Algorithmique TP 21 : Algorithmique géométrique

Cercle circonscrit d'un ensemble de points

Soit $P_1, P_2, ..., P_n$ un ensemble de n points du plan. L'algorithme suivant calcule le plus petit disque contenant les n points. Cet algorithme incrémental est fondé sur la propriété suivante (admise) :

> *Soit S un ensemble de points et P un point. Soit D le plus petit disque contenant les points de S. Si P n'est pas dans D alors le bord du plus petit disque contenant les points de S et le point P passe par P.*

MiniDisc($P_1, P_2, ..., P_n$)
  $D \leftarrow$ Mini2(P_1, P_2)
  `Pour` $i = 3$ `jusqu'a` n `Faire`
    `Si` Outside(P_i, D) `Alors`
      $D \leftarrow$ Mini2(P_1, P_i)
      `Pour` $j = 2$ `jusqu'a` $i - 1$ `Faire`
        `Si` Outside(P_j, D) `Alors`
          $D \leftarrow$ Mini2(P_j, P_i)
          `Pour` $k = 1$ `jusqu'a` $j - 1$ `Faire`
            `Si` Outside(P_k, D) `Alors`
              $D \leftarrow$ Mini3(P_k, P_j, P_i)
  Retourner D

---

où Mini2 (resp. Mini3) retourne le plus petit disque contenant 2 points (resp. 3 points) et Outside teste si un point est en dehors d'un disque.

1. Implémenter cet algorithme en OCaml. On pourra utiliser les fonctions `mini2` et `mini3` du fichier `mini3.ml` dans `~serveur/PROG/algo`.
2. Illustrer cet algorithme avec une fenêtre graphique. Utiliser la fonction suivante pour générer un tableau de points aléatoires :

```
#let random_points = fun n ->
#  Array.init n (fun _ -> Random.int 500, Random.int 500);;
val random_points : int -> (int * int) array = <fun>
```

3. En utilisant la fonction `Sys.time`, évaluer statistiquement la complexité de l'algorithme, c-à-d la relation entre le temps de calcul et la taille des données (le nombre de points).