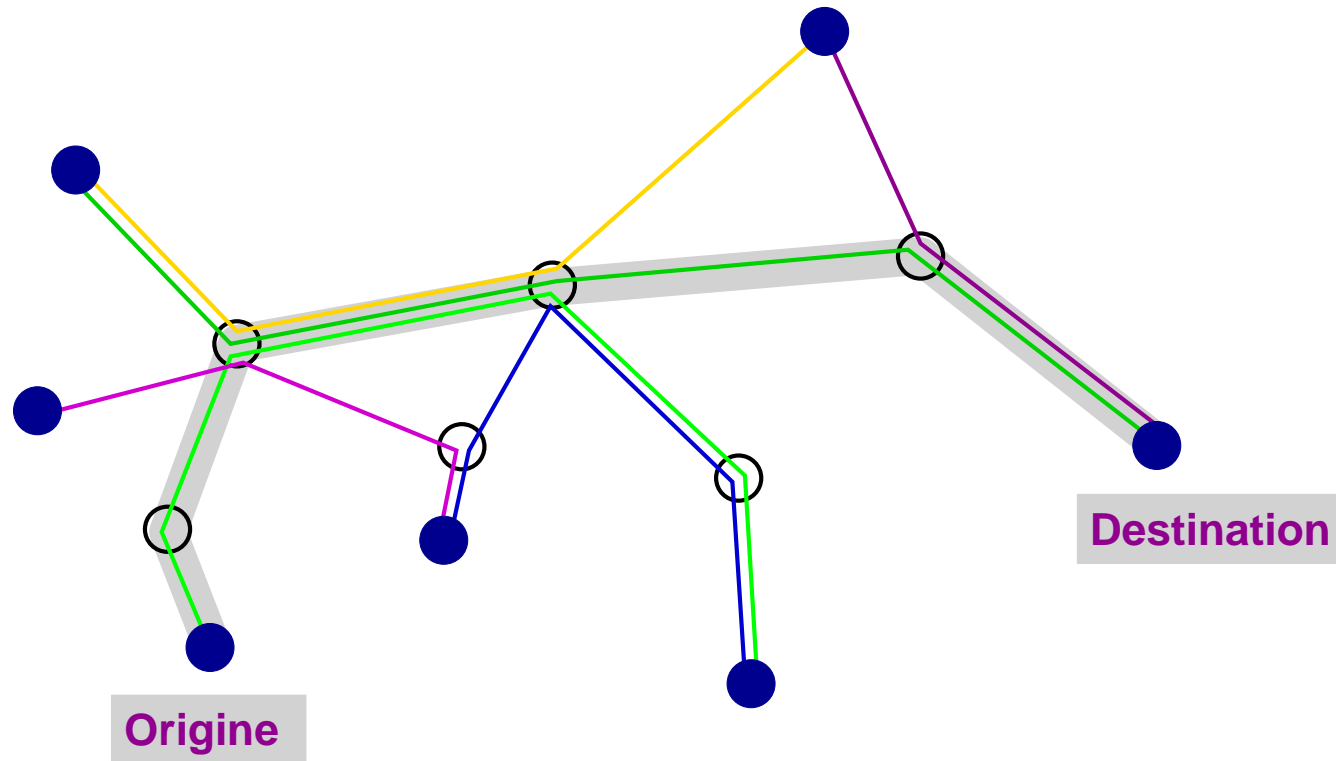


Calcul d'itinéraires ferroviaires.



Déroulement du projet

- Etape 1 : **calcul d'itinéraires ferroviaires.**
 - lecture des fichiers de données
 - construction des structures de données
 - calcul d'itinéraires optimaux (algorithme de Dijkstra)
 - **17 février** : 1er rapport intermédiaire
- Etape 2 : **tests, durées moyennes, clustering.**
 - génération aléatoire d'instances et tests
 - calcul des durées moyennes de trajet entre deux villes
 - problème de clustering
 - interface graphique (optionnel)
 - **14 avril** : rapport final

Evaluation

Le travail est à faire seul ou par groupe de deux (**en aucun cas par groupe de trois ou plus**).

Le rapport intermédiaire et le rapport final donnent lieu à une **présentation écrite** et **orale** du travail effectué incluant une description succincte des **structures de données** utilisées et des **principales fonctions**.

Rapport intermédiaire : 2 pages et 10 min. de présentation.

Rapport final : 5 pages et 15 min. de présentation.

Lecture des données.

Syntaxe du fichier de données

```
<nombre de villes n>  
<abscisse ville 1> <ordonnée ville 1>  
...  
<abscisse ville n> <ordonnée ville n>  
  
<nombre de lignes m>  
<definition de la ligne 1>  
...  
<definition de la ligne m>
```

Lecture des données.

Définition d'une ligne :

```
<nombre de villes k>  
<ville 1> <ville 2> ... <ville k>  
  
<nombre de passages journaliers p>  
<P1 horaire 1> <P1 horaire 2> ... <P1 horaire k>  
...  
<Pp horaire 1> <Pp horaire 2> ... <Pp horaire k>
```

Lecture des données : exemple.

```
10
383 886 777 915 793 335 386 492 649 ... 763 926 540
426 172 736

2
3
1 0 9
2
14h44 18h41 21h16
03h30 07h27 10h02
3
3 8 6
3
20h33 22h13 02h10
05h02 06h42 10h39
12h50 14h30 18h27
```

Codage des horaires : nombre de minutes (par ex. depuis minuit, le jour du départ)

Algorithme de Dijkstra

Calcul de la durée du trajet optimal entre la ville s et la ville t .

Fixer $d[v] \leftarrow \infty$ pour tout $v \in V$ et $d[s] = 0$

Placer tous les sommets $v \in V$ dans un tas T ordonné selon $d[v]$.

$S \leftarrow \emptyset$, l'ensemble des sommets traités est vide

Tant que T n'est pas vide faire

Extraire de T le sommet u de marque $d(u)$ minimum

Ajouter u à l'ensemble S des sommets traités.

Pour chaque voisin v de u faire

Si $d[v] > d[u] + l(u, v)$ alors

$d[v] \leftarrow d[u] + l(u, v)$, mettre à jour le tas T

$pere[v] = u$

Renvoyer $d[t]$

Construction des structures de données.

Déf. On dit qu'une ville v' est un **successeur** d'une ville v si un arrêt en v' intervient immédiatement après un arrêt en v sur **au moins** une ligne.

Structure de données.

- pour chaque ville v , calculer **la liste des successeurs** de v .
- pour chaque successeur v' de v , calculer **la liste des trajets** de v à v' **ordonnée par dates de départ croissante** ;
- un **trajet** est caractérisé par un **horaire de départ** et un **horaire d'arrivée** et le numéro de ligne.

Première étape

Ecrire un programme qui lit les **fichiers de données** au format indiqué et qui calcule les **structures de données**.

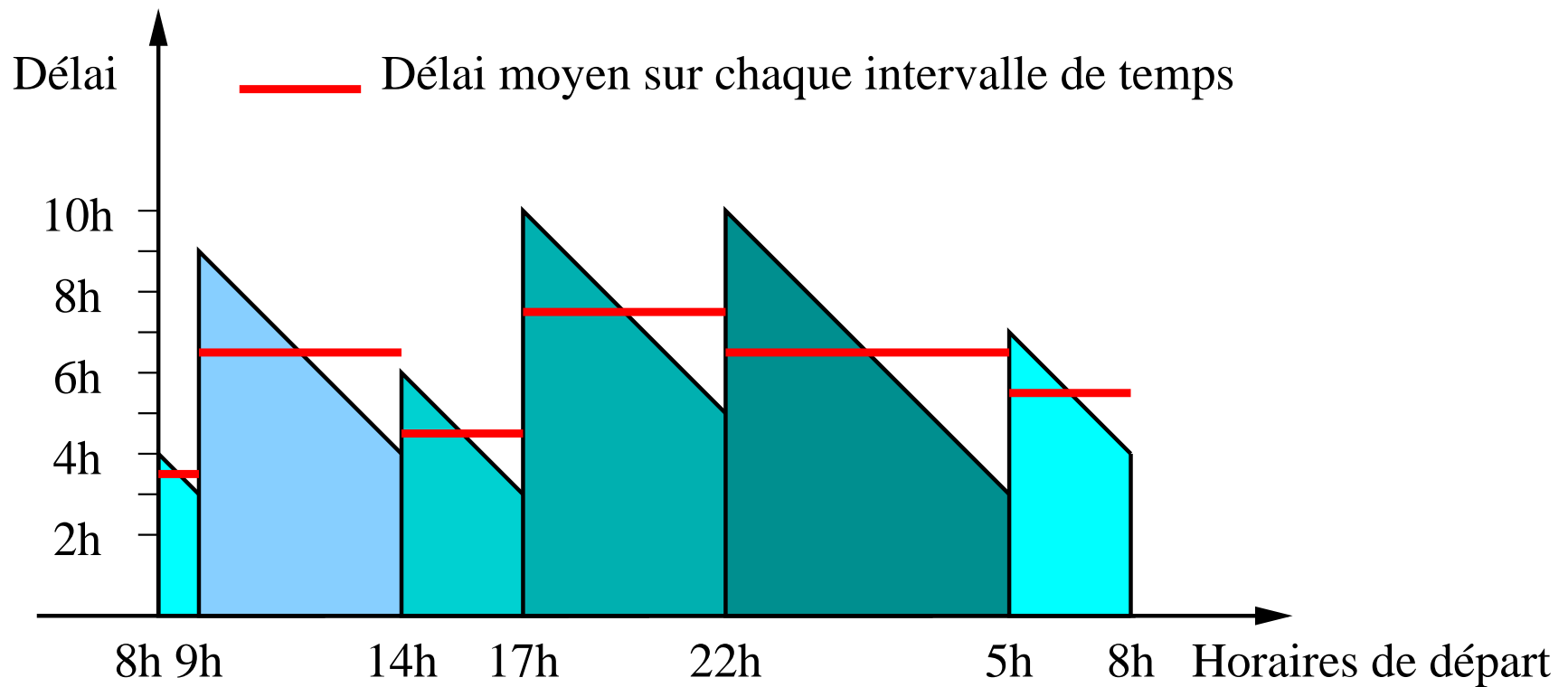
Problème : On se trouve dans une ville s à l'heure h , quel itinéraire suivre pour se rendre le plus vite possible dans une ville t ?

Algorithme : modifier la signification des marques dans l'algorithme de Dijkstra, i.e. la marque d'une ville v sera l'heure d'arrivée au plus tôt en v en partant de s à l'heure h .

Le programme devra permettre de calculer **la date d'arrivée au plus tôt en t** et d'**afficher un itinéraire optimal** (listes de villes traversées et horaires).

Calcul de la durée moyenne de trajet

On veut calculer la **durée moyenne** $m(v, u)$ du trajet pour se rendre de la ville v à la ville u . Cette durée inclut le **temps d'attente** dans chacune des villes intermédiaires et dans la ville de départ.



Départs : 9h, 14h, 17h, 22h, 5h

Temps trajets : 3h, 4h, 3h, 5h, 3h

Calcul de la durée moyenne de trajet

Soient $h_0 = 0$, $h_n = 24$ et soient h_1, \dots, h_{n-1} les **horaires de départ** des trains à partir de la ville de départ.

Entre deux départs de trains consécutifs h_{i-1} et h_i , le temps de trajet diminue de façon **linéaire**, la **durée moyenne** pendant cet intervalle est donc

$$m_i = \text{durée}(h_i) + (h_i - h_{i-1})/2.$$

La **durée moyenne** sur 24 heures est donnée par

$$m(u, v) = \frac{\sum_{i=1}^n m_i (h_i - h_{i-1})}{h_n - h_0}.$$

Votre programme devra permettre de **calculer et d'afficher la durée moyenne** entre deux villes u et v données.

Distance

On définit **la distance** $d(u, v)$ entre deux villes u et v de la façon suivante :

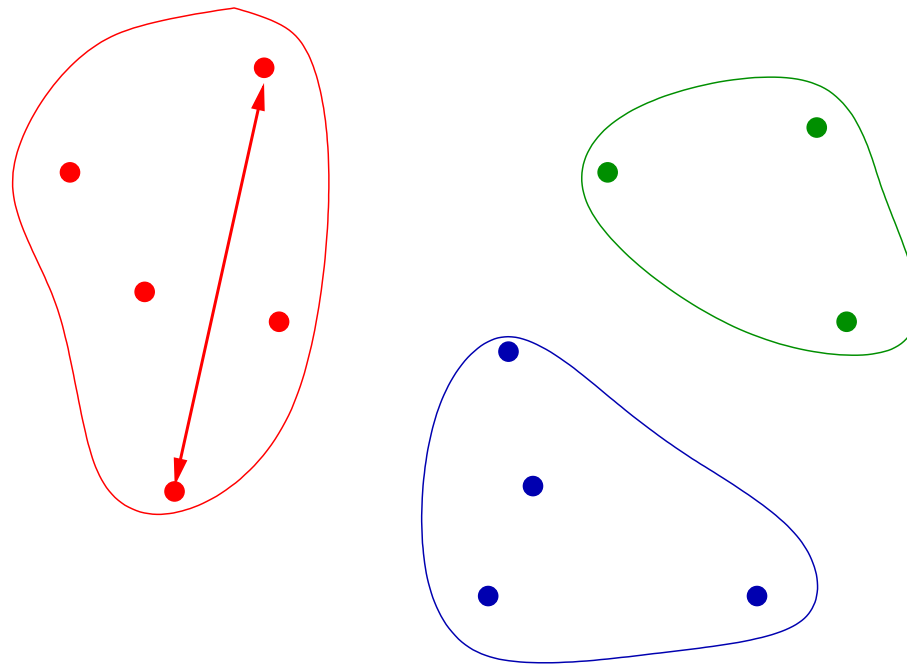
$$d(u, v) = \frac{m(u, v) + m(v, u)}{2}$$

Votre programme devra calculer la **matrice des distances** entre chaque paire de villes.

Remarque. Cette distance est définie de façon à être **symétrique**, i.e. $d(u, v) = d(v, u)$ ce qui n'est pas le cas pour $m(u, v)$.

Clustering

Pour un entier k donné, on veut partitionner les villes en k clusters V_1, V_2, \dots, V_k de telle sorte que la plus grande distance entre deux villes d'un même cluster soit la plus petite possible.



Clustering : solution initiale

L'**algorithme** suivant permet d'obtenir une solution (**pas nécessairement optimale**) :

- Trouver deux villes v_1 et v_2 à distance maximum l'une de l'autre.
- Placer v_1 dans V_1 et v_2 dans V_2 .
- Pour $i = 3, \dots, k$ faire
 - Trouver une ville v_i à distance maximum des villes $\{v_1, \dots, v_{i-1}\}$.
 - Placer la ville v_i dans un nouveau groupe V_i .
- Pour chaque ville $v \notin \{v_1, \dots, v_k\}$, placer v dans le groupe de la ville v_i la plus proche de lui.

Déf. La distance entre une ville v et un sous-ensemble S de villes est $d(v, S) = \min\{d(v, u) : u \in S\}$.

Clustering : recherche locale

Méthode :

- construire une solution initiale (aléatoirement ou avec le premier algorithme).
- tenter d'améliorer cette solution en déplaçant une ville d'un groupe à un autre.
- recommencer

Stratégie :

- recherche du meilleur déplacement
- recherche (aléatoire) d'un déplacement améliorant

Génération du réseau.

fonction **genereReseau**(n, d, m, min, max)
(génère m lignes de trains contenant entre min et max gares)
(dans un réseau ferroviaire de n gares et de densité d)

début

générer aléatoirement les positions géographiques des villes,
calculer les distances euclidiennes entre les villes,

$n_{ar} := d \times n \times (n - 1) / 200,$

$A := \emptyset,$

pour $i := 1$ à n_{ar} **faire**

choisir une ville u au hasard (non saturée),

soit v la ville la plus proche telle que $(u, v) \notin A,$

$A := A \cup \{(u, v)\},$

fin faire

calculer les longueurs D des plus courts chemins dans $A,$

pour $i := 1$ à m **faire**

$L[i] := \text{genererLigne}(min, max, A, D, n),$

fin faire

renvoyer $L,$

fin fonction

Génération d'une ligne.

fonction **genererLigne**(min, max, A, D, n)

Data : n le nombre de villes, A un réseau ferroviaire,

Result : une ligne contenant entre min et max gares.

begin

DEBUT :

choisir deux villes a et b au hasard;

choisir la longueur l au hasard, telle que $min \leq l \leq max$;

$i := 0$;

$T[i] := a$;

while $i < l$ et $T[i] \neq d$ **do**

┌ choisir au hasard une ville v voisine de $T[i]$ telle que $D(v, b) < D(T[i], b)$;

└ $i := i + 1$;

└ $T[i] := v$;

if $i < l$ **then**

└ **goto** DEBUT;

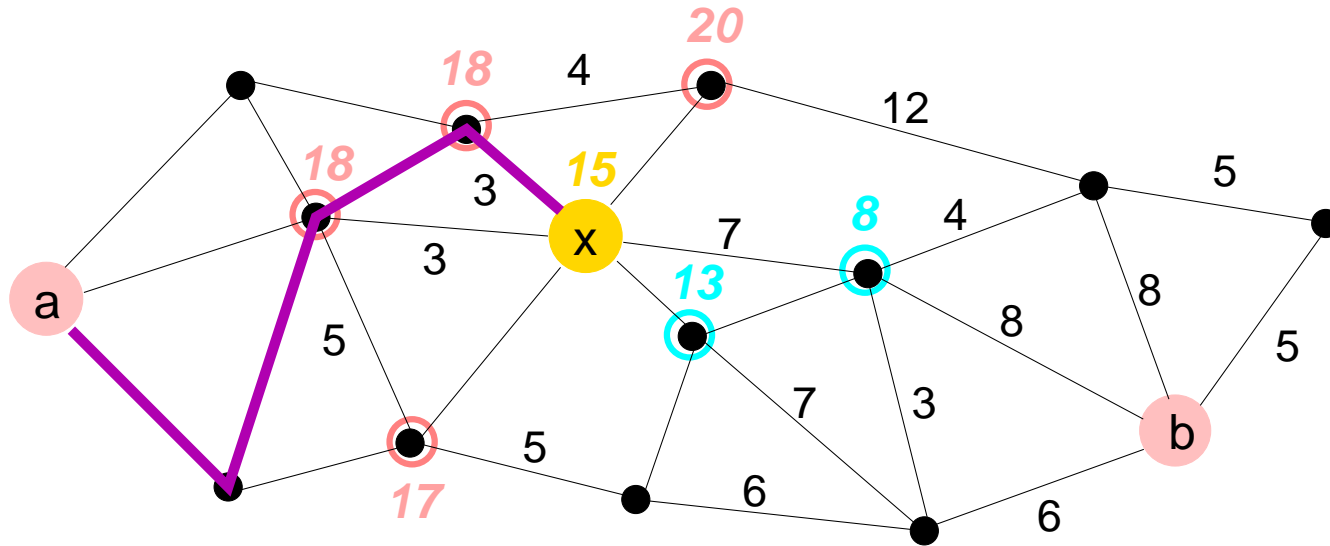
choisir h le nombre de passages par jour;

générer h horaires de passage H ;

return (T, l, H, h);

end

Génération aléatoire.



Challenge

- temps limite de calcul de 600 secondes par problème
- mémoire limitée à 1 Go
- jeu de problèmes donné
- modalités à venir

Critères d'évaluation et Barème

● Critères d'évaluation :

- travail effectué (calcul d'itinéraires, durées moyennes, clustering, interface),
- développement : lisibilité, simplicité du code, structuration du projet,
- algorithmes : compréhension, amélioration des méthodes,
- qualité de la présentation

● Barème :

- lecture (3pts), structure de données (3pts), Dijkstra (6pts)
- générateur (2pts), calcul des durées moyennes (2pts), clustering (2pts), interface graphique (2pts)

Interface graphique.

`http ://pageperso.lif.univ-mrs.fr/ edouard.thiel/ez-draw/index.html`

Présentation à venir