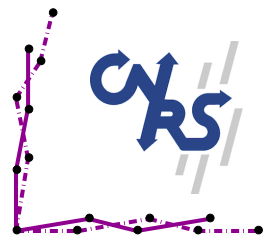


Conception et Pratique de l'Algorithmique

<http://www-apr.lip6.fr/~buixuan/cpa2019>

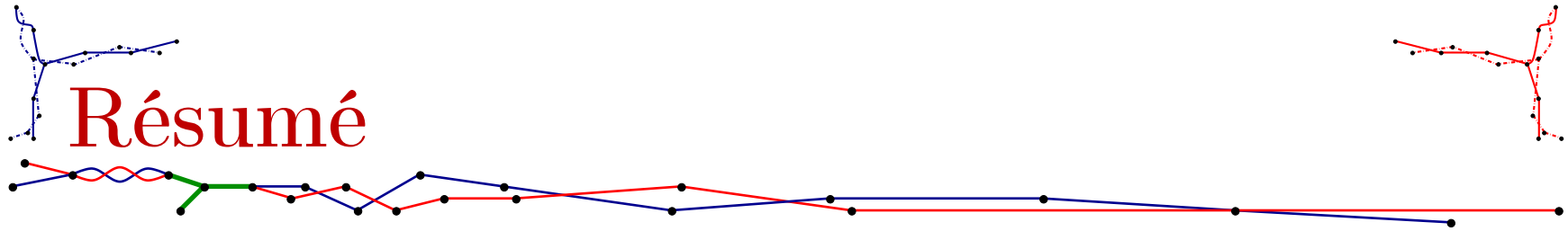
Binh-Minh Bui-Xuan



**SORBONNE
UNIVERSITÉ**
CRÉATEURS DE FUTURS
DEPUIS 1257

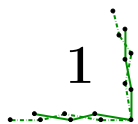
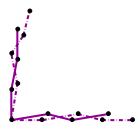
PARIS, Avril 2020

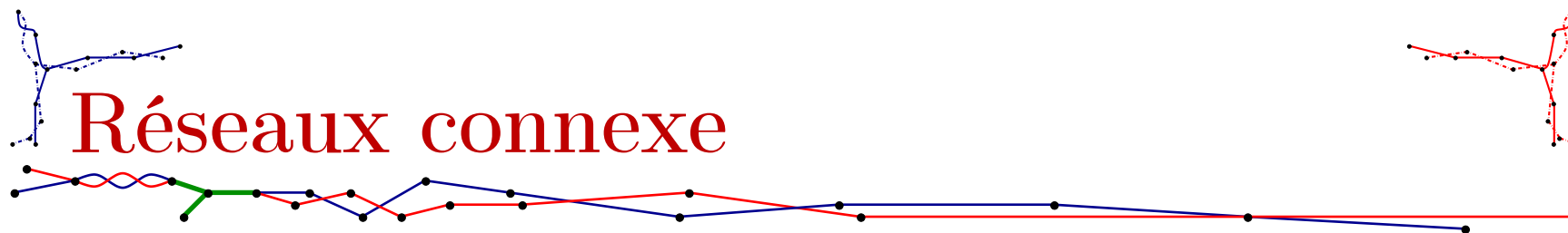




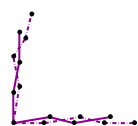
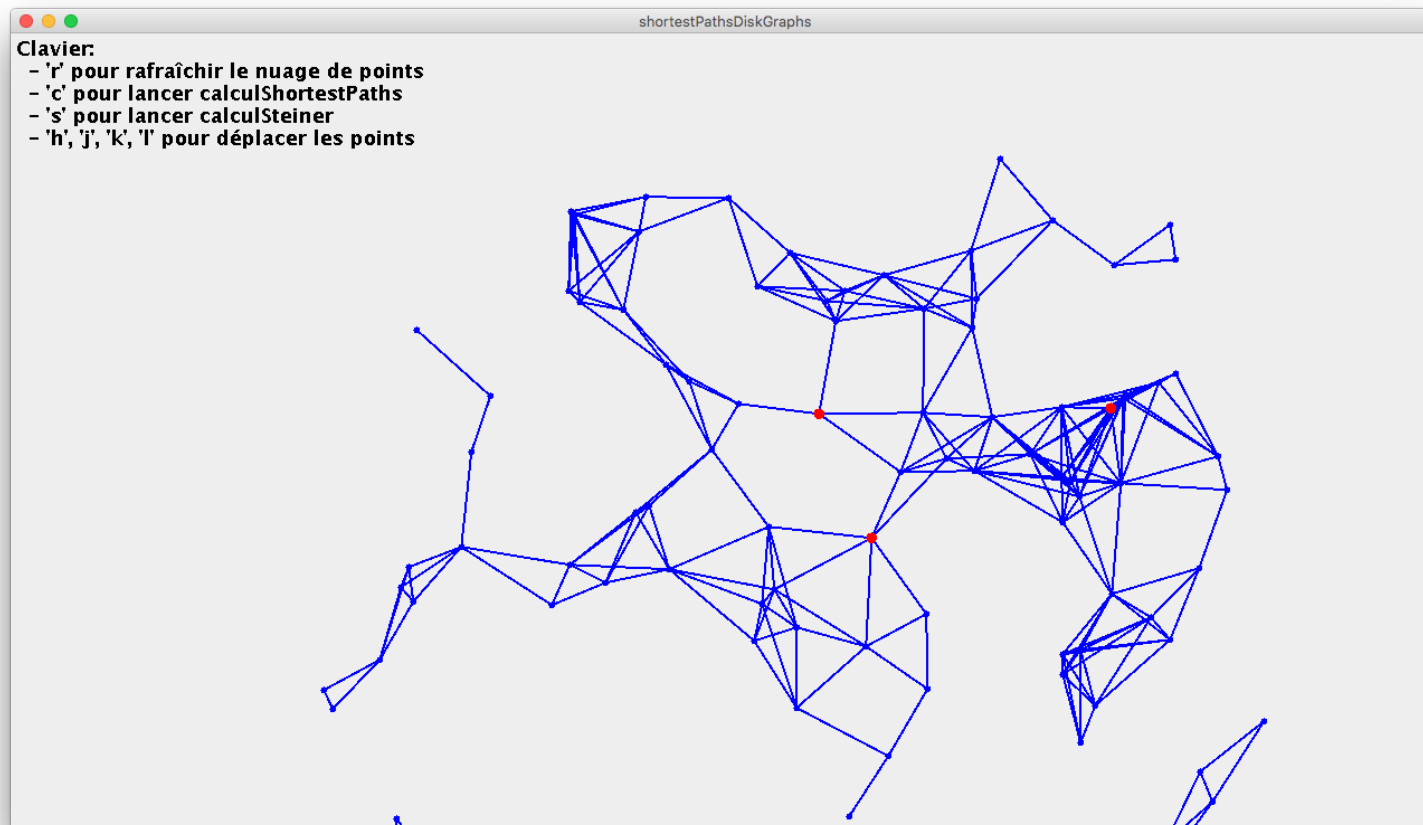
CONSTRUCTION D'ARBRE :

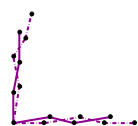
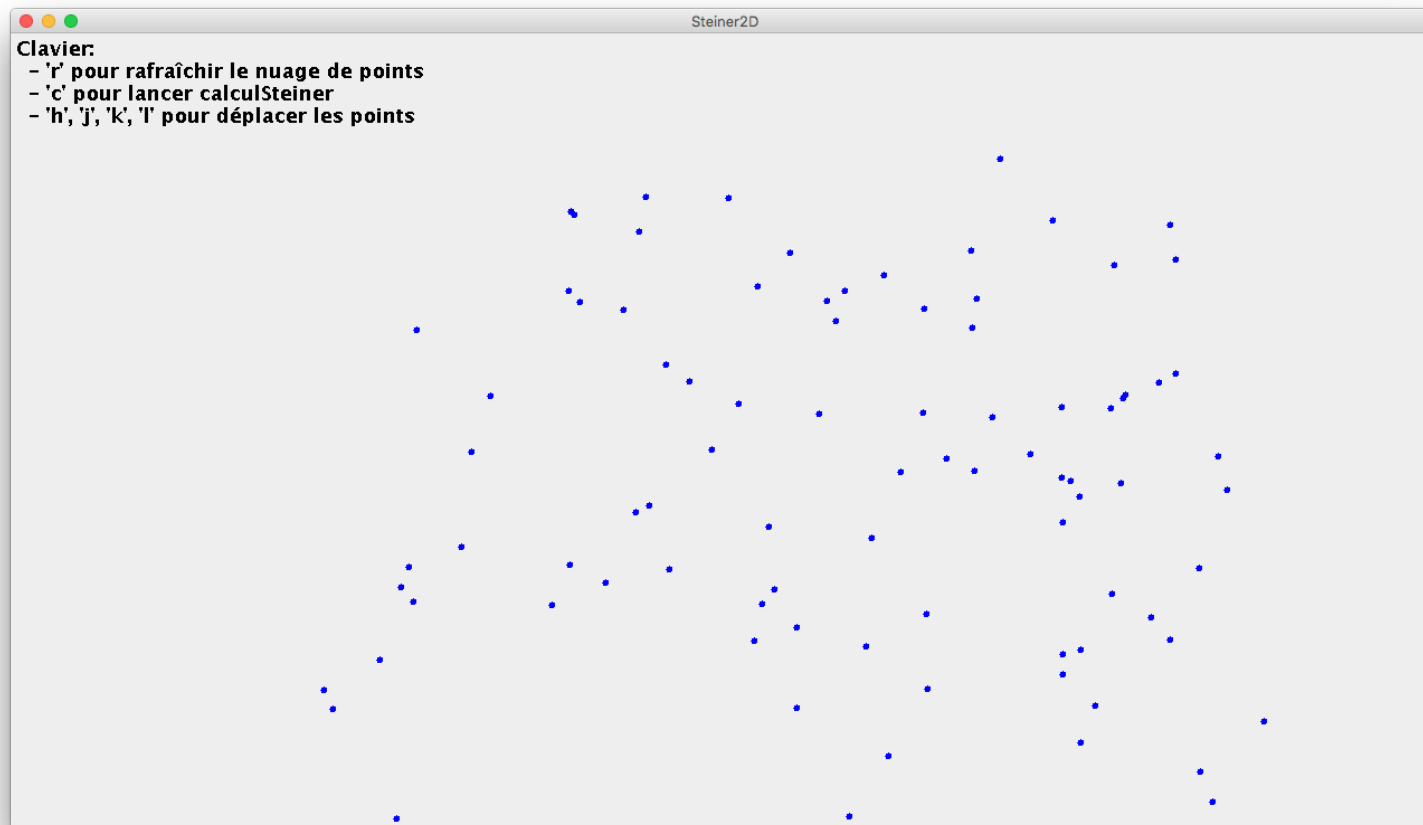
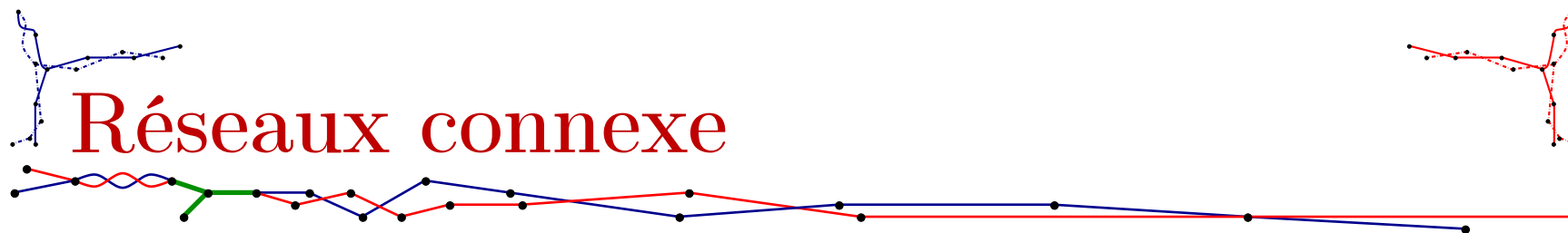
- arbre couvrant, arbre de Steiner, décomposition arborescente
- programmation dynamique, recherche locale
- concours de programmation

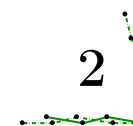
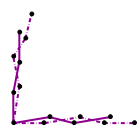
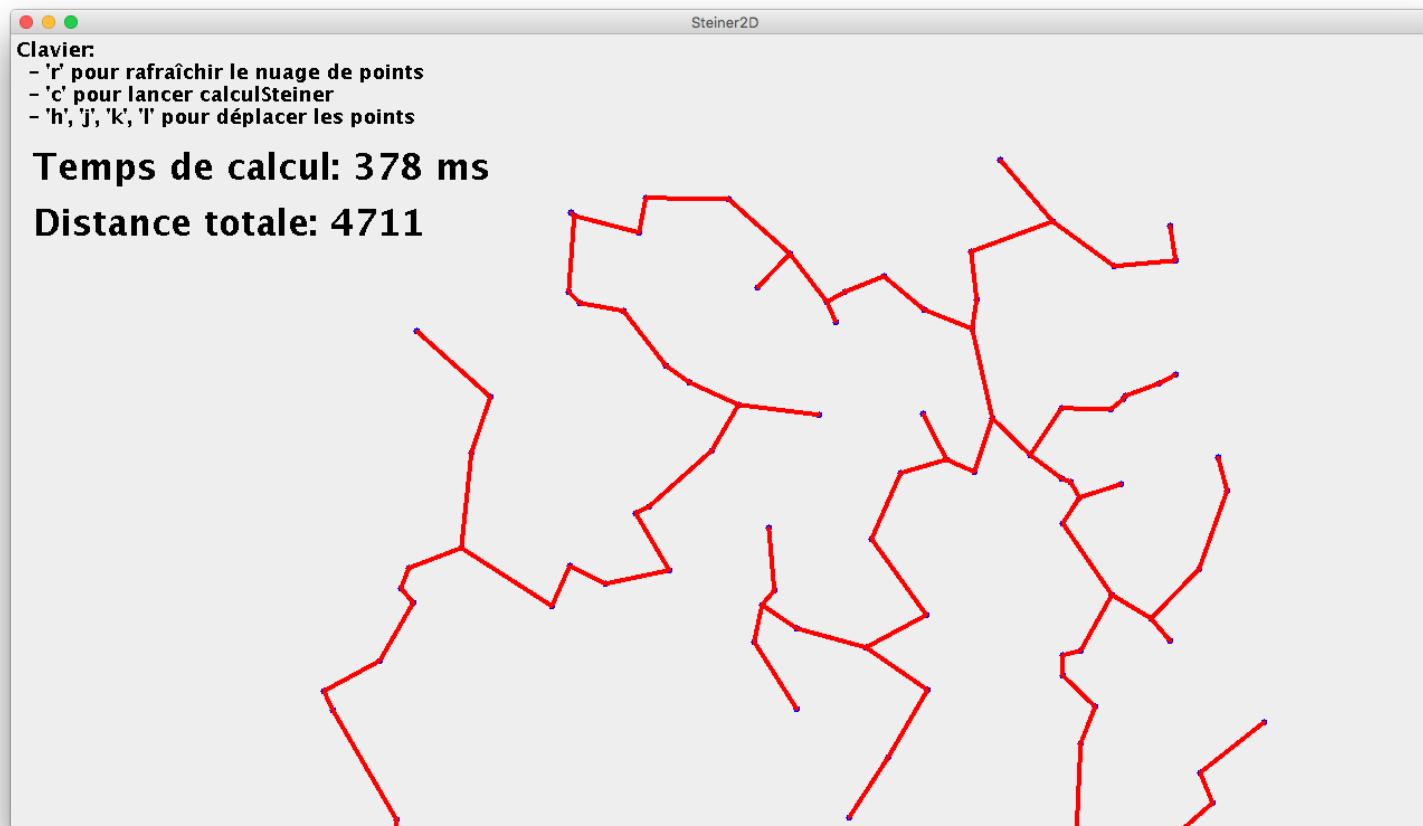
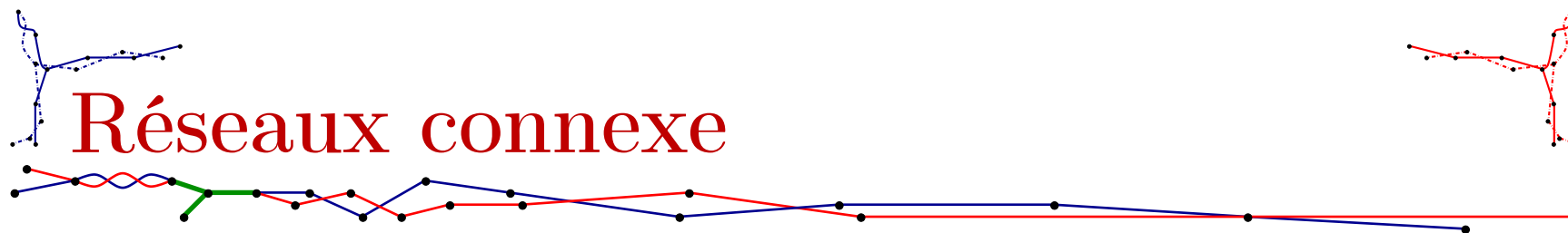


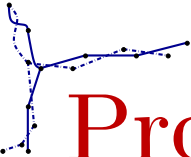


Réseaux connexe

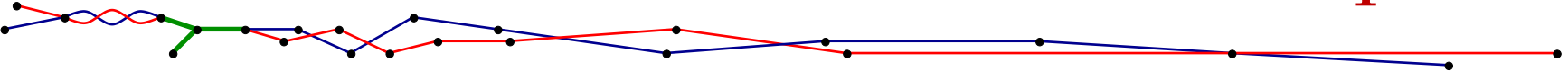







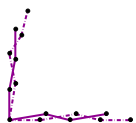


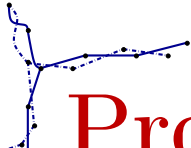
Problème d'arbre couvrant classique



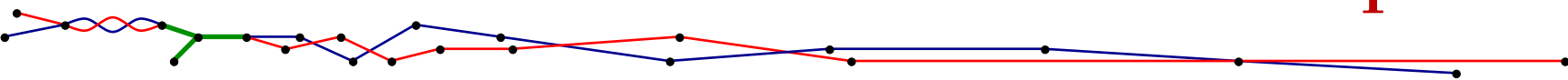
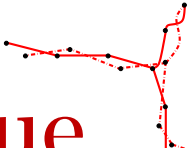
IN : Points, une liste de coordonnées de points en 2D

OUT : arbre MSTree couvrant tous les points de la liste, de poids total minimum (somme de la distance entre toutes les arêtes). De plus, tous les sommets de MSTree doivent appartenir à la liste Points.





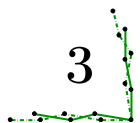
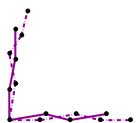
Problème d'arbre couvrant classique

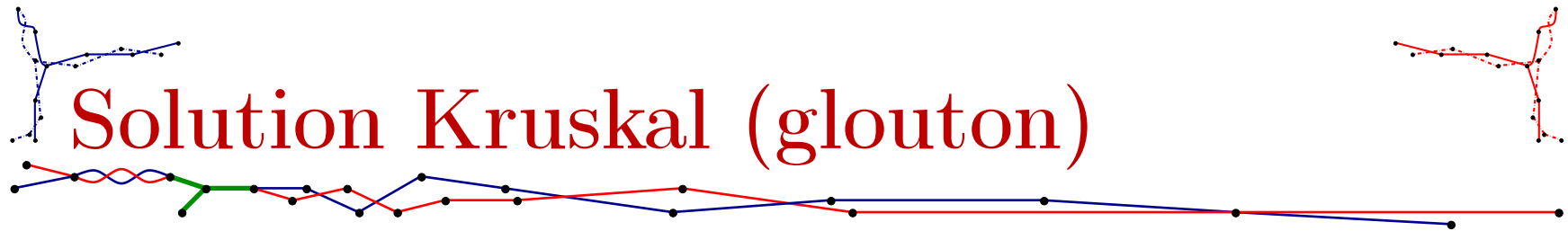


IN : Points, une liste de coordonnées de points en 2D

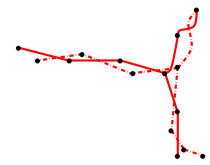
OUT : arbre MSTree couvrant tous les points de la liste, de poids total minimum (somme de la distance entre toutes les arêtes). De plus, tous les sommets de MSTree doivent appartenir à la liste Points.

EXERCICE : Structure de graphe ?

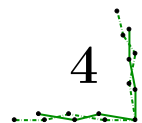
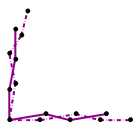


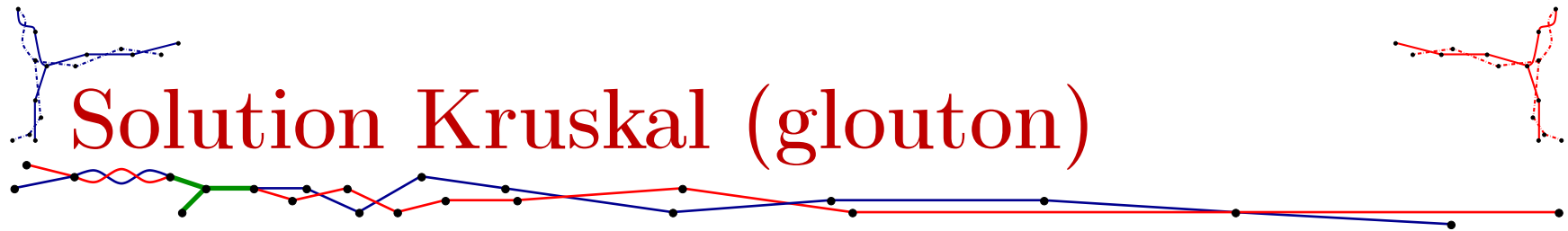


Solution Kruskal (glouton)



PRINCIPE : trier les arêtes par ordre croissant selon leur poids. Tant que l'ajout d'une arête de cette liste ne crée pas de cycle dans la solution (initialement vide), effectuer cet ajout dans la solution.

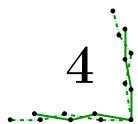
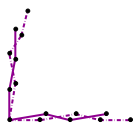


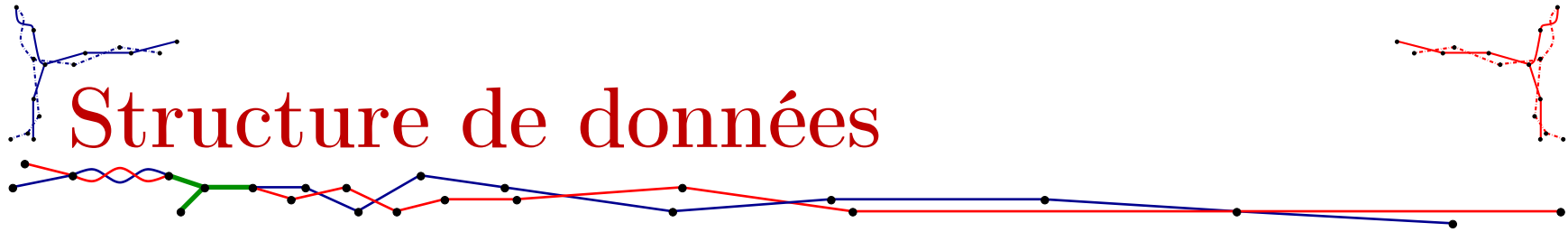


Solution Kruskal (glouton)

PRINCIPE : trier les arêtes par ordre croissant selon leur poids. Tant que l'ajout d'une arête de cette liste ne crée pas de cycle dans la solution (initialement vide), effectuer cet ajout dans la solution.

EXERCICE : Pseudo-code ?



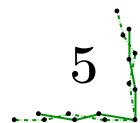
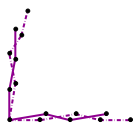


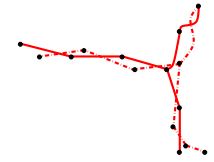
Graphe :

- matrice d'adjacence
- liste d'adjacence
- liste des arêtes

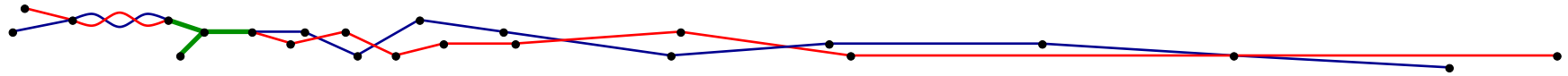
Arbre :

- matrice d'adjacence
- liste d'adjacence
- liste des arêtes
- hiérarchie (noeud, sousarbres)





Structure de données

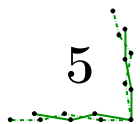
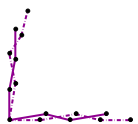


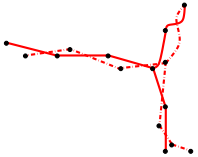

Graphe :

- matrice d'adjacence
- liste d'adjacence
- liste des arêtes

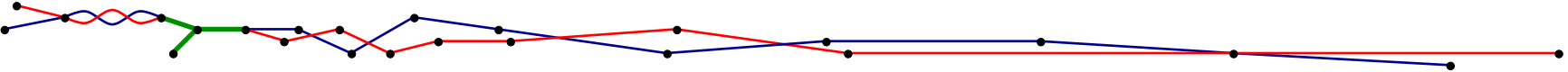
Arbre :

- matrice d'adjacence
- liste d'adjacence
- liste des arêtes
- hiérarchie (noeud, sousarbres)



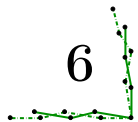
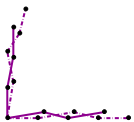


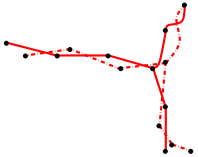

Structure de données : Kruskal



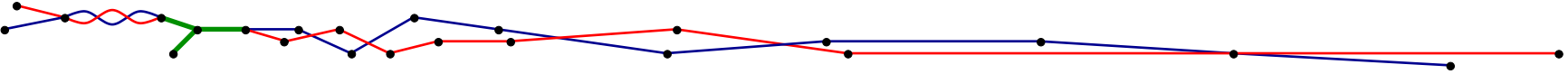
PRINCIPE : avec graphe G en input et arbre T en output :

$G \rightarrow$ liste d'arêtes de $G \rightarrow$ liste d'arêtes de $T \rightarrow$ hiérarchie





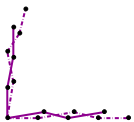
Structure de données : Kruskal

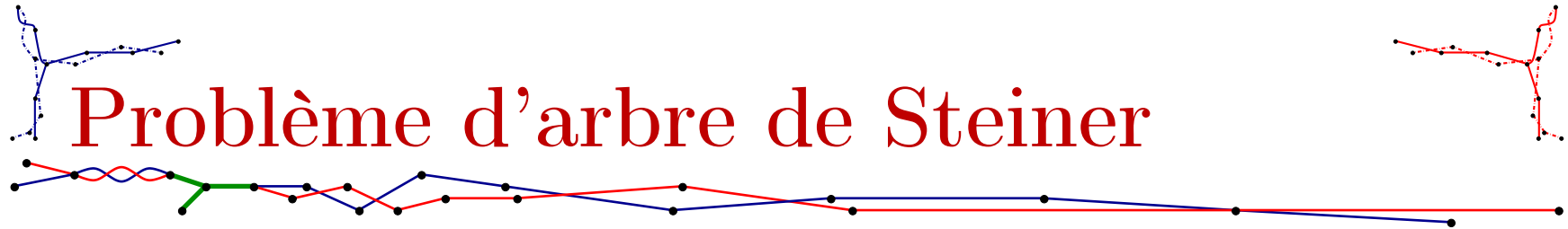


PRINCIPE : avec graphe G en input et arbre T en output :

$G \rightarrow$ liste d'arêtes de $G \rightarrow$ liste d'arêtes de $T \rightarrow$ hiérarchie

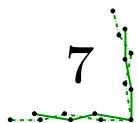
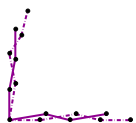
EXERCICE : liste d'arêtes de $T \rightarrow$ hiérarchie ?





IN : Points, une liste de coordonnées de points en 2D

OUT : arbre Tree couvrant tous les points de la liste, de poids total minimum (somme de la distance entre toutes les arêtes).



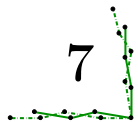
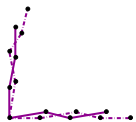


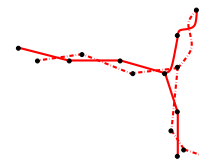
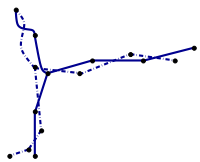
Problème d'arbre de Steiner

IN : Points, une liste de coordonnées de points en 2D

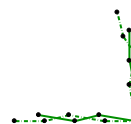
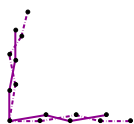
OUT : arbre Tree couvrant tous les points de la liste, de poids total minimum (somme de la distance entre toutes les arêtes).

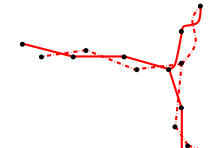
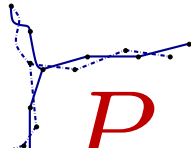
EXERCICE : différence avec arbre couvrant classique ?






Rappel express de la théorie de la complexité



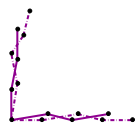


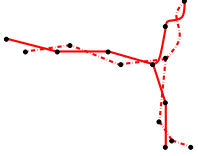
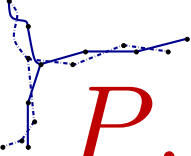
P , NP , et $co-NP$



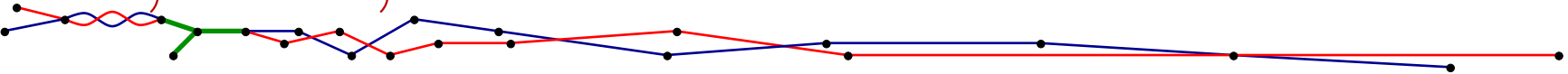
CLASSE P : l'ensemble de tous les problèmes de décision dont on peut calculer en temps polynomial en la taille de l'entrée (INPUT) et de la sortie (OUTPUT) la réponse (OUI/NON) au problème.

CLASSE NP : l'ensemble de tous les problèmes de décision dont on peut vérifier en temps polynomial en la taille de l'entrée (INPUT) et de la sortie (OUTPUT), moyennant un certain certificat, si la réponse est bien positive (OUI) au problème.



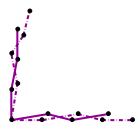


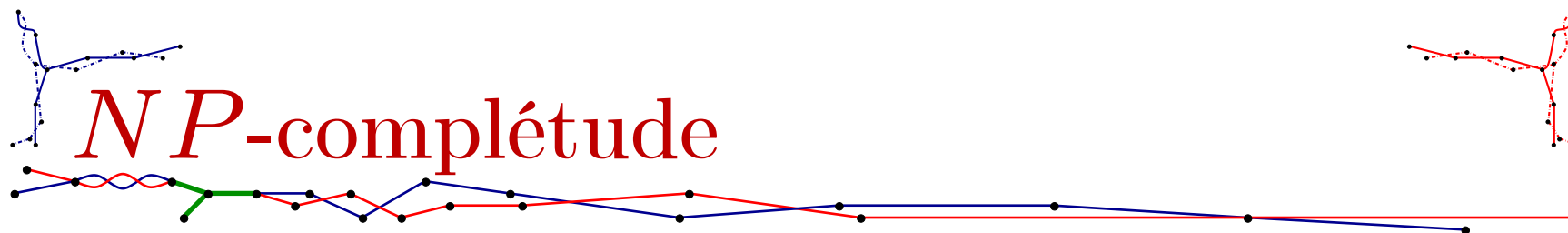
P , NP , et $co-NP$



CLASSE P : l'ensemble de tous les problèmes de décision dont on peut calculer en temps polynomial en la taille de l'entrée (INPUT) et de la sortie (OUTPUT) la réponse (OUI/NON) au problème.

CLASSE $co-NP$: l'ensemble de tous les problèmes de décision dont on peut vérifier en temps polynomial en la taille de l'entrée (INPUT) et de la sortie (OUTPUT), moyennant un certain certificat, si la réponse est bien négative (NON) au problème.

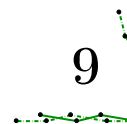
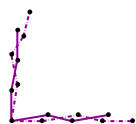


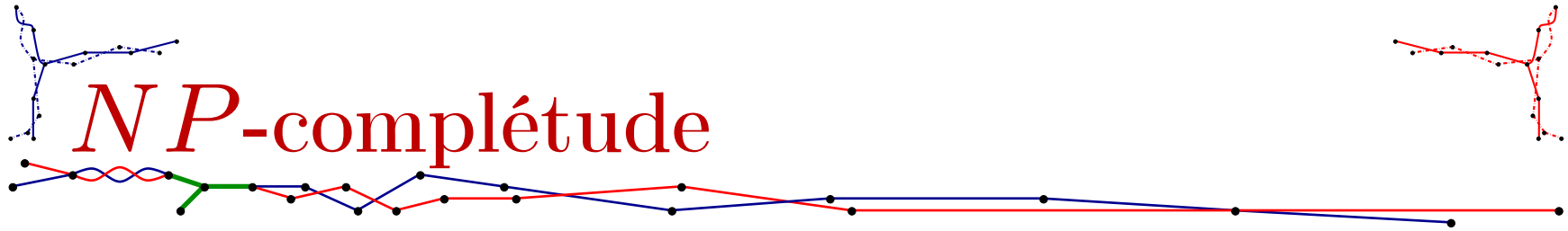


PROPRIÉTÉ TRIVIALE : $P \subseteq NP \cap co-NP$.

THÉORÈME : $P = NP \cap co-NP$.

QUESTION : qu'y a-t-il dans $NP \setminus P$?





PROPRIÉTÉ TRIVIALE : $P \subseteq NP \cap co-NP$.

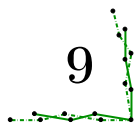
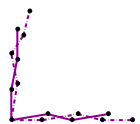
THÉORÈME : $P = NP \cap co-NP$.

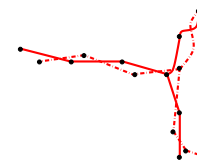
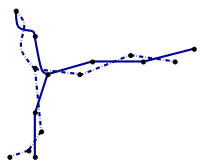
QUESTION : qu'y a-t-il dans $NP \setminus P$?

THÉORÈME (COOK) : *Si* $SAT \in P$ *alors* $NP \subseteq P$.

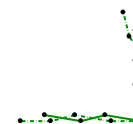
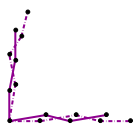
DÉFINITION : NP -complet = $\{Prob : \text{Si } Prob \in P \text{ alors } NP \subseteq P\}$.

WANTED (1M\$) : $P = NP$ ou $P \neq NP$?






Revenons à nos moutons (arbres)





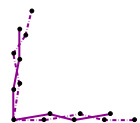
Problème d'arbre de Steiner – décision



IN : Points, une liste de coordonnées de points en 2D ; et B un réel.


OUT : existe-t-il un arbre Tree couvrant tous les points de la liste, de poids inférieur à B ?

EXERCICE : appartenance à NP ? à $co-NP$?





Problème d'arbre de Steiner – décision

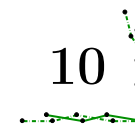
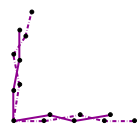


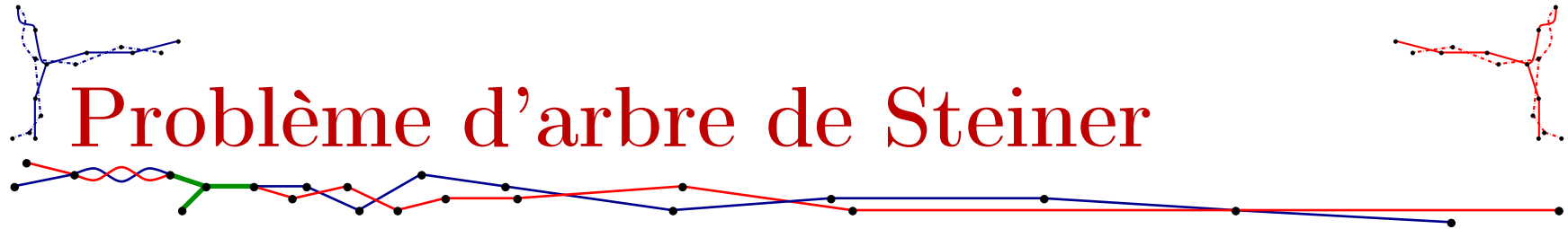
IN : Points, une liste de coordonnées de points en 2D ; et B un réel.

OUT : existe-t-il un arbre Tree couvrant tous les points de la liste, de poids inférieur à B ?

EXERCICE : appartenance à NP ? à $co-NP$?

THÉORÈME : *la version de décision du problème d'arbre de Steiner est NP -complet.*



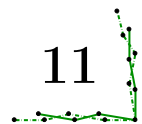
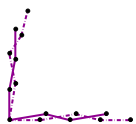


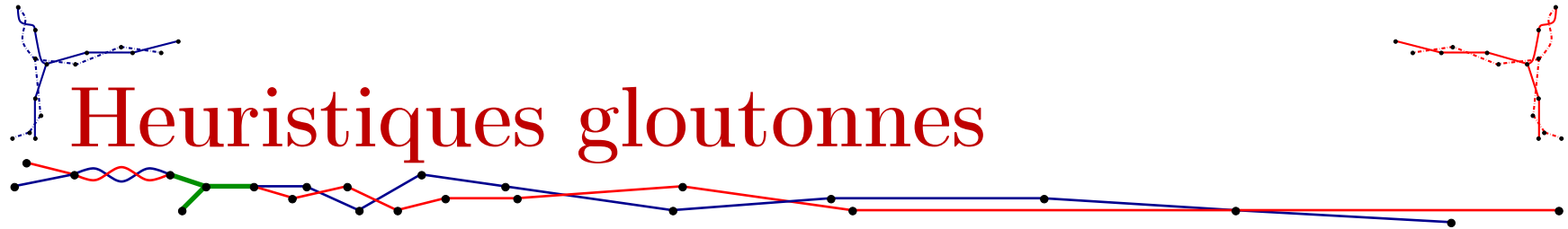
IN : Points, une liste de coordonnées de points en 2D

OUT : arbre Tree couvrant tous les points de la liste, de poids total minimum (somme de la distance entre toutes les arêtes).

EXERCICE : implémentation du test d'appartenance à NP ?

N.B. : problème de décision NP -complet \rightarrow problème d'optimisation associé est appelé NP -difficile

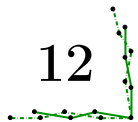
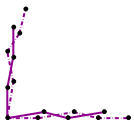


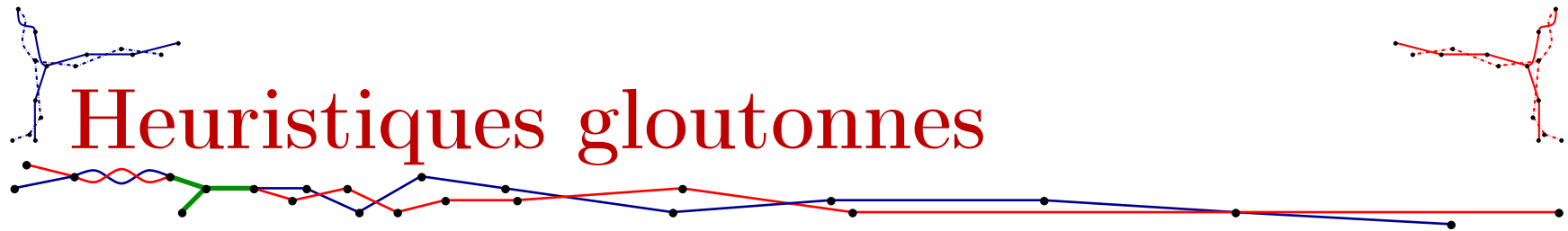


Heuristiques gloutonnes

PRINCIPE : ajouter des points à la liste `Points`, appeler `Kruskal` et comparer le résultat avec `CurrentTree`. Mettre à jour. Répéter...

NAÏF : essayer tous les coordonnées possibles.





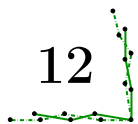
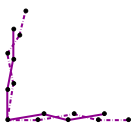
Heuristiques gloutonnes

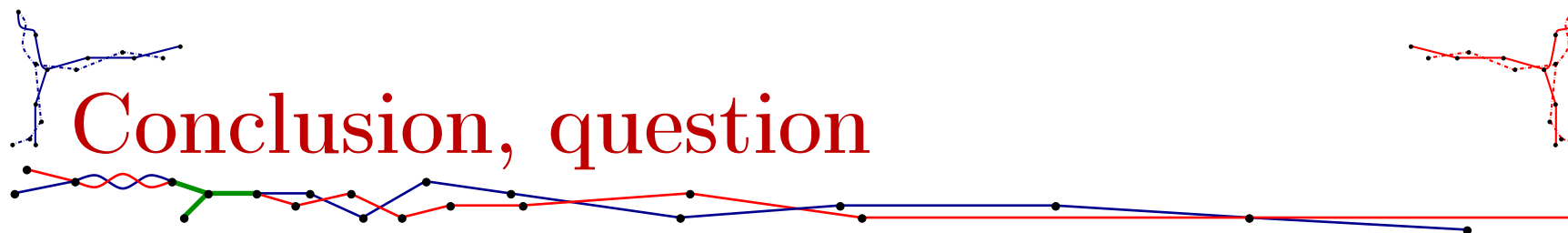
PRINCIPE : ajouter des points à la liste `Points`, appeler `Kruskal` et comparer le résultat avec `CurrentTree`. Mettre à jour. Répéter...

NAÏF : essayer tous les coordonnées possibles.

NAÏF AMÉLIORÉ : pour tout triangle qui ne contient aucun autre point, essayer tous les coordonnées possibles.

EXERCICE : mieux ?





Conclusion, question

CONCLUSION :

- ARBRECOUVRANTMIN algorithme Kruskal en $O(m \log n)$
- ARBRESTEINER NP-difficile
 - heuristique : recherche locale
 - optimisation locale : à l'aveugle, barycentre, Torricelli-Fermat

QUESTION :

- implantation ? (voir TME)

