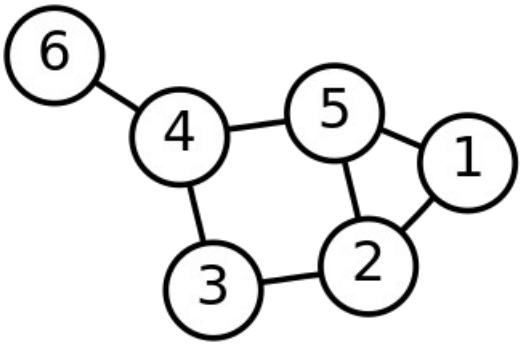
Projet de fin de module : Algorithme de pathfinding

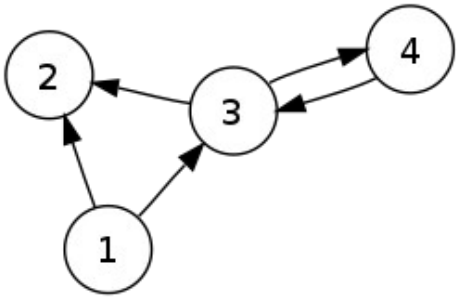
Introduction au langage C – Coda 1 ère année – Septembre 2024

Objectif

Le but du projet est d'implémenter un algorithme permettant de trouver le plus court chemin dans un graphe.

Un graphe est une structure de données composée de **nœuds** et de **liens** qui relient ces nœuds entre eux.





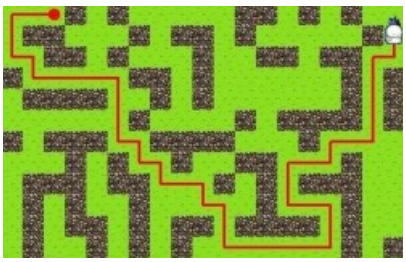
Les graphes sont utilisés dans beaucoup d'applications et permettent par exemple de modéliser des réseaux, des arbres de décisions, des labyrinthes etc.

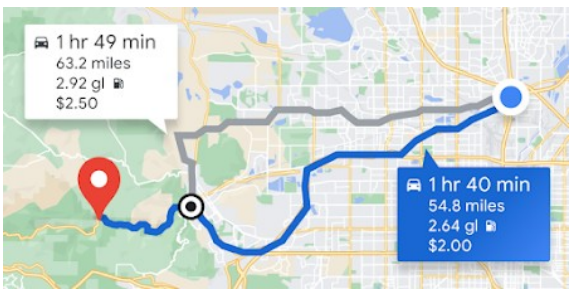
Les graphes peuvent être orientés ou non orientés, c'est à dire que les liens peuvent avoir une direction. Dans ce projet, on considérera que les graphes sont **non orientés** et que les liens sont à double sens.

Pathfinding

Le pathfinding est un problème algorithmique d'intelligence artificielle qui consiste à trouver **le plus court chemin dans un graphe depuis un nœud de départ un nœud d'arrivée.**

Il est utilisé dans de nombreux domaines comme la cartographie, les déplacements dans les jeux vidéos et autres applications nécessitant de trouver le chemin **optimal** dans un graphe





De nombreux algorithmes permettent pour résoudre ce problème, chacun avec ses avantages et ses inconvénients. Pour ce projet, il sera recommandé d'**utiliser des algorithmes simples de type BFS** (Breadth-First Search)

Nous vous conseillons d'implémenter l'**algorithme de Lee** pour le projet cependant **vous êtes libre du choix de l'algorithme tant que celui ci résout le problème**. Des algorithmes comme celui de Dijkstra ou A* sont également des possibilités.

Une des compétences attendues d'un développeur est de savoir choisir le bon algorithme en fonction de la situation et de savoir étudier les différentes solutions possibles.

Fonctionnement du projet et critères d'évaluation

Le programme doit dans un premier temps initialiser un graphe. Pour cela, vous devrez lire en C un fichier contenant la représentation d'un graphe sous forme textuelle. Le fichier est écrit dans un format particulier qui est détaillé dans la section suivante. Votre programme devra analyser le contenu du fichier et afficher un certain nombre d'informations dans le terminal.

Vous serez tout d'abord évalué sur votre capacité à écrire **un programme qui sait analyser et extraire des informations d'un fichier texte.**

Vous devrez, dans un deuxieme temps, initialiser un graphe en fonction des informations obtenues. Pour cela, vous devrez mettre en place les structures et les données vous permettant de représenter un graphe avec ses nœuds et ses liens. Il vous sera demandé d'implémenter un certain nombre de fonctions pour manipuler un graphe.

Vous serez évalué sur votre capacité à **écrire des fonctions indépendantes qui implémentent une fonctionnalité tout en respectant une signature imposée.**

Vous devrez ensuite implémenter un algorithme de pathfinding de votre choix. Votre programme devra trouver le plus court chemin entre le nœud de départ et d'arrivée du graphe en affichant dans la console la liste des nœuds à traverser. (l'affichage est détaillé dans les sections suivantes)

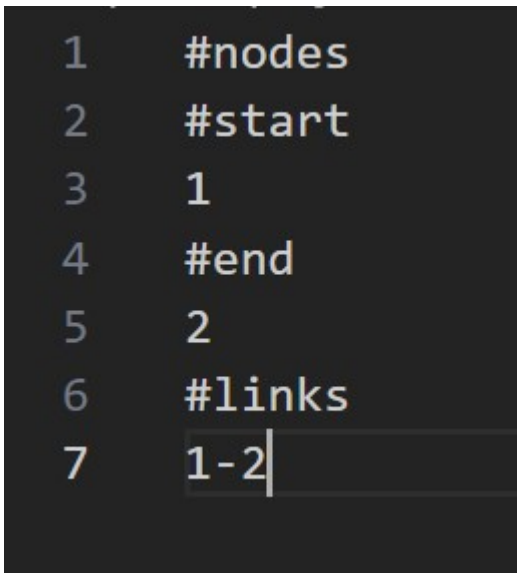
Vous serez évalué sur votre capacité à **comprendre et implémenter un algorithme pour résoudre un problème**. Un développeur doit savoir apprendre de nouvelles notions tout seul et être autonome. Pour ce projet, **vous devrez vous documenter sur le sujet et comprendre les problématiques et les solutions.**

Un développeur doit être capable de créer des codes robustes et résistant aux pannes. Une partie du projet vous demandera de savoir détecter des erreurs et retourner les codes d'erreurs correspondant. **Vous serez évalué sur votre capacité à gérer les erreurs et les cas particuliers**.

Enfin, la forme est aussi important que le fond et un développeur doit savoir écrire un code propre et lisible. Vous serez évalué sur votre **capacité à écrire un code de qualité en respectant les bonnes pratiques vues pendant le module.**

Format du fichier graphe

Le programme doit être capable de lire des fichiers textes contenant la représentation d'un graphe avec le format suivant



Le fichier doit d'abord contenir **une première section contenant la liste des nœuds du graphe.**
Les nœuds d'un graphe possèdent tous un numéro unique qui permet des les identifier. On appellera ce numéro **id** dans la suite du document.
Chaque ligne contient l'id d'un nœud. Le graphe contient 2 nœuds spéciaux : un nœud de départ et un nœud d'arrivée. Ces nœuds sont précédés respectivement des lignes #start et #end.

Un graphe sans nœud de départ ou de fin est incorrect et ne doit pas être traité par votre programme

Le fichier contient ensuite **une 2eme section indiquant les liens entre les nœuds**.
Chaque ligne de cette section indique un lien entre 2 nœuds sous la forme

```
[id noeud1]-[id noeud2]
```

On rappelle que les liens entre les nœuds sont **bidirectionnels**

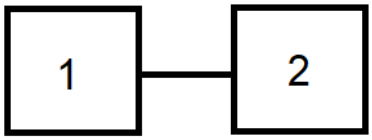
Un graphe ou il n'est pas possible de trouver un chemin permettant d'aller du nœud de départ jusqu'au nœud d'arrivée est incorrect et ne doit pas être traité par votre programme

Exemple 1 :

```
1  #nodes
2  #start
3  1
4  #end
5  2
6  #links
7  1-2
```

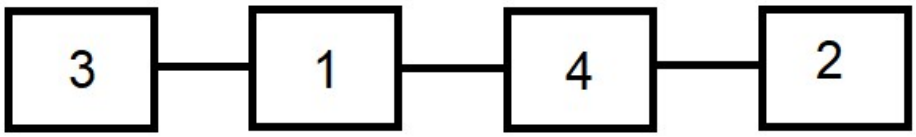
Liste des noeuds

Liens



Exemple 2

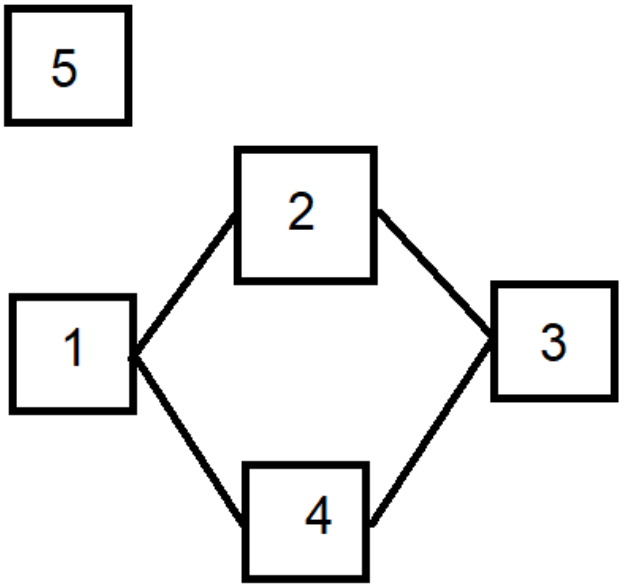
```
#nodes
1
#end
2
4
#start
3
#links
3-1
1-4
4-2
```



note : les id des nodes sont uniques mais ne se suivent pas forcement dans le fichiers
Pareil pour les tunnels

Exemple 3

```
#nodes
#end
3
2
4
#start
1
5
#links
1-2
2-3
1-4
4-3
```

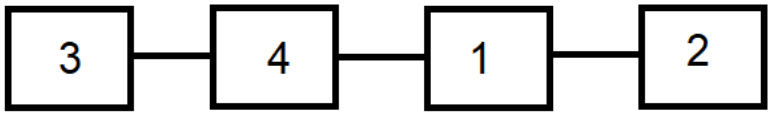


Affichage attendu

Votre programme doit afficher les choses suivantes

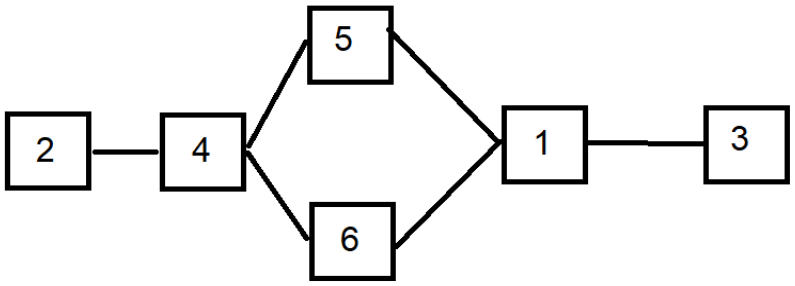
- Le nombre de nœuds sur une ligne
- Le nombre de liens sur une ligne
- Le nœud de départ du graphe sur une ligne
- Le nœud d'arrivée du graphe sur une ligne
- Un des chemins les plus courts
  - pathfinding sur une ligne
  - les id des nœuds séparés par des espaces

```
#nodes
1
#end
2
4
#start
3
#links
3-4
4-1
1-2
```




```
nombre de noeud(s): 4
nombre de lien(s): 3
start: 3
end: 2
pathfinding:
3 4 1 2
```

```
#nodes
1
#end
2
4
5
#start
3
6
#links
2-4
4-5
4-6
6-1
5-1
1-3
```



Dans le cas ou plusieurs chemins sont les plus courts, vous pouvez afficher l'un des 2 chemins

Le nœud start et end peuvent se trouver à n'importe quel endroit de la section de noeuds

ici 2 chemins sont possibles

- 2 4 5 1 3
- 2 4 6 1 3

Évaluation du projet

Le projet sera corrigé automatiquement par un programme. Vous devrez suivre très rigoureusement les consignes et nommer les fichiers de votre projet **exactement** comme indiqué dans ce document. Votre programme devra afficher un certain nombre de messages dans le terminal en fonction du graphe donné en entrée.

Le programme de correction va vérifier si votre projet affiche les bons messages en fonction de la situation, Le barème est détaillé dans ce document et des points seront accordés pour chaque fonctionnalité implémentée.

Faites bien attention à afficher exactement ce qui est indiqué. **Vous devrez faire attention à respecter à la lettre près le format indiqué dans les consignes**. **Si vos affichages diffèrent ne serait ce que légèrement par rapport à ce qui est attendu, vous obtiendrez un 0 par rapport aux critères du barème, même si votre programme fonctionne correctement**.

Vous serez également évalué sur la qualité de votre code et devrez mettre en place les bonnes pratiques vues lors de ce module. A savoir :

- Des noms de variables et de fonctions claires
- Utiliser la convention snake_case
- Des fonctions de moins de 20 lignes
- Une indentation correcte
- Un code correctement organisé

- Mise en forme du code uniforme dans le projet (accolades, saut de ligne, conventions de codage)
- Des commentaires si besoin

Des petits écarts peuvent être toléré, cependant si aucun effort n'est réalisé ou si certaines pratiques ne sont pas mises en place, **vous pourrez perdre jusqu'à 5 points sur votre note finale** même si votre projet fonctionne parfaitement.

Mise en place du projet:Makefile (0.5 point)

Créer un fichier Makefile à la racine de votre projet permettant de compiler votre projet. Votre makefile doit contenir les règles permettant de lancer les commandes : **make, make re, make clean**

La commande make doit compiler votre projet et créer un exécutable nommé **pathfinding**
Votre projet doit pouvoir s'exécuter avec les commandes suivantes :

make
./pathfinding nom_fichier

Fichier header (0.5 point)

Votre programme doit utiliser un fichier **header** pour la déclaration des fonctions et des structures. Votre projet doit contenir les fichiers **library.h** et **library.c** qui se doivent se trouver à la racine et doivent être compilé lors de l'utilisation de la commande make.

Structures (0.5 point)

Dans le fichier library.h, déclarez une structure pour représenter le nœud d'un graphe

```
typedef struct n{
    int id;
    struct n **links;
} Node;
```

Affichage du nombre de nœud (1 point)

Votre programme doit afficher le nombre de nœuds contenu dans le fichier

./pathfinding file.txt
...
nombre de noeud(s): 10
…

Affichage du nombre de liens (1 point)

Votre programme doit afficher le nombre de liens contenu dans le fichier

./pathfinding file.txt
...
nombre de liens(s): 10
…

Affichage du nœud de départ et de fin (1 point)

Votre programme doit afficher l'id du nœud de départ et d'arrivé depuis le fichier

./pathfinding file.txt
...
start: 1
end: 10
…

Initialisation des nœuds (3 points)

Implémenter une fonction node** init_node(char *filename) qui prend en paramètre le nom d'un fichier contenant un graphe et qui retourne un tableau de node contenant l'ensemble des nœuds du graphe. La fonction doit initialiser les liens entre les noeuds

Votre fonction doit se trouver dans le fichier library.c et library.h

Recherche de noeud (0.5 point)

Implémenter une fonction node* get_node_by_id(node **nodes, int id) qui prend en paramètre un tableau de nodes et l'id d'un nœud et qui retourne un pointeur vers le nœud recherché

Votre fonction doit se trouver dans le fichier library.c et library.h

Parcours du graphe (2 points)

Implémenter une fonction void display_nodes(node* start) qui prend en paramètre le node de départ d'un graphe et qui affiche la liste des nœuds du graphes. **Votre fonction ne doit pas afficher deux fois l'id d'un nœud et éviter les doublons**

Votre fonction doit se trouver dans le fichier library.c et library.h

L'affichage doit avoir la forme : 1 2 3 5 6

Détecter les nœuds isolés (2 points)

Implémenter une fonction node** get_unconnected_nodes(node **nodes, node *head) qui prend en paramètre un tableau de nœud et le nœud de départ du graphe puis qui retourne la liste des nœuds du graphe qui ne sont connectés a aucun des nœuds de la liste chaînée

Votre fonction doit se trouver dans le fichier library.c et library.h

Chemin valide (2 points)

Votre programme doit afficher un chemin valide entre le nœud de départ et le nœud d'arrivée
Vous obtiendrez 2 point si dans n'importe quel scénario votre programme affiche un chemin correct. Implémenter un algorithme de pathfinding valide vous donnera ces points gratuitement

Le programme de correction testera différents scénario et vous attribuera des points en fonction de la robustesse de votre programme : La correction va vérifier que votre programme affiche un chemin sous la forme '1 2 3 4'

Pathfinding simple (2 points)

Votre programme doit afficher le plus court chemin entre le nœud de départ et le nœud d'arrivée

Le programme de correction testera des scénarios simples et vous attribuera des points en fonction de la robustesse de votre code.

Pathfinding cas 2 (2 points)

Votre programme doit afficher le plus court chemin entre le nœud de départ et le nœud d'arrivée

Le programme de correction testera des graphes contenant des boucles et plusieurs chemins possible. Des points vous seront attribué en fonction de la robustesse de votre programme

Pathfinding cas complexes (2 points)

Votre programme doit afficher le plus court chemin entre le nœud de départ et le nœud d'arrivée

Le programme de correction testera des scénarios complexe et vous attribuera des points en fonction de la robustesse de votre programme

Gestion des erreurs (4 points)

Votre programme doit retourner des codes d'erreurs spécifiques en cas de problème.

Déclarer une enum contenant toutes les erreurs que votre programme doit gérer

```
typedef enum {
    FILE_NOT_FOUND = 1,
    NO_START_NODE = 2,
    NO_END_NODE = 3,
    NO_VALID_PATH = 4
    BAD_FILE_FORMAT = 5
} Error;
```

Votre programme doit retourner ces codes d'erreurs en fonction de la situation
FILE_NOT_FOUND : impossible d'ouvrir le fichier
NO_START_NODE : pas de nœud de début dans le fichier
NO_END_NODE : Pas de nœud de fin dans le fichier
NO_VALID_PATH : Le nœud de départ et de fin ne sont pas connecté
BAD_FILE_FORMAT : pas de section nodes ou links dans le fichier

Barème

- Makefile : 0.5 point
- Fichier header: 0.5 point
- Structure: 0.5 point
- Affichage du nœud de départ et d'arrivée : 1 point
- Affichage du nombre de noeud: 1 point
- Initialisation des nœuds du graphe : 3 points
- Recherche d'un noeud : 0.5 points
- Parcours du graphe: 2 point
- Détecter les nœuds isolés: 1 points
- Algorithme de pathfinding – Trouver un chemin valide: 2 points
- Algorithme de pathfinding - cas simple: 2 point
- Algorithme de pathfinding - cas 2: 2 points
- Algorithme de pathfinding - cas complexe: 2 points
- Gestion des erreurs : 4 points

Rendu

Vous devez rendre votre projet sous la forme d'un fichier compressé .zip contenant l'ensemble de vos fichiers par email à l'adresse suivante

dany.siriphol@gmail.com

Vous avez jusqu'au **vendredi 25 Octobre 23h59** pour rendre votre projet. 2 points seront retirés à votre note par jour de retard et tout projet rendu après le dimanche 27 Octobre 23h59 ne sera pas évalué.