Be Graphes

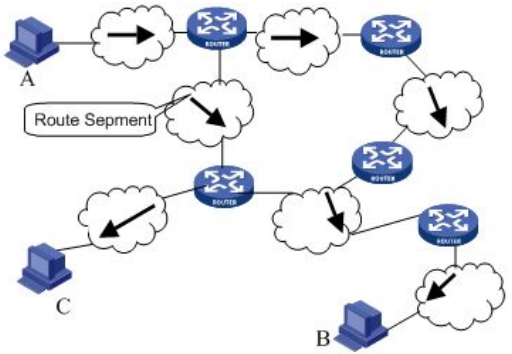
Algorithmes de plus court chemin

Introduction

GPS

Réseaux sociaux

Routage IP

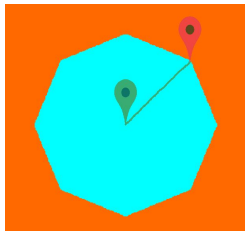


Contexte de développement

- Utiliser l'IDE Eclipse (pas une grande contrainte, on s'en est remis)
- Programmer en Java (POO)
- Comprendre et exploiter un code écrit par quelqu'un d'autre
- Ne pas modifier ou supprimer des classes, mais seulement en ajouter lorsque c'est nécessaire (principe ouvert/fermé)
- Utiliser JUnit pour les tests de validité

Tests de validité

I - Vérification visuelle:



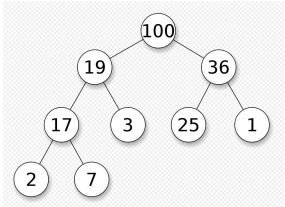
Djikstra rayonne autour de l'origine

A* sélectionne les sommets qui s'éloignent peu de la destination

II - Vérification des propriétés de l'algorithme:

	E	L	M	N	S	T
Départ	∞	∞	0_M	∞	∞	∞
M (0)	10_M	7_M		4_M	∞	∞
N (4)	10_M	6_N			12_N	∞
L (6)	10_M				11_L	∞
E (10)					11_L	14_E

Coût des sommets marqués croissants



Tas valide

Tests de validité

III- Avec oracle

Scénario \Algorithme	Bellman-Ford	Dijkstra	A*
carte non routière, chemin court, distance	5365	5365	5365
carte non routière, chemin court, temps	6:25	6:25	6:25
carte non routière, chemin long, distance	92801	92801	92801
carte non routière, chemin long, temps	1:51:21	1:51:21	1:51:21
carte routière, chemin nul, distance	Aucun chemin trouvé	Aucun chemin trouvé	Aucun chemin trouvé
carte routière, chemin court, distance	2796	2796	2796
carte routière, chemin court, temps	3:51	3:51	3:51
carte routière, chemin long, distance	113197	113197	113197
carte routière, chemin court, temps	1:13:02	1:13:02	1:13:02
carte routière, chemin inexistant, temps	Aucun chemin trouvé	Aucun chemin trouvé	Aucun chemin trouvé

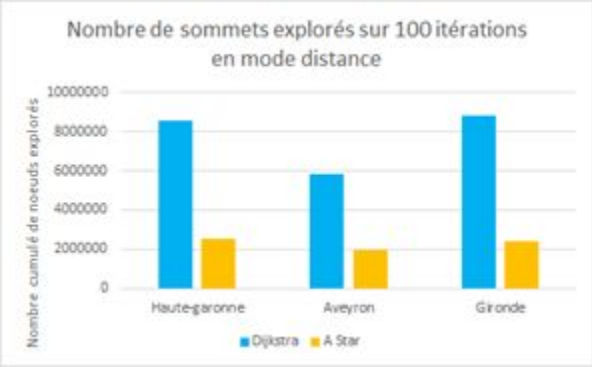
IV- Sans oracle

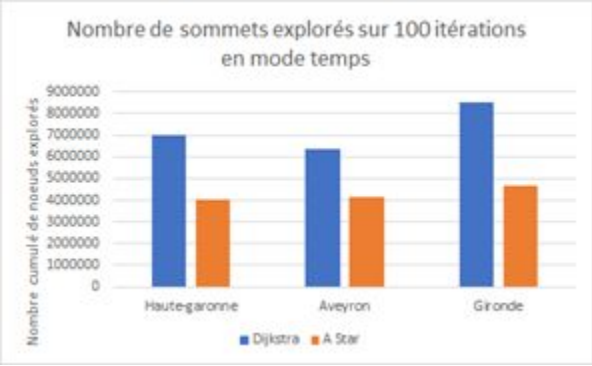
Tentative d'encadrement du plus court chemin

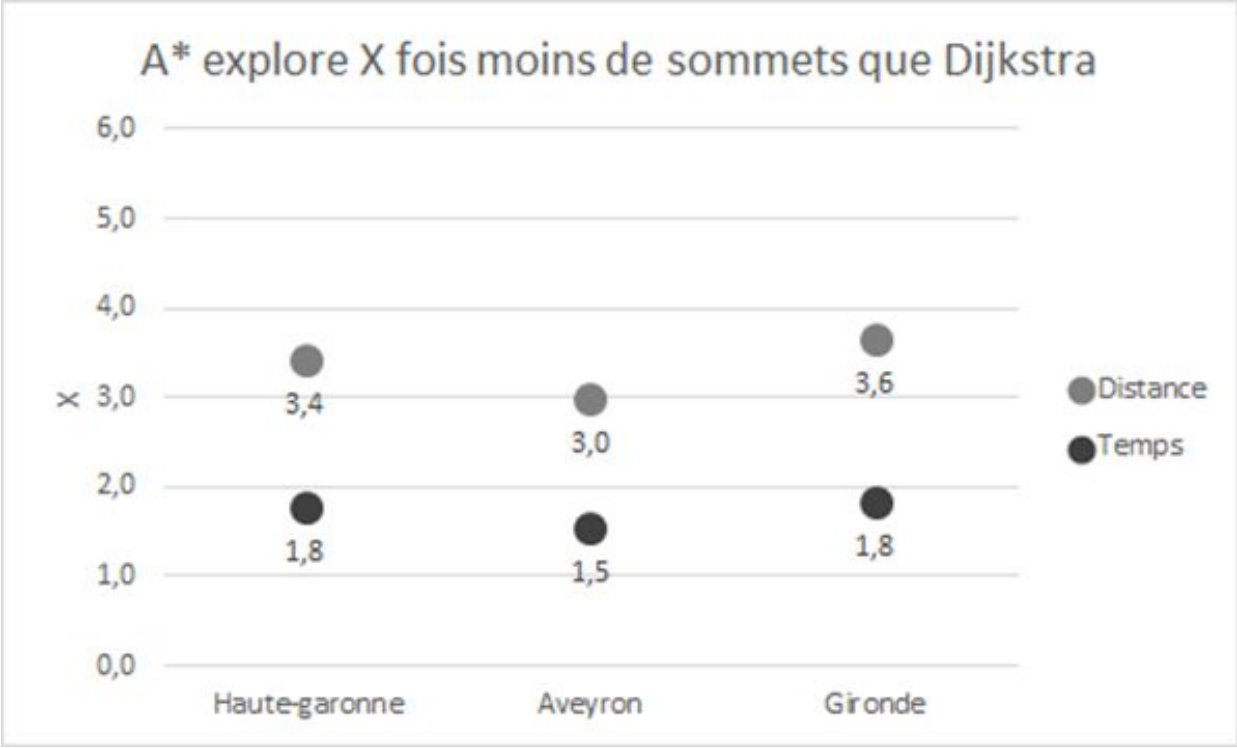
chemin à vol d'oiseau

déplacement en latitude et en longitude

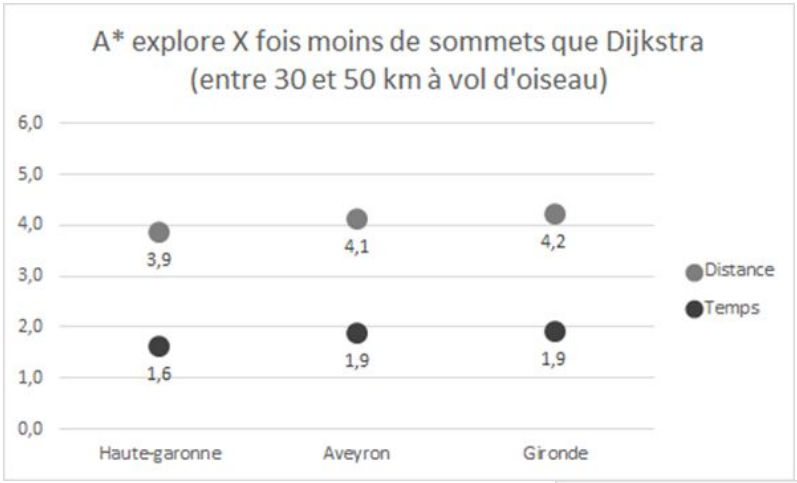
Tests de performance



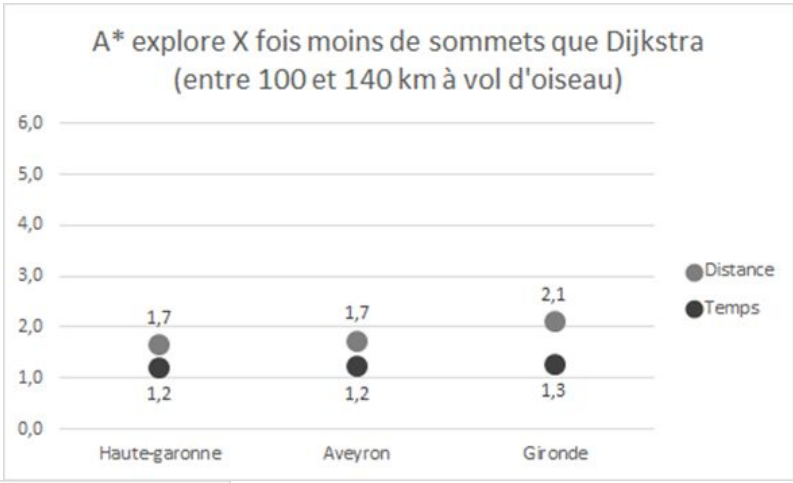




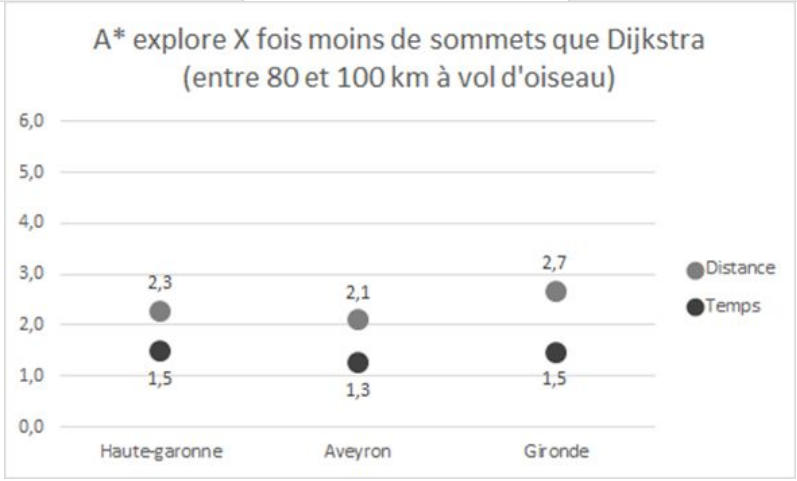
Tests de performance



Distances moyennes



Courtes distances



Longues distances

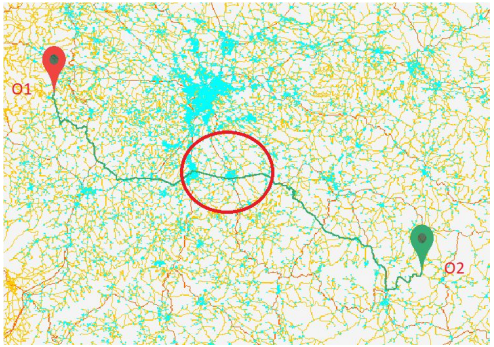
Problème ouvert : Point de rencontre

Objectif : trouver l'ensemble des point "globalement" au milieu de O1 et O2

1) Déterminer le coût O1 O2

2) Djikstra inverse pour connaître les coûts depuis O2

3) Sélection des sommets à coût égal depuis O1 et O2 et à mi-chemin



Conclusion

Nous avons appris à :

- Se confronter à un code déjà écrit qu'il faut s'approprier avant d'aller plus loin
- Réaliser des diagrammes UML dans cette optique
- ~~Utiliser Git~~ Faire des copier-coller dans Github
- Programmer en orienté objet

Un BE très satisfaisant : Interface graphique ludique, des algorithmes aux applications concrètes, palpables.