TD7 : Géométrie algorithmique

1. Plus proches voisins

Soit T un ensemble de n points du plan euclidien. Pour un quadrillage, dont la maille est de taille δ, et A un point de T, on définit le voisinage de A, noté $V^\delta(A)$, comme l'ensemble des points de T dans les neuf cellules autour de A. On note d^* la plus proche distance entre deux points.

1.1. Montrer la proposition suivante.

a. $d(A,B) \geq 3\delta \Rightarrow B \notin V^\delta(A)$;

b. $d(A,B) \leq \delta \Rightarrow B \in V^\delta(A)$;

c. si $\frac{\delta}{3} \leq d^* \leq \delta$

(i) pour tout A, $V^\delta(A)$ contient au plus un nombre constant de points

(ii) $d(A,B) = d^* \Rightarrow A \in V^\delta(B) \wedge B \in V^\delta(A)$.

Donnons à présent un algorithme probabiliste de calcul de d^* basé sur un crible.

Algorithm 1 Crible probabiliste

$S_0 := T$ et $i := 1$
while $S_{i-1} \neq \emptyset$ **do**
  Choisir $A \in S$ aléatoirement
  Calculer $d_{A,S} = \min_{B \in S \setminus \{A\}} d(A,B)$
  Construire un quadrillage de maille $\delta = \frac{d_{A,S}}{3}$
  $X := \{B \in S \mid V^\delta(B) = \{B\}\}$
  $S_i := S_{i-1} \setminus X$
  $i := i + 1$
end while
$A^* := A$ et $i^* := i - 1$
Construire un quadrillage de maille $\delta = d_{A^*,S_{i^*}}$
for $B \in T$ **do**
  Calculer $\delta^*(B) := \min_{C \in V^\delta(B) \setminus \{B\}} d(B,C) \leq$
end for
  return $\min_{B \in T} \delta^*(B)$

1.2. Montrer qu'en entrée de la boucle **for**, avec $\delta = d_{A^*,S_{i^*}}$, on a $\frac{\delta}{3} \leq d^* \leq \delta$.

1.3. Conclure sur la correction de l'algorithme.

Pour en étudier la complexité, on fait l'hypothèse suivante :

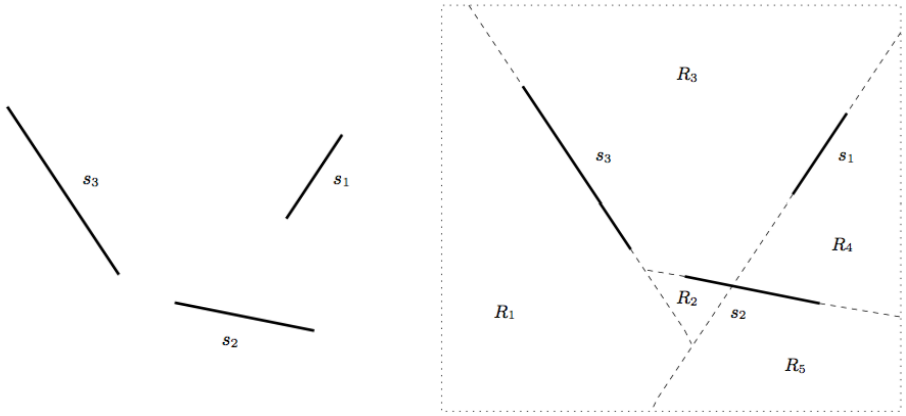
— la mise à jour de S, c'est-à-dire une itération de la boucle **while**, peut être implémentée en $O(|S_i|)$ en moyenne et espace linéaire grâce à des fonctions de hachage.

On admet donc que l'itération i de la boucle **while** est de complexité linéaire en $|S_i|$. Montrons que l'espérance de la taille cumulée des S_i est en $O(n)$.

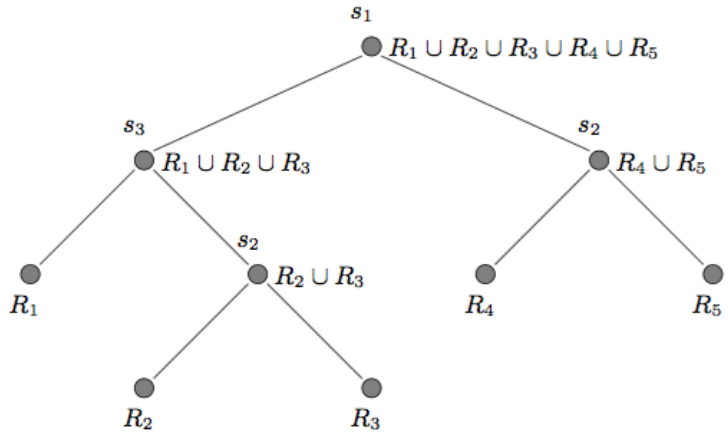
1.4. Montrer que $\mathbb{E}(\sum_{i=1}^{i^*} |S_i|) \leq 2n$ et conclure.

2. Bipartition du plan

Étant donnés n segments (orientés) du plan qui ne s'intersectent pas et qui sont non alignés, on cherche à calculer un arbre de décision qui définit une autopartition du plan, c'est-à-dire une partition par les droites orientées qui prolongent les segments. Étant donné un point du plan, on cherche alors à savoir dans quelle partie il se trouve. Chaque nœud de l'arbre de décision est associé à un segment. Soit on détermine que le point est sur la droite orientée définie par le segment (et potentiellement sur le segment), soit on continue à chercher un segment qui peut contenir le point, en déterminant si le point est à gauche ou à droite de la droite.
On représente ci-dessous un exemple avec trois segments, et une autopartition du plan possible par ces trois segments.



L'arbre de décision associé est alors le suivant.



2.1. Pourquoi dans l'arbre binaire peut-il y a voir plusieurs nœuds étiquetés par le même segment ?

Considérons l'algorithme probabiliste suivant, qui consiste à

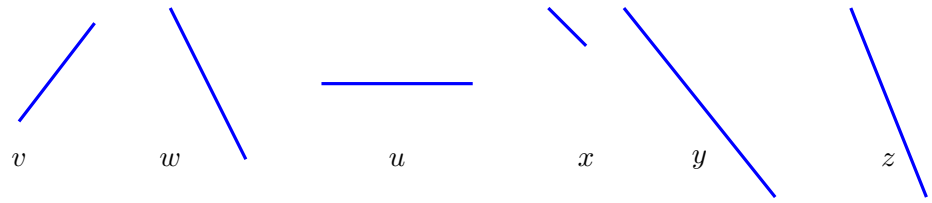
1. choisir de façon aléatoire uniforme un segment de la liste,
2. séparer la région par la droite portée par ce segment,
3. appliquer récursivement l'algorithme dans les deux sous-régions, avec la liste des segments (ou parties de segments) qui leur appartient.

Algorithm 2 Bipartition probabiliste

`RandAuto`(L, R)
if L vide **then**
Return arbre vide
else
Choisir un segment e de L de façon aléatoire uniforme
$R_g = R \cap$ demi-plan à gauche du segment e
$R_d = R \cap$ demi-plan à droite du segment e
L_g = ensemble des segments obtenus à partir des segments de $L \setminus \{e\}$ en les intersectant avec R_g.
L_d = ensemble des segments obtenus à partir des segments de $L \setminus \{e\}$ en les intersectant avec R_d.
Return Arbre de racine (e, R) ayant pour sous-arbre gauche `RandAuto`(L_g, R_g) et sous-arbre droit `RandAuto`(L_d, R_d)
end if

2.2. Donner une borne sur le nombre de nœuds en fonction de n et du nombre de fois qu'un segment est séparé en deux régions.

On souhaite montrer que l'arbre de recherche a en espérance $O(n \log(n))$ nœuds.
Pour tous les segments u, v, avec $u \neq v$, on définit $h(u, v)$ comme le nombre de segments qui sont touchés par la droite $L(u)$ avant de toucher v, plus 1. Par exemple :



$$h(u, v) = 2\,;\ h(u, w) = 1\,;\ h(u, x) = +\infty\,;\ h(u, y) = 1\,;\ h(u, z) = 2.$$

On utilisera la notation $u \dashv v$ pour représenter l'événement « $L(u)$ coupe v dans la partition construite », et $X_{u,v}$ la variable aléatoire qui vaut 1 si cet événement est vrai et 0 sinon.

Supposons $h(u,v) = i$, et soient $u_1, u_2, \ldots, u_{i-1}$ les segments qui sont touchés par $L(u)$ avant de toucher v.

2.3. Borner $\mathbb{P}(u \dashv v)$ en fonction de $h(u,v)$.

2.4. Pour tout segment u, et pour tout $i \in \mathbb{N}$, combien y a-t-il au plus de segments v tels que $h(u,v) = i$?

2.5. En déduire une borne sur l'espérance du nombre de coupes de segments distincts $\mathbb{E}(\sum_u \sum_{v \neq u} X_{u,v})$ et conclure.