Sujets Projet Note B (30%)
Algorithmique Avancée
2023-2024
Nicolas Loménie, Gael Mahé (élaboré avec Sylvain Lobry)

La note A (20%) correspond à votre compréhension théorique des algorithmes étudiés et en particulier applicables à des structures de graphes de données. C'est une préparation exemplaire à l'examen final (qui comptera pour 50% de la note finale) (Cette note correspond au Rendu Toussaint Section A des TPs/TDs + Rendu Noël Section A des TPs/TDs).

La note B correspond à votre capacité d'implémentation et de résolution de problèmes. Elle est pédagogiquement complémentaire de la compréhension théorique des algorithmes vus en cours. Elle est constituée de deux parties :

- Les rendus de la Toussaint et de Noël correspondant aux sections B des TPs/TDs (sur 10)
- Une deuxième note sur 20 expliquée ci-dessous. Vous aurez le choix entre 3 scenarii en fonction de vos projets d'étudiants ou motivations professionnelles (en surlignage).

Donc la note finale sera le maximum entre note d'Examen Final ou

Sections A des TD	+	*Section B des TP*	+	*Mini-Projet*	+	*Examen final*	=	
20 points	+	*10 points*	+	*20 points*	+	*50 points*	=	*100 pts*

Les documents nécessaires pour cette partie mini-projet se trouveront ici http://helios.mi.parisdescartes.fr/~lomn/Cours/AV/Projet/ .
Même mode opératoire que pour la première session.

Nous vous recommandons de vous en débarrasser pour le Samedi 23 Décembre 2023, 23:59
Mais la date de rendu fixé est le : 30 décembre 2023 23:59

Partie Commune

Vous rendez dans un format à part les exercices d'implémentations qui correspondent aux parties B des TD d'Algorithmique Avancée du semestre. Idéalement, vous ajoutez une analyse de la complexité de vos algorithmes éventuellement améliorés par vos choix d'implémentation ou de structures intermédiaires comme les tas etc. Vous obtiendrez une note sur 10 correspondant à ce rendu. Ce rendu est constitué d'un fichier *Nom_PartB.pdf* au format *pdf* qui explicite vos réponses et ce que vous avez réalisé et une arborescence de code (voir fin du document)

- Scénario 1 : Suite TP Labyrinthe (Dijkstra et statégie A*)

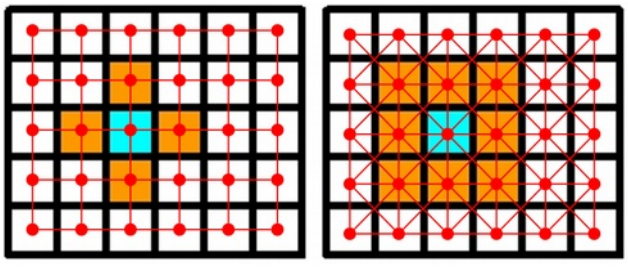
Vous faites évoluer le code fourni (choix du langage C, Python ou Java indifférent) qui résout le problème du labyrinthe. Vous modéliserez le problème comme la recherche d'un chemin de résolution dans un graphe d'états par une stratégie A*. Chaque mouvement/coût vous rapproche de la solution.
Une première heuristique que vous verrez en TP[(1)] sera la distance de la cellule courante à la sortie. Il faudra définir une meilleure heuristique et comparer le résultat. Dans un premier temps le Feu sera assimilé à un mur et donc statique (cas du TP). Puis pour les plus aventureux, vous pourrez réfléchir à une solution pour le cas où le feu avance comme dans l'énoncé.

Fichiers fournis à titre d'exemple: ***sujetLabyrinthe.pdf*** *avec une résolution naïve en C* ***labyrinthe.c.***

(1) Concernant A,vous aurez un TP sur ce sujet en Java la semaine du 14 Novembre et s'étendant sur deux semaines). Vous pouvez donc avancer sur l'infrastructure de votre programme et commencer à réfléchir d'ici là si vous le souhaitez à la problématique A* (https://en.wikipedia.org/wiki/A*_search_algorithm) mais en sortie du TP dédié à cet algorithme vous aurez une bonne base de travail pour votre rendu final.*

Ce rendu est constitué d'un fichier *Nom_MiniProjet.pdf* au format *pdf* qui explicite vos réponses et ce que vous avez réalisé et une arborescence de code. Toute plus-value personnelle notamment sur l'interfaçage graphique ou l'analyse des heuristiques ou de la complexité sera valorisée. Vous obtiendrez une note sur 20 correspondant à ce rendu.

- Scénario 2 : Graphes et Image (Dijkstra et modélisation)



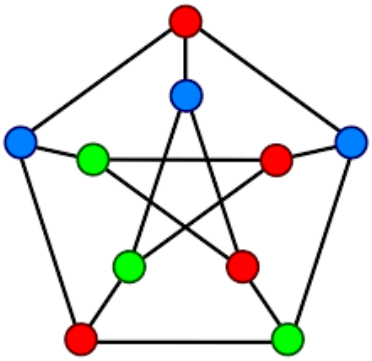
Vous travaillerez sur une structure de graphe particulière *: la grille image*. En effet, les pixels d'une image peuvent être considérés comme les sommets d'un graphe et la relation de voisinage la relation binaire d'arête. Ces arêtes peuvent être valuées par la différence d'intensité entre les deux pixels voisins. Vous construirez un tel graphe à partir de n'importe quelle image. Puis vous concevrez une interface graphique qui permette de désigner deux pixels un de départ et d'arrivée et de construire le plus court chemin dans ce graphe non orienté valué au sens de Dijkstra entre ces deux sommets ainsi qu'une visualisation de ce chemin sur l'image. Le choix du langage de programmation est libre entre Java,

Python ou C++. (https://perso.esiee.fr/~perretb/I5FM/TAI/connexity/index.html : la différence avec le labyrinthe est qu'un sommet est représenté par l'indiçage (i,j), ième ligne et jème colonne de la matrice image et les relations de voisinage ou arêtes sont immédiates en 4-coonexité les sommets voisins sont (i+1,j), (i-1,j), (i,j+1) et (i,j-1) ce qui confère une structure sans doute efficace d'accès aux informations de sommets et d'arêtes à partir de la matrice image). A titre d'exemple, des images sont proposées mais la taille devra peut-être encore réduite en fonction de votre implémentation (commencez par créer une image de taille 16x16 par exemple).

Ce rendu est constitué d'un fichier *Nom_MiniProjet.pdf* au format *pdf* qui explicite vos réponses et ce que vous avez réalisé et une arborescence de code. Toute plus-value personnelle notamment sur l'interfaçage graphique ou l'analyse des heuristiques ou de la complexité sera valorisée. Vous obtiendrez une note sur 20 correspondant à ce rendu.

- Scénario 3 : Sujet de concours grandes écoles (Coloriage de graphes)

Vous traitez l'épreuve d'informatique d'entrée aux ENS/Polytechnique. Vous avez le choix d'implémentation des exercices abordés en Python, Java, ou C/C++ (ou Caml). L'épreuve est censée durer 4h pour évaluer le travail en mode maîtrise du sujet. Il sera pris en compte de la difficulté éventuelle de certains exercices si vous justifiez pour un nombre limité d'entre eux leur non-résolution. En conséquence, il n'est pas nécessaire de traiter tous les points pour avoir la note maximale.



Voir le fichier **2018_mp_sujet_infoENSPolytechnique.pdf**

Ce scénario est en particulier intéressant pour ceux qui se destinerait à tenter des concours d'entrée en Ecole d'ingénieurs. Ou simplement pour se comparer à des étudiants qui suivent des classes préparatoires aux Grandes Ecoles.

Ce rendu est constitué d'un fichier *Nom_MiniProjet.pdf* au format *pdf* qui explicite vos réponses et ce que vous avez réalisé et une arborescence de code. Toute plus-value personnelle notamment sur l'interfaçage graphique ou l'analyse des heuristiques ou de la complexité sera valorisée. Vous obtiendrez une note sur 20 correspondant à ce rendu.

Tout plagiat sera sanctionné. Vous devez dans tous les cas utiliser vos propres formulations, reformulations et codes commentés en citant vos sources et le rendu en général doit avoir ce format d'arborescence (bien sûr avec votre propre nom).

```
.
├── CodeMiniProjet
├── CodePartB
│   ├── TP1
│   └── TPn
├── Lomenie_MiniProjet.pdf
└── Lomenie_PartB.pdf

4 directories, 2 files
```

Par ailleurs tous les codes Java doivent être exécutables sur une machine de l'UFR avec un .jar et la ligne de commande d'exécution que vous aurez testé. Les commentaires devront être en français et personnalisés. Une attention pour la notation sera portée sur l'interfaçage graphique (en particulier Java et Python) et les considérations d'optimisation éventuelle. Un README.txt expliquant l'usage et les modalités de compilation si besoin est nécessaire pour tout code.