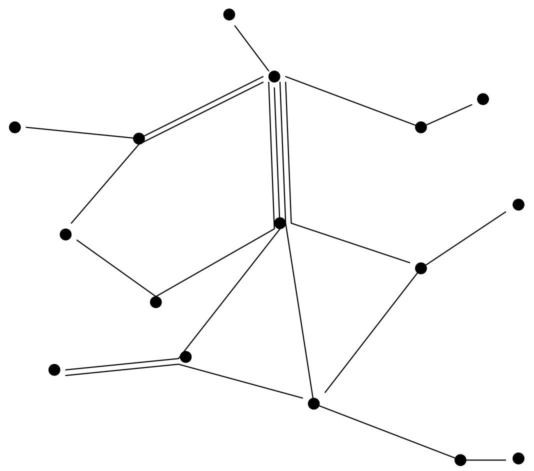
Problèmes de routage dans un réseau ferroviaire.

Description du problème.

Un réseau ferroviaire est constitué d'un ensemble de gares, de chemins de fer reliant ces gares et de lignes de trains. Une ligne est constituée de la liste des gares visitées pendant le trajet et des horaires d'arrivée et/ou de départ dans ces gares. Nous allons considérer plusieurs problèmes qui consistent à identifier des itinéraires optimaux dans un réseau ferroviaire. Tout d'abord on s'intéressera aux itinéraires optimaux pour aller d'une ville à une autre, ensuite on cherchera les meilleurs parcours d'un ensemble de villes avec retour au point de départ.



Les réseaux ferroviaires seront définis dans des fichiers textes. Pour simplifier vous considèrerez qu'un train part d'une ville intermédiaire au moment où il y arrive (on ne tient pas compte du temps d'arrêt en gare). Vous respecterez la syntaxe suivante :

```
<nombre de villes>
<nombre de lignes>
<definition de la ligne 1>
...
<definition de la ligne m>
```

où une ligne est définie de la façon suivante

```
<nombre k de villes traversées>
<ville 1> <ville 2> ... <ville k>
<nombre p de passages sur la ligne (par jour)>
<passage 1> <passage 2> ... <passage p>
```

où un passage est défini par les horaires de départ de chacune des villes et l'horaire d'arrivée en la dernière ville (NB une ligne est définie par la suite des villes parcourues et par les différents horaires de parcours ; on ne s'occuppe pas du parcours inverse qui constituera une autre ligne)

```
<heure depart ville 1> <heure depart ville 2> ...
             <heure depart ville p-1> <heure arrivee ville k>
```

Exemple. Les commentaires sont entre accolades et n'apparaîtront pas dans les fichiers texte que vous manipulerez.

```
5  {Nombre de villes ; elles sont numérotées 0, 1, 2, 3, 4}
4  {nombre de lignes}

{definition de la ligne 1}
  3  {nombre de villes pour la ligne 1}
  0 3 1 2 {les villes de la ligne dans l'ordre de visite}
  2  {le nombre de passages (d'horaires) dans la journee}
    09h43 11h43 16h13 18h20 {passage 1}
    13h30 15h30 20h00 21h30 {passage 2}

{definition de la ligne 2 (d'ailleurs le retour de la ligne 1)}
  3  {nombre de villes pour la ligne 2 }
  2 1 3 0  {les villes de la ligne dans l'ordre de visite}
  3  {le nombre de passages (d'horaires) dans la journee}
    08h00 12h30 14h30 14h56 {passage 1}
    16h20 20h50 22h50 23h14 {passage 2}
    20h30 01h00 02h00 02h47 {passage 3}

{definition de la ligne 3}
...
{definition de la ligne 4}
...
```

Problème 1 : recherche d'un itinéraire optimal.

Le problème est de répondre aux requêtes des passagers qui veulent se rendre d'une ville à une autre en partant au plus tôt à une heure h. Ils veulent biensûr arriver le plus tôt possible. Pour ce problème vous adapterez l'algorithme de Dijkstra vu en cours pour calculer le meilleur itinéraire. Remarquez que vous aurez besoin pour chaque gare, de l'ensemble des trains en partance avec leurs heures de départs. Les sommets voisins d'une gare g sont les gares directement accessibles (i.e. sans gares intermédiaires). Les poids des arêtes correspondantes sont les durée minimales pour atteindre ces gares calculées à partir de l'instant où on a atteint g.

Ce que vous devez faire.

1. définir des structures de données adaptées aux réseaux ferroviaires ; il faut d'une part représenter le réseau mais en plus simplifier les opérations élémentaires dont vous aurez besoin par la suite ; notamment, il est utile d'accéder pour une gare donnée aux trains qui en partent et aux horaires correspondants,
2. écrire un programme indépendant de génération aléatoire de réseaux (rien n'oblige à utiliser dans ce programme les structures de données que vous aurez définies pour représenter les réseaux),
3. à ce stade vous pourrez développer la représentation graphique du réseau ; pour çela il faudra ajouter les positions géographiques des gares dans la génération,
4. **Problème 1 :** écrire un programme qui, à partir d'un réseau R, de deux villes a et b et d'un horaire de départ h calcule le meilleur itinéraire pour aller de la ville a à la ville b en partant au moins à l'heure h,
5. **Problème 2 : quand vous aurez fini et validé les trois étapes précédentes**, écrire un programme qui, à partir d'un réseau R, d'une ville de départ d, d'un horaire de départ h et d'un ensemble de villes V calcule un bon parcours (voire le meilleur) pour visiter toutes les villes de V en partant de la ville d au moins à l'heure h et revenir en d (dans ce cas un bon parcours est un parcours pour lequel on arrive au plus tôt). *Voir les indications dans le dernier chapitre de ce document.*

Développez votre projet en suivant cet ordre. Il est **absolument obligatoire** d'avoir un générateur aléatoire et des structures de données simples et bien adaptées avant de passer au calcul d'itinéraires et de parcours.

Génération des problèmes.

Toutes les parties de ce projet doivent être validées au fur et à mesure tout d'abord sur de petits exemples pour vérifier que les résultats sont corrects, puis sur des exemples de grandes tailles (par exemple une centaine de villes et des parcours de 20-30 villes) pour vérifier l'efficacité et la pertinence des choix que vous aurez fait.

Pour la génération de réseaux, commencez par générer des positions géographiques aléatoires des villes dans le plan, ensuite connectez entres elles les villes qui sont proches au sens de la distance Euclidienne. Vous suivrez la procédure suivante

procédure **GénérationDeLaCarteDesVoies**(D, n, m, C)
(**In :** D les distances Euclidiennes entre les n villes)
(**Out :** C la carte constituée de m voies.)
    $C := \emptyset$,
    tant que $(m > 0)$ **faire**
        Choisir une ville a au hasard,
        Soit b la ville la plus proche de a telle que $(a, b) \notin C$,
        $C := C \cup \{(a, b)\}$,
        $m := m - 1$,
    fin faire,
    renvoyer C,

Ensuite il vous faudra générer des lignes. Tout d'abord calculez les plus courts chemins entre tous les couples de gares (Warshall-Floyd) dans le graphe C, puis pour chaque ligne à générer choisissez deux villes au hasard a et b. La ligne de a en b est par exemple le parcours correspondant au plus court chemin entre a et b. Vous pouvez aussi la générer en partant *à l'aveugle* de la ville a et en choisissant chaque fois la ville suivante au hasard parmi celles qui rapprochent strictement de la ville b (au sens des longueurs des plus courts chemins dans C) (ça fait des itinéraires beaucoup plus jolis).
Les horaires des lignes seront calculés en choisissant une heure de départ au hasard, puis en calculant les horaires intermédiaires à partir d'une vitesse donnée et constante du train.

Problème 2 : recherche d'un parcours optimal.

Calcul d'un parcours optimal.

Vous allez appliquer la recherche énumérative décrite ci-dessous à la recherche d'un parcours optimal (au sens de l'heure de retour d'un voyageur qui effectuerait ce parcours) d'un ensemble de villes donné $\mathcal{V}$ avec retour au point de départ et avec départ à une heure donnée (type voyageur de commerce mais avec la contrainte des correspondances de trains). Le voyageur doit **obligatoirement** changer de train en chacune des villes de $\mathcal{V}$ (il a un truc à faire qui prends peu de temps mais il ne peut remonter dans le même train). **Attention :** la durée du parcours est donc calculée en utilisant la méthode de calcul du meilleur itinéraire (problème 1) pour chaque bout de trajet (i.e. les couples de villes successives composant le parcours). La méthode consiste à énumérer tous les parcours possibles récursivement. Cette énumération conduit à parcourir un arbre virtuel $\mathcal{A}$ dont les branches représentent tous les parcours possibles. Chaque noeud correspond à un parcours en cours de construction. Ses fils constituent tous les choix possibles parmi les villes restant à visiter. Chaque fois qu'un nouveau parcours est construit on le compare au meilleur parcours rencontré jusqu'à présent et s'il est meilleur on le mémorise.
Remarque 1. Il n'est évidemment pas utile de chercher à prolonger un parcours dont la durée courante excède la durée du meilleur parcours déja rencontré.
Remarque 2. Soit P un parcours en construction. Parmi les différents choix possibles pour la ville suivante certains sont à priori mieux adaptés que d'autres. Par exemple on peut supposer qu'une ville très proche de la dernière ville visitée a plus de chances de conduire à un bon parcours qu'une ville très éloignée. Il est donc judicieux d'orienter le choix de la ville suivante, i.e. d'envisager toutes les possibilités, mais de commencer par les villes les plus proches. Une règle de ce type s'appelle une *heuristique de choix*. Vous pourrez utiliser de telles heuristiques pour améliorer la recherche.

S'il vous reste du temps vous pourrez chercher d'autres méthodes pour ce problème (problème du voyageur de commerce), notamment les méthodes incomplètes de recherche locale (croisement 2-opt).