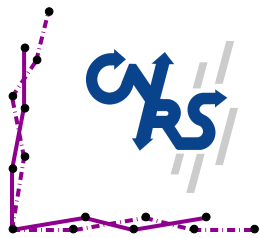


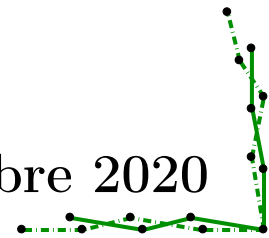
# Algorithmique Avancée

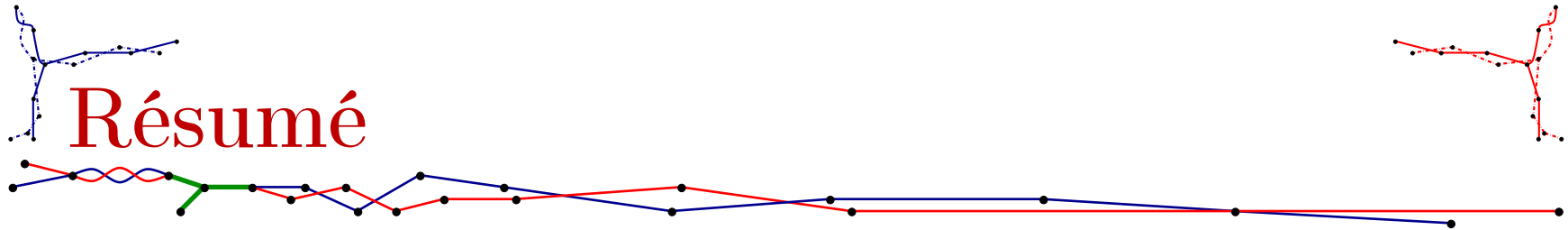
<http://www-apr.lip6.fr/~buixuan/algav2020>

Binh-Minh Bui-Xuan



PARIS, Octobre 2020



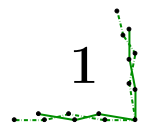
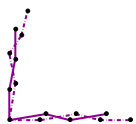


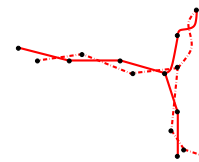
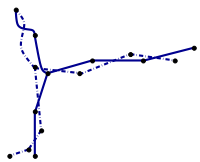
## GÉOMÉTRIE ALGORITHMIQUE (3 SÉANCES) :

- collision d'objets, conteneur (cercle, rectangle, polygone)

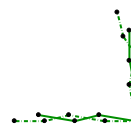
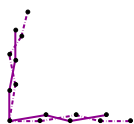
### SUITE :

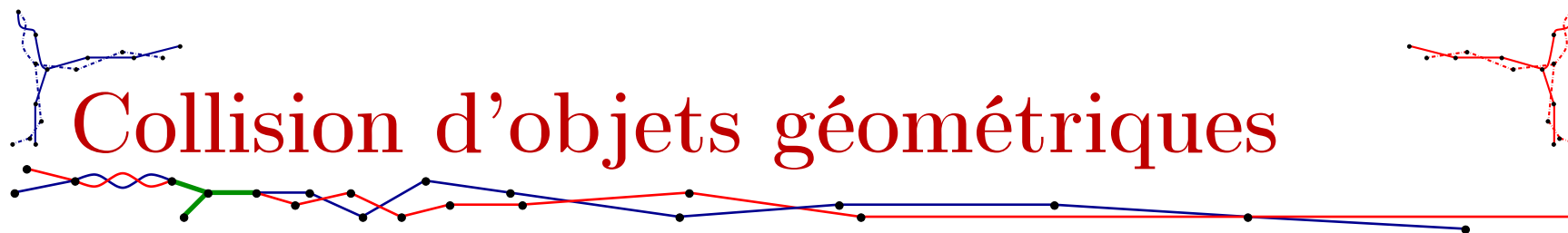
- CPA : Conception et Pratique de l'Algorithmique
- DAAR : Développement des Algorithmes d'Appli. Réticulaire
- AAGA : Analyse des Algorithmes et Génération Aléatoire
- Concours en ligne #HashCode, Code Jam, TopCoder, ...



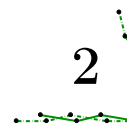
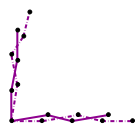
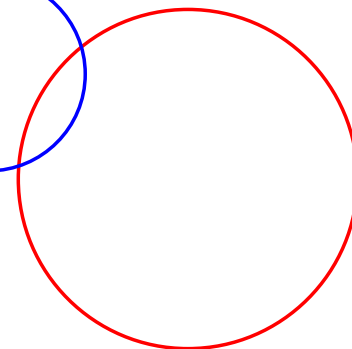
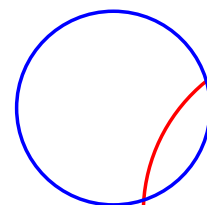
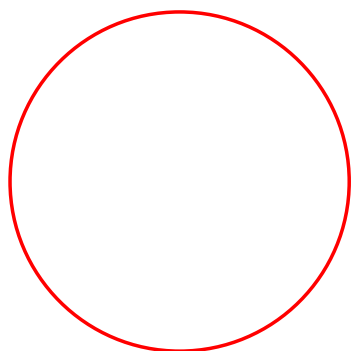
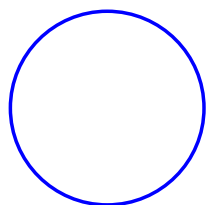


# Géométrie algorithmique





QUESTION : touché ?



# Collision d'objets géométriques

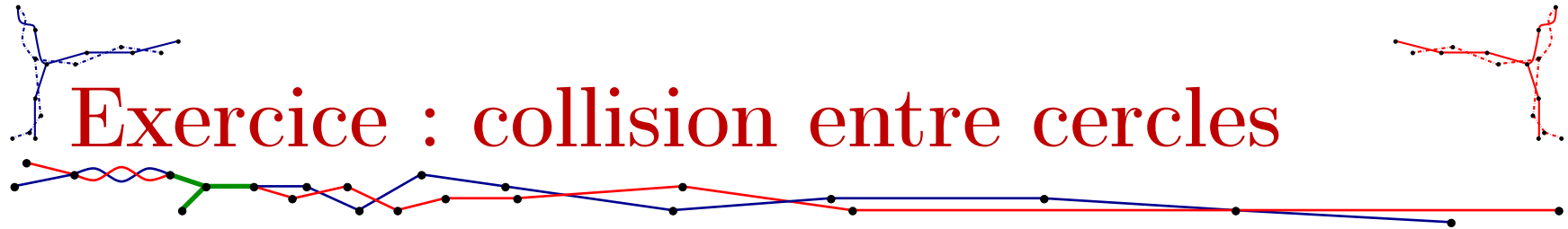
QUESTION : touché ?



# Collision d'objets géométriques

QUESTION : touché ?



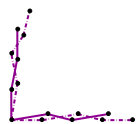


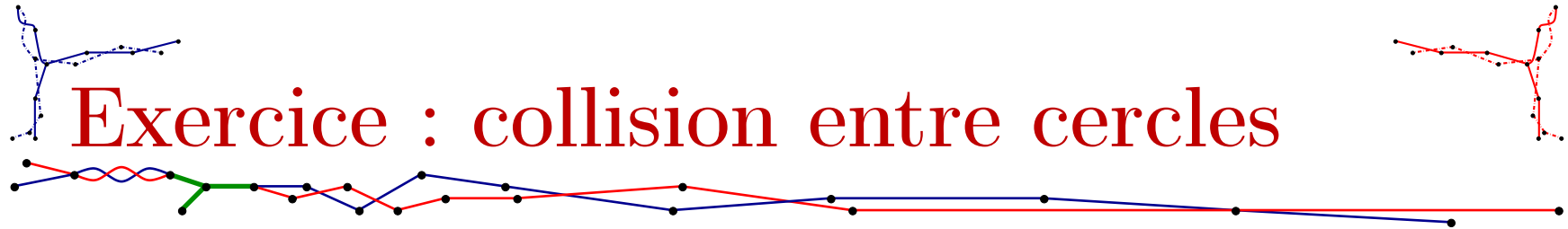
## Exercice : collision entre cercles

EXERCICE : Soient deux cercles  $c1$  et  $c2$  de rayons  $c1.radius$  et  $c2.radius$ , dont les coordonnées des centres sont  $(c1.x, c1.y)$  et  $(c2.x, c2.y)$ . Déterminer une condition nécessaire et suffisante pour que les deux cercles s'intersectent.

SUPPORT :

[http://www-apr.lip6.fr/~buiquan/files/RBB\\_collision\\_canevas.html](http://www-apr.lip6.fr/~buiquan/files/RBB_collision_canevas.html)



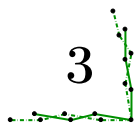
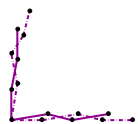


EXERCICE : Soient deux cercles  $c1$  et  $c2$  de rayons  $c1.radius$  et  $c2.radius$ , dont les coordonnées des centres sont  $(c1.x, c1.y)$  et  $(c2.x, c2.y)$ . Déterminer une condition nécessaire et suffisante pour que les deux cercles s'intersectent.

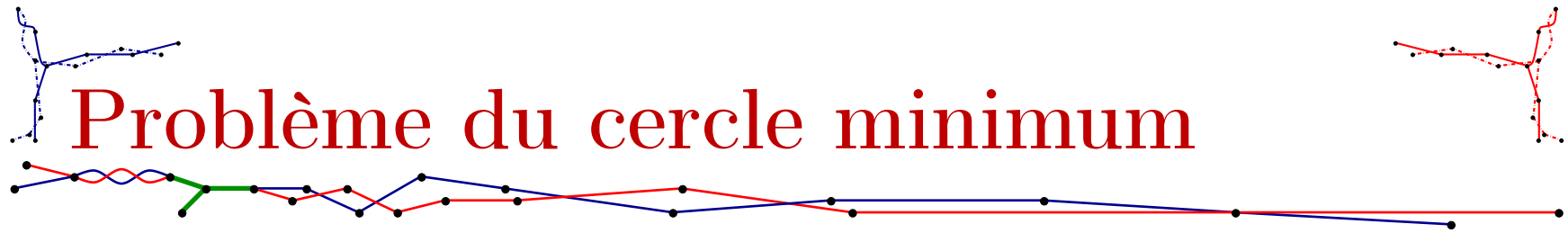
SUPPORT :

[http://www-apr.lip6.fr/~buixuan/files/RBB\\_collision\\_canevas.html](http://www-apr.lip6.fr/~buixuan/files/RBB_collision_canevas.html)

QUESTION : erreurs d'arrondi ?

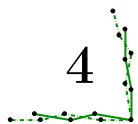
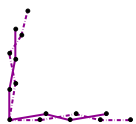


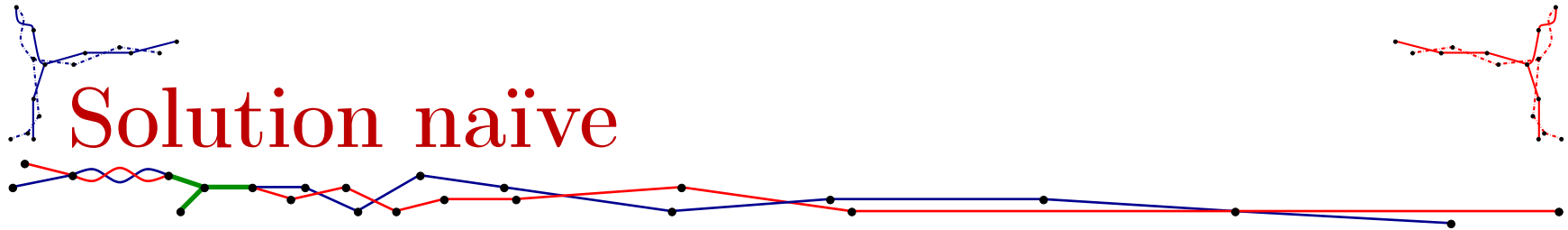




IN : Points, une liste de coordonnées de points en 2D

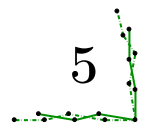
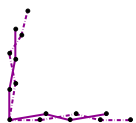
OUT : cercle couvrant tous les points de la liste, de rayon minimum

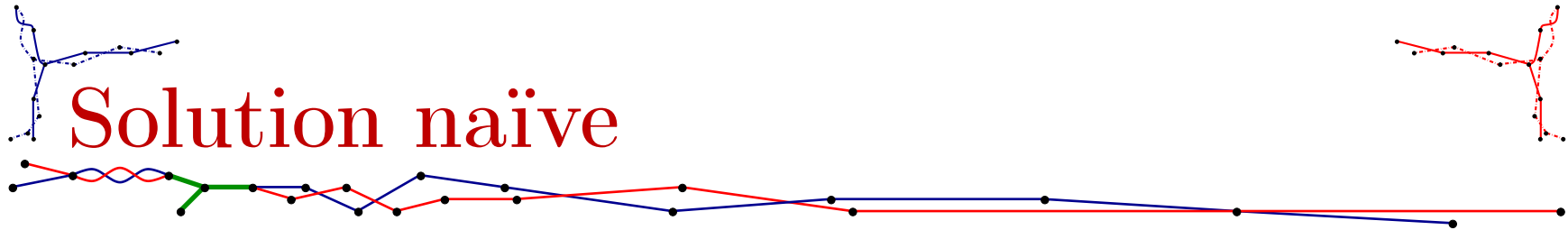




LEMME 1 : *si un cercle de diamètre égale à la distance de deux points de la liste couvre tout autre point de la liste, alors ce cercle est un cercle couvrant de rayon minimum.*

LEMME 2 : *en 2D, il existe un et un seul cercle passant par 3 points non-colinéaires.*





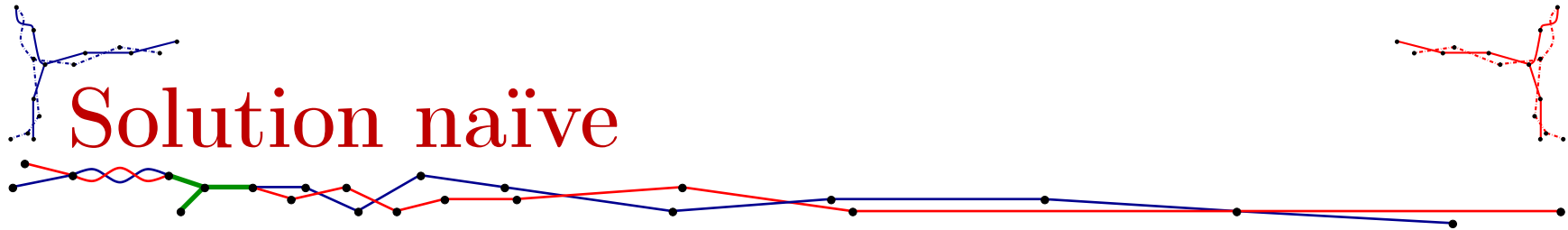
LEMME 1 : *si un cercle de diamètre égale à la distance de deux points de la liste couvre tout autre point de la liste, alors ce cercle est un cercle couvrant de rayon minimum.*

LEMME 2 : *en 2D, il existe un et un seul cercle passant par 3 points non-colinéaires.*

THÉORÈME : *le problème du cercle minimum peut être résolu en temps  $O(n^4)$ .*

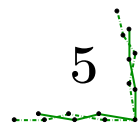
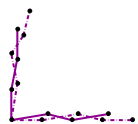
QUESTION : *algorithme prouvant ce théorème ?*

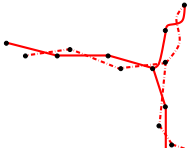
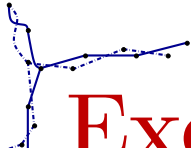




L'algorithme naïf en question :

1. pour tout  $p$  dans Points
2.   pour tout  $q$  dans Points
3.      $c \leftarrow$  cercle de centre  $\frac{p+q}{2}$  de diamètre  $|pq|$
4.     si  $c$  couvre tous les points de Points alors retourner  $c$
5. pour tout  $p$  dans Points
6.   pour tout  $q$  dans Points
7.     pour tout  $r$  dans Points
8.      $c \leftarrow$  cercle circonscrit de  $p$ ,  $q$  et  $r$
9.     si  $c$  couvre tous les points de Points alors retourner  $c$



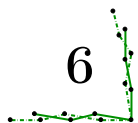
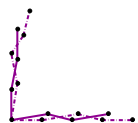


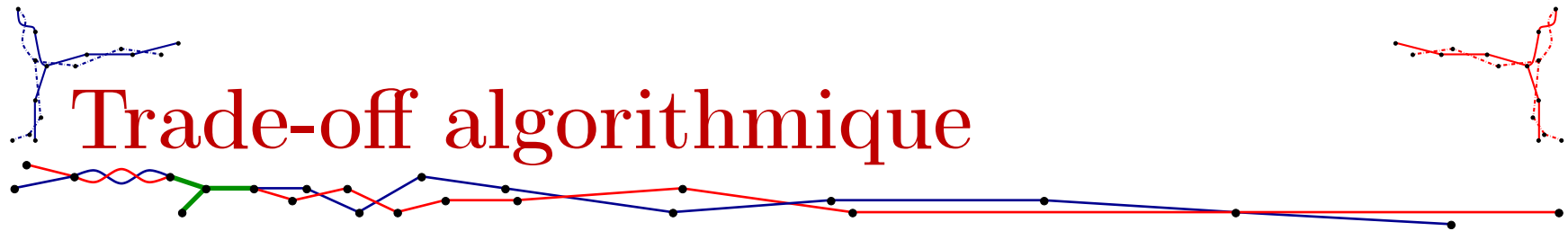
# Exercice : estimation du temps



QUESTION : un ordinateur de l'ordre du Giga-Hertz exécutant un algorithme en  $O(n^4)$ , avec  $n = 10000$ , quel est le temps d'exécution attendu (en ordre de grandeur) ?

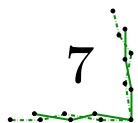
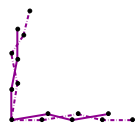
- algorithme en  $O(n^3)$  ?
- algorithme en  $O(n^2)$  ?
- algorithme en  $O(n)$  ?
- algorithme en  $O(\log n)$  ?

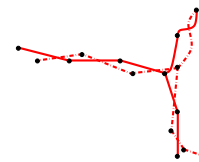
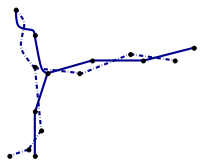




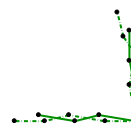
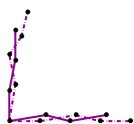
Qualité du résultat vs. temps d'exécution :

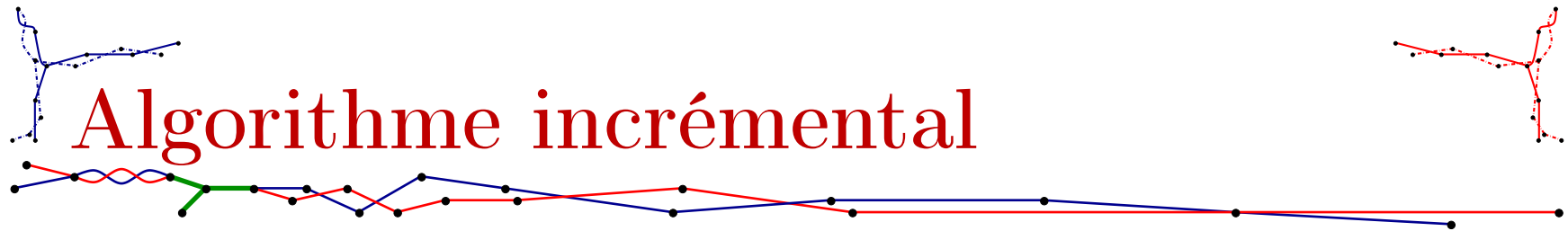
QUALITÉ GAGNE	TEMPS GAGNE	TRADE-OFF
imagerie médicale	audio-visuel	concours de prog.
systèmes critiques	appli. en temps réel	critère d'optimisation





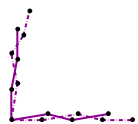
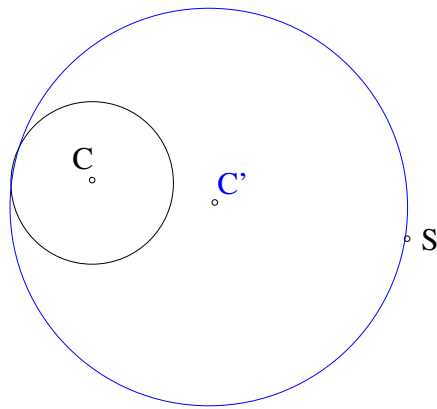
# Techniques d'approximation






# Algorithme incrémental


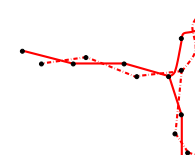
IDÉE : si un cercle ne couvre pas tous les points, on l'agrandit pour couvrir l'ancien cercle, plus au moins un nouveau point.



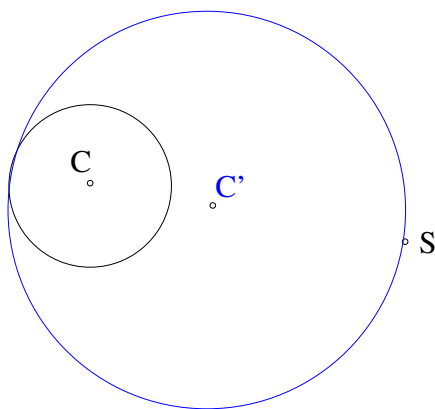




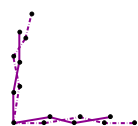
# Algorithme incrémental

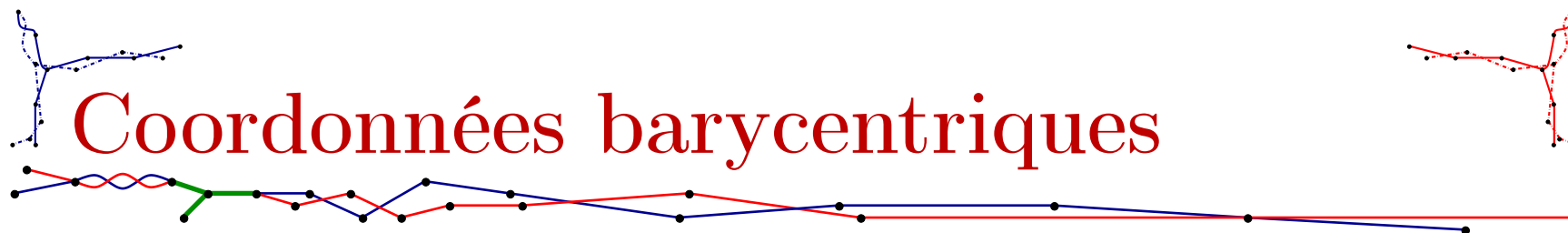


IDÉE : si un cercle ne couvre pas tous les points, on l'agrandit pour couvrir l'ancien cercle, plus au moins un nouveau point.

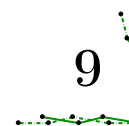
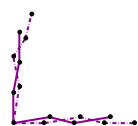
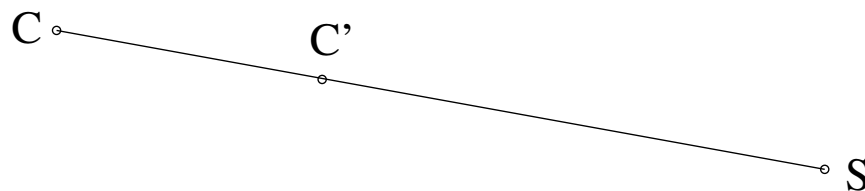


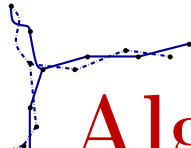
QUESTION : coordonnées de  $C'$  sachant  $C$ ,  $S$ , ancien rayon  $r$  ?



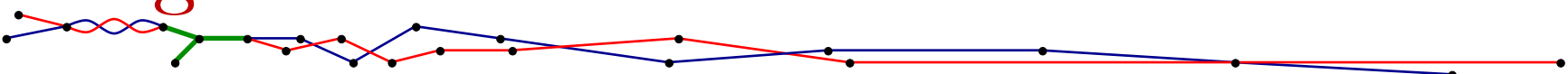
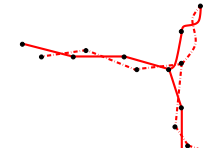


FORMULE :  $C' = \alpha \cdot C + \beta \cdot S$ , avec  $\alpha = \frac{|C'S|}{|CS|}$  et  $\beta = \frac{|C'C|}{|CS|}$ .

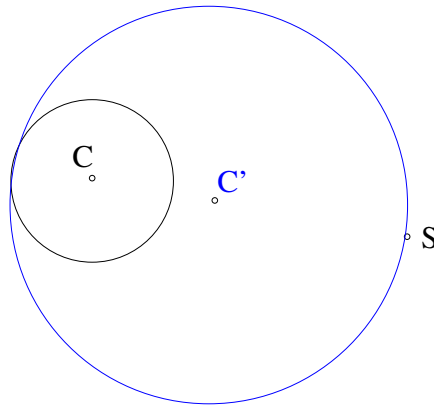




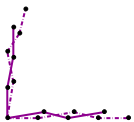
# Algorithme incrémental

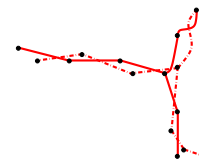
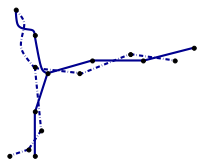


IDÉE : si un cercle ne couvre pas tous les points, on l'agrandit pour couvrir l'ancien cercle, plus au moins un nouveau point.

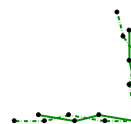
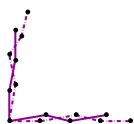


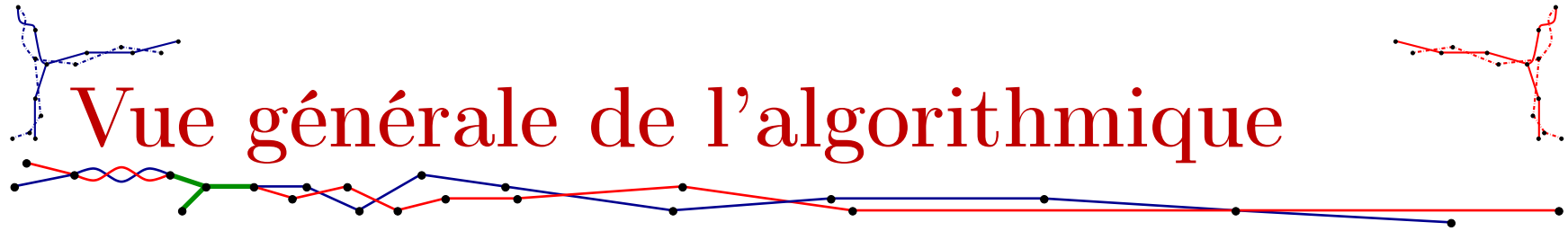
QUESTION : coordonnées de  $C'$  sachant  $C$ ,  $S$ , ancien rayon  $r$  ?





Peut on faire mieux ?

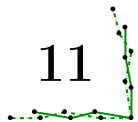
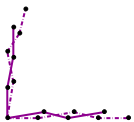


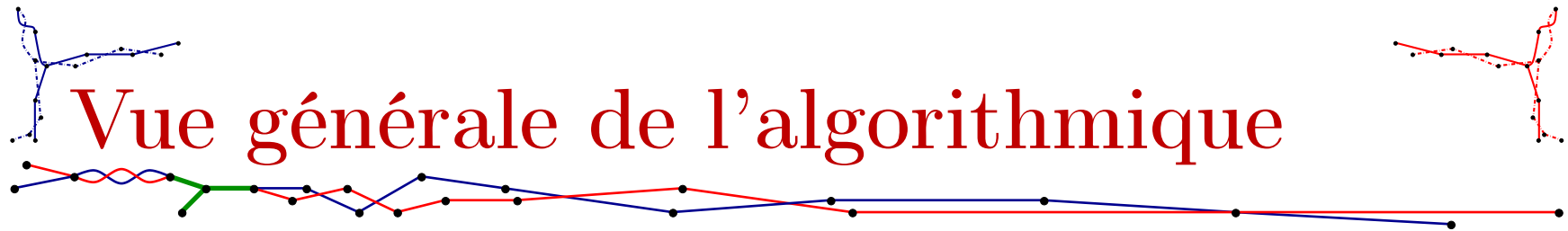


# Vue générale de l'algorithmique

BRUTE-FORCE : parcours exhaustif de l'espace de recherche

EXEMPLE : pour toute coordonnée possible du centre du cercle, pour toute valeur possible du rayon du cercle, vérifier si tous les points sont couverts.



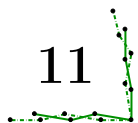
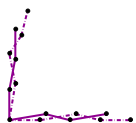


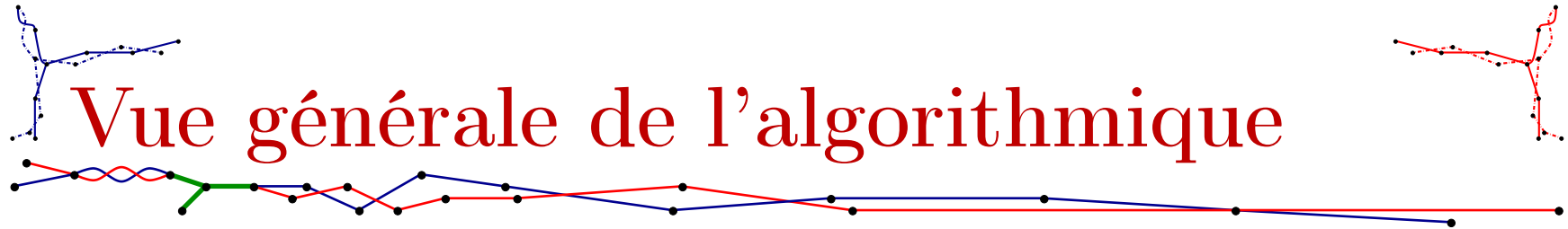
# Vue générale de l'algorithmique

RECHERCHE ORDONNÉE : réorganisation de l'espace de recherche

EXEMPLE : pour toute coordonnée possible du centre *définie à partir de deux ou de trois points de la liste*, soit  $\times \times \times$  la seule valeur *intéressante du rayon*, vérifier si tous les points sont couverts.

EXEMPLE : algorithme naïf pour cercle couvrant.

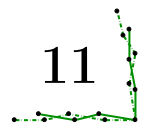
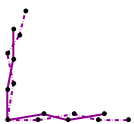




# Vue générale de l'algorithmique

RECHERCHE PARCIMONIEUSE : réorganisation de l'espace de recherche  
+ localisation

IDÉE : filtrer l'espace de recherche par un précalcul.

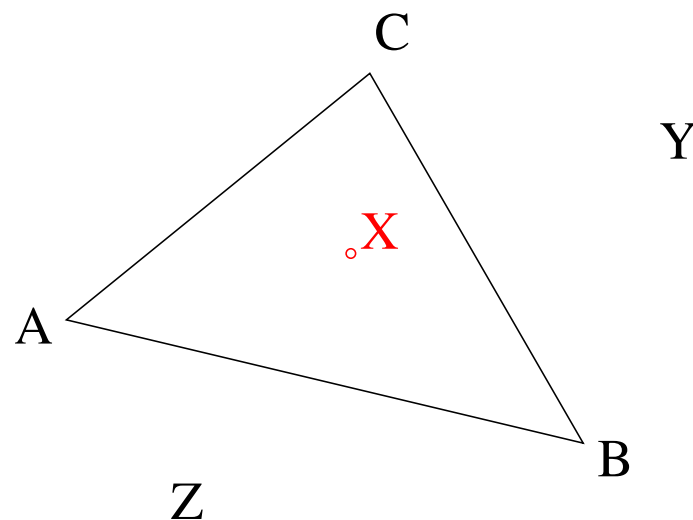




# Borner la recherche par un précalcul



IDÉE : filtrer le “INPUT” pour écarter les zones de recherche inutiles



QUESTION :  $X$  est inutile, comment le détecter, numériquement ?





CONCLUSION :

- algorithme naïf
- technique incrémental
- technique de filtrage (précalcul)

QUESTION :

- meilleur précalcul ? (voir TME pour une réponse)

