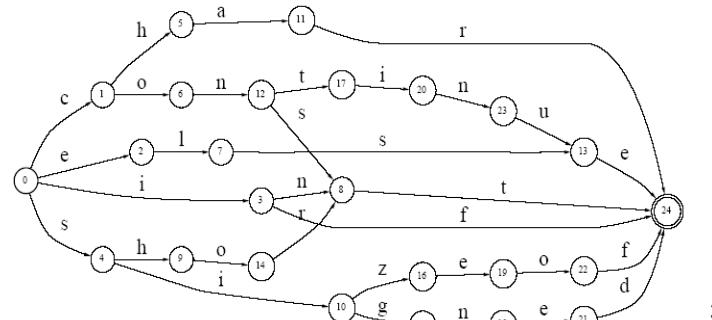


Graphes

Graphe

- Théoriquement la meilleure implémentation pour coder un lexique
 - Optimisation en termes de taille de stockage
- Quels mots sont représentés ci-dessous ?



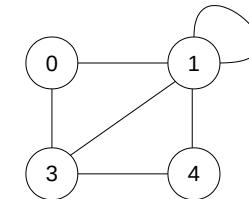
Graphe

- Structure de données qui permet de représenter un réseau de données
 - Réseaux de télécommunication
 - Réseaux sociaux
 - Réseaux routiers
 - Carte
 - ...
- Objectifs
 - Représentation d'un graphe
 - Parcours de graphe
 - Quelques problèmes

2

Définitions

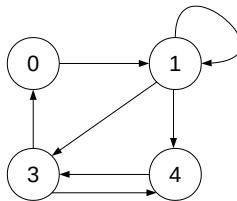
- Graphe : couple $G[X,A]$ où
 - X est un ensemble de nœuds ou sommets et
 - A est l'ensemble des paires de sommets reliés entre eux (« arêtes » du graphe ou « arcs »)
- Chemin = séquence d'arêtes menant d'un sommet i à un sommet j
- Circuit = chemin dont les sommets de départ et d'arrivée sont identiques
- Degré d'un sommet = nombre d'arêtes ayant ce sommet pour extrémité
- Voisins : les voisins des sommets sont ceux qui sont reliés à ce sommet par une arête



Graphes orientés

- Graphe dans lequel chaque arête a une direction associée

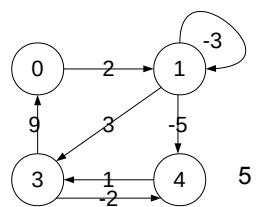
■ Arc = arête orientée



- Graphe orienté pondéré:

– Graphe étiqueté où chaque arc a un coût associé

- Valuation, coût = valeur numérique associée à un arc ou à un sommet

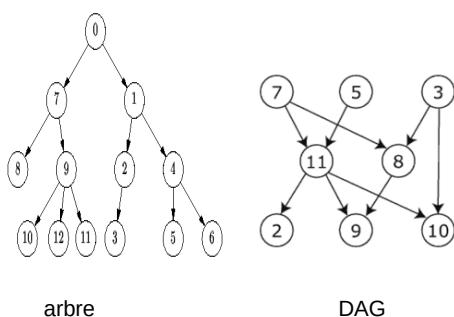


Opérations

- voisin : $G \times y \rightarrow \text{booléen}$
– Teste si x et y sont voisins
- voisins : $G \times x \rightarrow \text{liste}$
– Retourne la liste des voisins de x
- ajoute_noeud : $G \times x \rightarrow G$
– Ajoute le noeud x si x n'est pas déjà présent
- supprime_noeud : $G \times x \rightarrow G$
– Supprime x de G si x présent
- ajoute_arc : $G \times y \rightarrow G$
– Ajoute un arc entre x et y si absent
- supprime_arc : $G \times y \rightarrow G$
– Supprime l'arc entre x et y si présent
- valeur_arc : $G \times y \rightarrow N$
– Retourne la valeur de l'étiquette de l'arc entre x et y ; +oo sinon

7

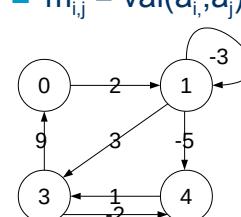
Types de graphe



6

Graphe : représentation matricielle

- Tout graphe $G = (X, A)$ peut être représenté par une matrice d'adjacence
- $m_{i,j} = \text{val}(a_i, a_j)$ si $(a_i, a_j) \in A$, ∞ sinon



Arrivée →	0	1	3	4
0	∞	2	∞	∞
1	∞	-3	3	-5
3	9	∞	∞	-2
4	∞	∞	1	∞

8

Graphe : représentation matricielle

- Complexité spatiale $\rightarrow O(|X|^2)$
- Coût d'ajout/suppression d'un nœud $\rightarrow O(|X|^2)$
- Coût d'ajout/suppression d'un arc $\rightarrow O(1)$
- Fortement déconseillé pour les graphes creux !
- Meilleur choix si la rapidité d'accès est cruciale ou si le graphe est très dense (\rightarrow complet)

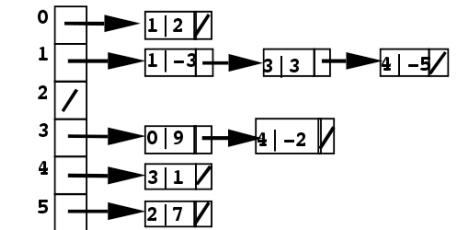
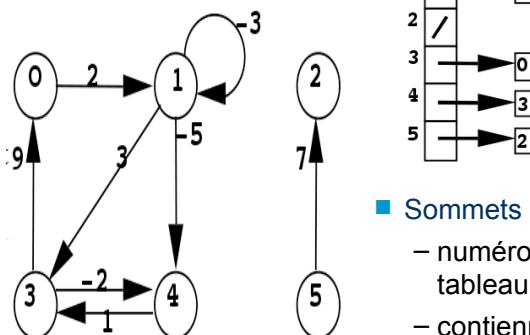
9

Graphe : représentation par liste

- Complexité spatiale $\rightarrow O(|X|+|A|)$
- Coût d'ajout d'un nœud $\rightarrow O(1)$
- Coût d'ajout d'un arc $\rightarrow O(1)$
- Coût suppression d'un nœud $\rightarrow O(|A|)$
- Coût suppression d'un arc $\rightarrow O(|A|)$
- Parfait pour représenter les graphes creux !
- Meilleur compromis de complexité

11

Graphe : représentation par liste d'adjacence



- Sommets
 - numérotés et stockés dans un tableau
 - contiennent une liste des arcs qui partent de ce sommet
- Arc
 - triplet <départ,arrivée,coût>

10

Parcours de graphe

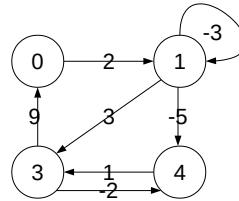
- Parcours non ordonnés :
 - Parcours du tableau de nœuds (p.ex., affichage des valeurs des nœuds)
 - Parcours des listes d'arêtes (p.ex., initialisation de poids)
- Parcours ordonnés selon les arêtes :
 - Problème de cycle (comment être sûr de ne pas passer plusieurs fois au même endroit ?)
 - Problème d'initialisation (début) et de terminaison (fin)
 - Problème de sens de parcours (tous les fils ou tous les frères d'abord ?)

12

Parcours en profondeur d'abord

■ Parcours_profondeur(graphe G, sommet X)

- Si X n'a pas encore été parcouru alors
 - Noter X parcouru
 - Pour tout voisin V de X :
 - parcours_profondeur(G,V)



■ Exemple:

■ Même algorithme que pour les arbres

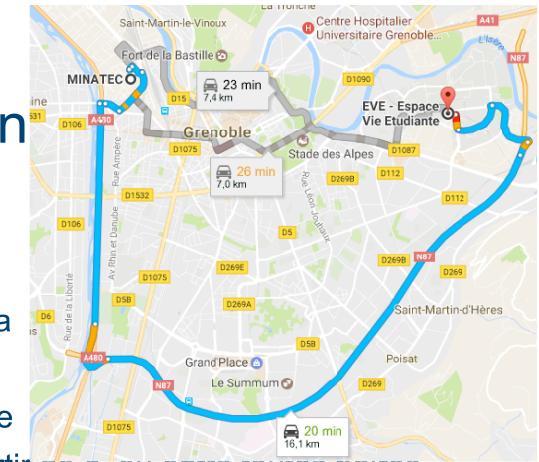
13

Problème : plus court chemin

■ Les noeuds sont les intersections

■ Les arcs sont valus par la distance ou le temps

- Un plus court chemin entre les noeuds d et a, ou à partir de a, ou entre toutes paires
 - Dijkstra $O(|X|^2)$ une seule source
 - Bellman $O(|X| |A|)$ une seule source
 - A* ($e^{|X|}$ pire cas ; $|X| \log |X|$ meilleur cas) une seule paire
 - Floyd-Warshall $O(|X|^3)$ toutes les paires



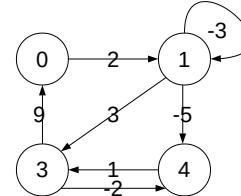
15

Parcours en largeur d'abord

■ Stockage des frères par file d'attente

■ Parcours_largeur(graphe G, sommet X)

- Noter X parcouru
- File F $\leftarrow \{X\}$
- Tant que $F \neq \emptyset$
 - N = défile(F)
 - Action(N) // exemple Afficher N.
 - Pour tout voisin V de N :
 - Si V non parcouru alors
 - » Noter V parcouru
 - » F = enfiler(F,V)

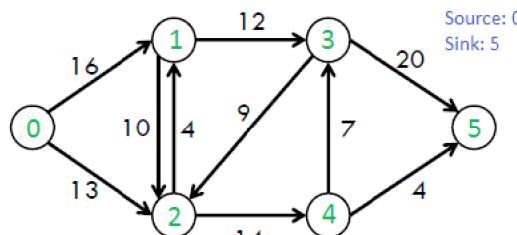


■ Exemple:

■ Même algorithme que pour les arbres

14

Exemple de problème : Max flow



■ Trouver le flux maximal entre une source et une cible (sink).

■ Résolution par l'algorithme de Ford-Fulkerson

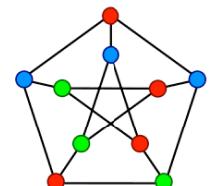
■ Exemples

- Flux de voitures entre deux villes selon un réseau
- EDF et les coupures possibles :)
- Optimiser le temps d'attente des remontées mécaniques

16

Problème : coloration de graphe

- Attribuer une couleur à chaque nœud de façon à ce qu'aucun nœud adjacent n'ait la même couleur
- Si on cherche le minimum alors on cherche le nombre chromatique du graphe
 - Utilisation en allocation de ressource minimale
 - Incompatibilité : des produits co-combustibles sont placés dans des wagons non voisins ==> de couleurs différentes



17

Projet 2019 : PPC

- Plus court chemin dans
 - Un réseau routier
 - Le métro/Rer parisien
 -
- Projet sur 4 séances
 - En binôme : bien s'organiser
 - Travail personnel entre séances indispensable
 - Pas de plagiat : ni interne, ni externe :-)
 - Des données réelles: 16 millions de sommets

19

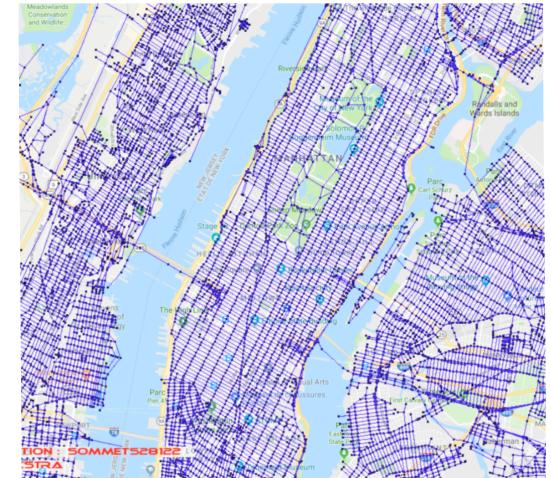
Et encore

- Test de planéité : un graphe est-il planaire ?
 - Routage et circuits imprimés
- Arbre couvrant de poids minimal d'un graphe : sous-ensemble qui connecte tous les sommets dont la somme des poids des arêtes est minimale
 - Réseaux informatiques
- Ordonnancement et méthode PER, MPM
 - Ordonnancement des tâches

18

Projet 2018 : PPC

- Réseau routier USA
 - Tous les arcs = toutes les routes
 - Pas de noms de sommets réels
 - Coordonnées GPS des sommets



20

Représentation du graphe

- Les sommets

```
typedef struct {
    char* nom;
    double x,y ;
    L_ARC voisins;}
```

T_SOMMET ;

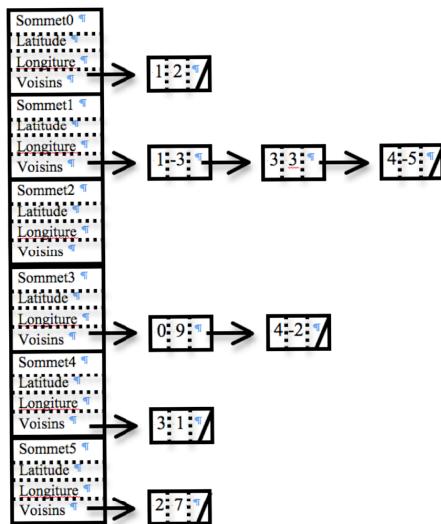
- Les arcs

```
typedef struct {
    int arrivee;
    double cout } T_ARC ;
```

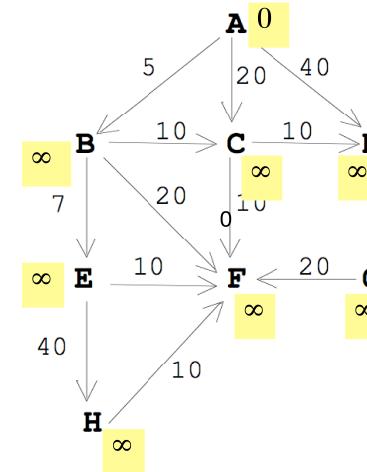
- Les listes de successeurs :

```
typedef struct lsucc {
    T_ARC val;
    struct lsucc* suiv ; }*
```

L_ARC;



A* avec $H(s,a)=0$;



Recherche du Chemin de A à F

$X = \{A,B,C,D,E,F,G\}$

Liste fermée = sommets atteints et PCC connu

- LF = {}

Liste ouverte : sommets dont un voisin a été atteint .

- LO = { A }

$F(s) = G(s) + H(s,d)$

- $G(s)$: cout du PPC de a à s

- $H(s,d)$: évaluation empirique du cout pour aller de s à d

Cout F={0,∞,∞,∞,∞,∞,∞}

Cout plus court G={0,∞,∞,∞,∞,∞,∞}

■ Cout évalué H={0,0,0,0,0,0,0}

23

Format du fichier

- Nb sommets, Nb arcs

- Une ligne inutile

- Tous les sommets

Numéro Latitude Longitude
Ligne Nom

- Une ligne inutile

- Les arcs

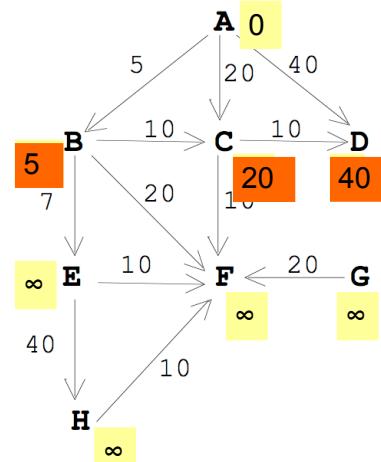
Départ Arrivée Coût

```

8 12
Sommets du graphe
0 0.5 0.95 M1      Aaa
1 0.1 0.7 M1       Baa
2 0.5 0.7 M1       Caa
3 0.9 0.7 M1       Daa
4 0.1 0.35 M1      Eaa
5 0.5 0.35 M1      Faa
6 0.9 0.35 M1      Gaa
7 0.1 0.05 M1      Haa
Arêtes du graphe : noeud1 noeud2 valeur
0 1 5
0 2 20
0 3 40
1 2 10
1 4 7
1 5 20
2 3 10
2 5 10
4 5 10
4 7 40
6 5 20
7 5 10
laptop-235:src2 desvigrm$ 
laptop-235:src2 desvigrm$ 
    
```

22

Etape 1



■ F={0,∞,∞,∞,∞,∞,∞}

■ LF = {}

■ LO = { A }

- Recherche du meilleur sommet de LO

- ⇒ Sommet A

- LF={A} // A est atteint avec un cout de 0
- LO={}

- Mise à jour des voisins de A : B,C,D

- Pour B : pas dans LO

- $G(B)=G(A)+cout(A,B)=0+5$
- $F(B)=G(B)+H(B,F)=5+0$
- LO = {B}

- Pour C : pas dans LO

- $G(C)=G(A)+cout(A,B)=0+20$
- $F(C)=20$
- LO = {C,B}

- Pour D : pas dans LO

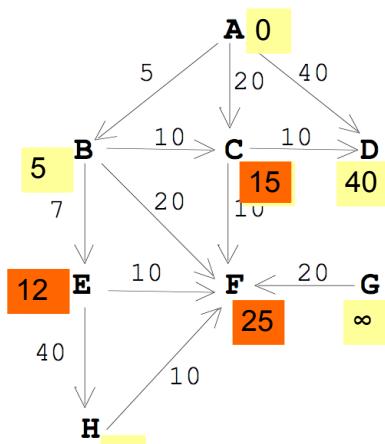
- $G(D)=G(A)+cout(A,B)=0+40$
- $F(D)=40$
- LO = {D,C,B}

■ F={0,5,20,40,∞,∞,∞}

24

Etape 2

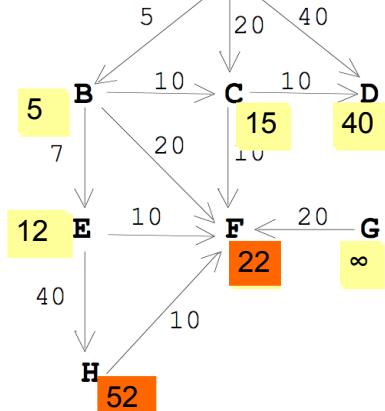
- $F=\{0,5,20,40,\infty,\infty,\infty\}$
- $LF = \{A\}$
- $LO = \{ D,C,B \}$
- Recherche du meilleur sommet de LO
 - > Sommet B
 - $LF=\{B,A\}$ // B est atteint avec un cout de 5
 - $LO=\{D,C\}$
 - Mise à jour des voisins de B : C,E,F
 - Pour C : déjà dans LO
 - $G(C) > G(B)+cout(B,C)=5+10 = \text{Meilleur chemin}$
 - $F(C)=G(C)=15$
 - $LO = \{D,C\}$
 - Pour E : pas dans LO
 - $G(E)=G(B)+cout(B,E)=5+7=12$
 - $F(E)=12;$
 - $LO = \{E,D,C\}$
 - Pour F : pas dans LO
 - $G(F)=G(B)+cout(B,F)=5+20=25$
 - $F(F)=25$
 - $LO = \{F,E,D,C\}$
 - $F=\{0,5,15,40,12,25,\infty,\infty\}$



25

Etape 3

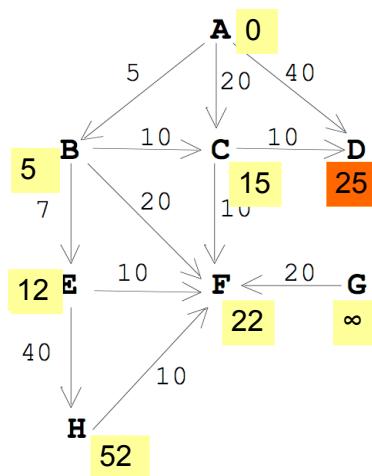
- $F=\{0,5,15,40,12,25,\infty,\infty\}$
- $LF = \{B,A\}$
- $LO = \{ F,E,D,C \}$
- Recherche du meilleur sommet de LO
 - > Sommet E
 - $LF=\{E,B,A\}$ // E est atteint avec un cout de 12
 - $LO=\{F,D,C\}$
 - Mise à jour des voisins de E : F,H
 - Pour F : déjà dans LO
 - $G(F) > G(E)+cout(E,F)=12+10 = \text{Meilleur chemin}$
 - $F(F)=G(F)=22$
 - $LO = \{F,D,C\}$
 - Pour H : pas dans LO
 - $G(H)=G(E)+cout(E,H)=12+40=52$
 - $F(H)=52;$
 - $LO = \{H,F,D,C\}$
 - $F=\{0,5,15,40,12,22,\infty,52\}$



26

Etape 4

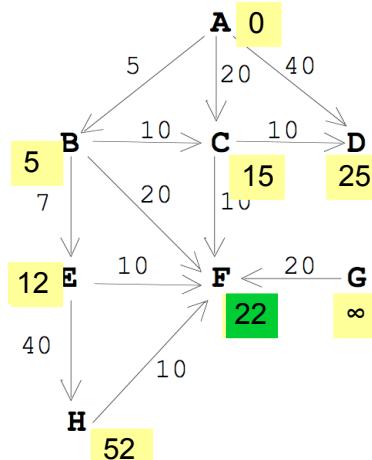
- $F=\{0,5,15,40,12,22,\infty,52\}$
- $LF = \{E,B,A\}$
- $LO = \{ H,F,D,C \}$
- Recherche du meilleur sommet de LO
 - > Sommet C
 - $LF=\{C,E,B,A\}$ // C est atteint avec un cout de 20
 - $LO=\{H,F,D\}$
 - Mise à jour des voisins de C : F,D
 - Pour F : déjà dans LO
 - $G(F) < G(C)+cout(C,F)=15+10=25$
 - Pas de changement
 - $LO = \{H,F,D\}$
 - Pour D : déjà dans LO
 - $G(D)>G(C)+cout(C,D)=15+10=25$
 - $F(D)=25;$
 - $LO = \{H,F,D\}$
 - $F=\{0,5,15,25,12,22,\infty,52\}$



27

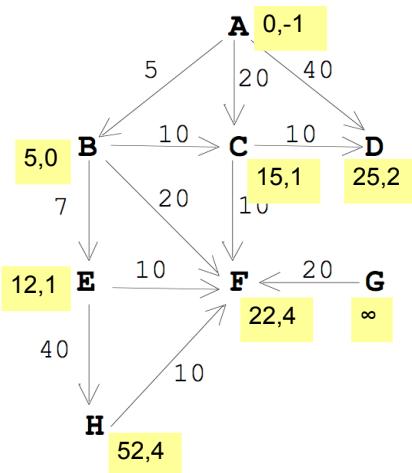
Etape 5

- $F=\{0,5,15,25,12,22,\infty,52\}$
- $LF = \{C,E,B,A\}$
- $LO = \{ H,F,D \}$
- Recherche du meilleur sommet de LO
 - > Sommet F
 - F est atteint avec un cout de 22
 - FIN



28

A* : le chemin



- $F=G=\{0,5,15,25,12,22,\infty,52\}$
- Etape de mise à jour : ajout de l'information du sommet dont on vient quand on change la valeur de G d'un voisin d'un sommet atteint
- Sur le schéma, pour B
 - 5 est la valeur du chemin
 - 0 est l'indice du sommet par lequel on arrive sur 5 pour obtenir le PCC en numérotant les sommets $\{A,B,C,D,E,F,G,H\}$ de 0 à 7.
- Retrouver le chemin de A à F
 - Plus court chemin de F
 - $Pere[5] = 4$
 - Chercher le père du sommet tant qu'il existe
 - Père de F : E (4)
 - Père de E : B (1)
 - Père de B : A (0)
 - C'est le départ !!
 - Donc le chemin est A,B,E,F

29

TODO

- Définir les représentations
 - Comment représenter un sommets avec les coûts et les pères?
 - Comment représenter le graphe ?
 - Quelles structures de données pour LO et LF ?
- Définir les fonctions utiles
 - Commencer par les fonctions les plus basiques
 - Listes, lecture du graphe, affichage du graphe
 - Tester au fur et à mesure
 - Tester d'abord sur des graphes simples
- Cas particulier du métro
 - Correspondance incluse dans le fichier
 - Gestion des sommets grâce à leur nom
 - Attention : plusieurs sommets ont le même nom
 - Si on part de la gare du Nord, il y a 4 sommets de départ possibles
- Structure de données utiles pour optimiser
 - Tas de sommets : trouver le min de LO
 - Table de hachage : trouver le numéro d'un sommet à partir d'une chaîne de caractères

30