Chapitre 10

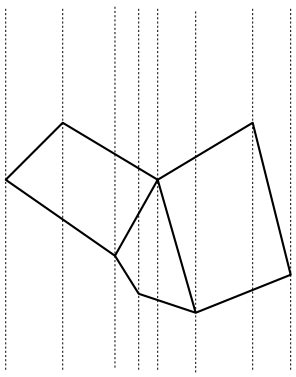
Localisation

Le problème de la localisation consiste, étant donnée une collection fixée de n objets dans $\mathbb{R}^d$, à trouver (les) l'objet(s) de la collection contenant un point de requête quelconque. C'est d'une certaine manière le problème dual de la recherche multidimensionnelle du chapitre 11. Pour $d = 1$, cela revient à trouver les intervalles (ce sont les objets de la collection) contenant le point de requête. Les structures *d'arbre d'intervalles* et *d'arbre de segments* permettent de répondre efficacement à ce type de requête mono-dimensionnelle en temps $O(\log n + k)$ où k est le nombre d'intervalles contenant le point de requête. On pourra consulter [dBCvKO08, chap. 10] pour des détails sur ces structures. Dans ce chapitre on se restreindra au cas $d = 2$ où les objets forment une subdivision d'une partie du plan, c'est-à-dire que les régions s'identifient aux faces de la subdivision. Ce cas est en particulier utile pour les interactions avec les systèmes d'information géographique.

Les deux sections suivantes sont essentiellement des traductions du chapitre 6 de de Berg et al. [dBCvKO08].

10.1 Localisation par découpe en bandes verticales

On suppose ici que l'ensemble des régions est donné sous forme d'une carte planaire combinatoire (cf. section 9.2) de taille n, i.e. possédant n arêtes. Notons que si la carte est connexe, on peut marquer, lors d'un simple parcours en temps linéaire, chaque demi-arête de la carte avec l'indice de la face située à sa gauche. La localisation consiste alors à reporter l'indice de la face de la carte combinatoire contenant le point de requête. La méthode qui suit est due à Dobkin et Lipton [DL76] et repose sur une découpe de la carte en bandes verticales. On commence par tracer par chaque sommet de la carte une droite verticale comme sur la figure ci-après. Ces droites verticales découpent le plan en bandes verticales. À l'intérieur d'une bande les régions traversées sont ordonnées verticalement. Ainsi pour localiser un point p de requête il suffit d'effectuer deux recherches mono-dimensionnelles : une recherche *horizontale* pour déterminer la bande contenant p, suivie d'une recherche *verticale* permettant d'identifier la région, ou face, de la carte contenant p. Chaque recherche s'effectuant sur un ensemble de taille $O(n)$, la localisation peut s'effectuer en temps $O(\log n)$ à l'aide d'une structure de dictionnaire classique



(déterministe) ou randomisée (cf. chapitre 5).

Plus précisément, pour construire la structure de recherche, on commence par ranger les sommets de la carte dans un dictionnaire en utilisant l'ordre lexicographique sur leurs coordonnées pour les comparaisons. On effectue ensuite un balayage des sommets de la carte de gauche à droite. Dit autrement, on parcourt les sommets dans l'ordre lexicographique. Pour chaque sommet q balayé on construit une structure de recherche $V[q]$ correspondant à la bande verticale située à droite de q. Cette structure de recherche est elle même un dictionnaire sur les arêtes, ordonnées verticalement, qui traversent $V[q]$. Clairement $V[q]$ peut être obtenue à partir du dictionnaire de la bande précédente en ajoutant les arêtes incidentes à q et tournées vers la droite et en supprimant les arêtes incidentes à q et tournées vers la gauche.

Dans le pseudo-code suivant H désigne le dictionnaire des sommets de la carte pour l'ordre lexicographique et $V[q]$ désigne un dictionnaire sur les arêtes traversant la bande verticale à droite du sommet q.

Construction de la structure de recherche
On suppose les sommets $p_0, ..., p_m$ de la carte ordonnés de gauche à droite.

Initialiser un dictionnaire V_{aux} à vide
Pour $i := 0$ à m **faire**
  Insérer p_i dans le dictionnaire H
  $V[p_i] := V_{aux}$
  Pour chaque demi-arête a d'origine p_i **faire**
    Si a est orientée vers la gauche **Alors**
      Supprimer la demi-arête opposée de a dans $V[p_i]$
    Sinon Insérer a dans $V[p_i]$
  $V_{aux} := V[p_i]$

Localisation d'un point p
Rechercher dans H le sommet q de la carte juste à gauche de p
Si $p == q$ **Alors** renvoyer cette information
Sinon Si $p == NILL$ (p est à l'extrême gauche de la carte) **Alors**
  retourner la face la plus à gauche de la carte
Sinon rechercher la demi-arête a (orientée vers la droite) sous p dans $V[q]$
  Si $a == NILL$ **Alors** retourner la face la plus basse de la carte

Sinon Si p est sur a **Alors** retourner cette information
Sinon retourner la région à gauche de a

Le temps de construction de la structure de recherche, hormis la recopie des dictionnaires verticaux est $O(n \log n)$: chaque demi-arête est insérée et supprimée une fois dans l'ensemble des bandes au cours du balayage. Cela dit on recopie $O(n)$ fois la structure V_{aux}. Comme cette structure peut avoir une taille $O(n)$ ceci donne finalement une construction en temps et espace $O(n^2)$. On peut cependant améliorer ces résultats en utilisant des dictionnaires (arbres) "persistants".

Dans un dictionnaire persistant on ordonne dans le temps les opérations d'insertion et de suppression et on peut faire une requête du type : rechercher un élément dans le dictionnaire tel qu'il se trouvait au moment de la $k$ième opération. On peut obtenir sur n éléments et m opérations un temps d'insertion/suppression en $O(\log n)$, un temps de recherche en $O(\log m)$ et une taille amortie en $O(n)$. Appliqué au présent problème cela donne

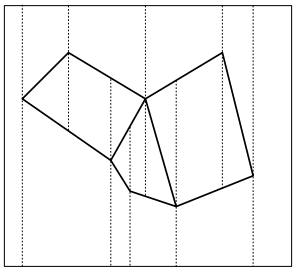
Théorème 10.1 *Soit S un ensemble de n segments formant une carte. L'algorithme exposé calcule en temps $O(n \log n)$ une structure de recherche associée à S de taille $O(n)$. De plus pour tout point p du plan le temps de localisation dans cette structure est $O(\log n)$.*

Référence :

- Éléments d'algorithmique. D. Beauquier, J. Berstel et P. Chrétienne. MASSON, 1992.

10.2 Localisation par décomposition trapézoïdale

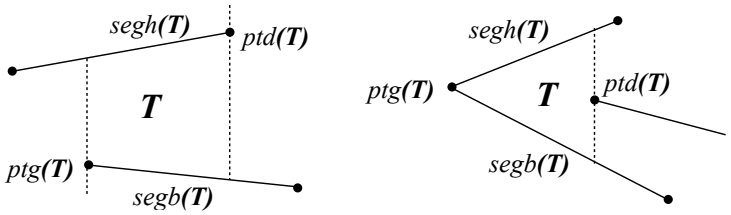
On peut obtenir un temps de recherche équivalent à l'approche par découpe verticale avec une structure de recherche de taille linéaire sans utiliser de structure de données persistante. Pour cela on trace comme précédemment une verticale par chaque sommet de la subdivision mais on stoppe cette verticale dès que l'on rencontre un segment de la carte. Afin de simplifier l'exposé, on commence par entourer la carte d'une grande boîte rectangulaire et on ne s'intéresse qu'à l'intérieur de cette boîte. En particulier chaque verticale est bornée (cf. figure suivante). Chaque face de la subdivision ainsi obtenue est



convexe : les angles en chaque sommet d'une telle face (sommet d'arête ou intersection d'une verticale et d'une arête) sont plus petits que π. Chaque face a donc au plus deux

côtés verticaux et donc au plus deux non-verticaux (un sommet commun à deux segments non-verticaux est l'origine d'une verticale) et donc exactement deux non-verticaux puisque bornée. Dit autrement chaque face de la subdivision est géométriquement un *trapèze* possédant quatre côtés dont deux verticaux ou trois côtés dont un vertical. La subdivision ainsi obtenue porte ainsi le nom de *carte des trapèzes*.

Un trapèze est entièrement défini par les deux segments qui le bordent en haut et en bas et par les deux sommets qui ont permis de construire ses parois verticales gauche et droite. On note *segh*, *segb*, *ptg* et *ptd* respectivement ces quatre attributs. La figure suivante montre deux exemples de trapèzes avec les attributs correspondants.



On distingue cinq configurations pour le côté vertical gauche d'un trapèze comme illustré figure 10.2 :

1. il est réduit à un point qui est l'extrémité gauche commune aux segments supérieur et inférieur.
2. c'est l'extension verticale du sommet gauche du segment inférieur qui aboutit sur le segment supérieur.
3. même chose en échangeant supérieur et inférieur.
4. c'est l'extension verticale de l'extrémité droite d'un autre segment qui le coupe donc en deux.
5. c'est un côté de la boîte englobante.

Il existe de même cinq configurations distinctes pour le côté vertical droit d'un trapèze.

Lemme 10.2 *La carte des trapèzes a un nombre linéaire de faces, arêtes et sommets en fonction du nombre n de segments de la carte planaire d'origine.*

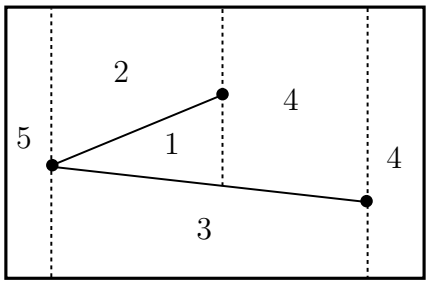
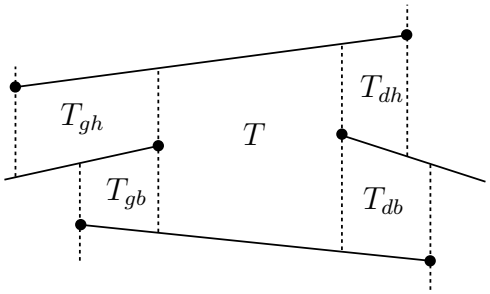


FIGURE 10.1 – Le côté vertical gauche d'un trapèze peut se trouver dans cinq configurations distinctes.

Preuve : Chaque sommet d'origine (au plus $2n$) donne lieu à au plus trois sommets : lui-même + 2 sommets pour les extensions inférieures et supérieures des verticales. Avec les 4 sommets de la boîte englobante cela donne au plus $6n+4$ sommets en tout. De plus chaque sommet d'origine donne lieu à 4 nouveaux segments : ses deux extensions verticales + 2 segments obtenus par subdivision des segments rencontrés par ses extensions verticales. La carte des trapèzes possèdent donc au plus $4.2n+n+4 = 9n+4$ segments en tenant compte de la boîte englobante. La relation d'Euler permet finalement de borner trivialement le nombre de faces par le nombre d'arêtes plus un. □

Exercice 10.3 *Montrer qu'on peut borner le nombre de faces par $3n+1$ en associant judicieusement chaque face à l'un des segments intersectant le bord de cette face.*

On associe à la carte des trapèzes une structure $\mathcal{T}$ la représentant, mais plus simple que la représentation classique en demi-arêtes, et une structure $\mathcal{R}$ de recherche pour localiser un point. Nous supposerons par la suite les sommets de la carte en position générale de sorte que chaque droite verticale passe par au plus un sommet. De cette manière chaque trapèze possède au plus quatre voisins adjacents à ses parois verticales. On note T_{gh}, T_{gb}, T_{dh} et T_{db} ces voisins comme sur la figure ci-dessous.



• La structure $\mathcal{T}$ se présente sous forme d'une liste d'enregistrements correspondant à chacun des trapèzes. Chaque enregistrement contient les pointeurs sur les attributs *segh*, *segb*, *ptg*, *ptd*, T_{gh}, T_{gb}, T_{dh} et T_{db} précédemment définies.

• La structure $\mathcal{R}$ est un graphe orienté sans cycle possédant une unique racine et exactement une feuille par trapèze de $\mathcal{T}$. Chaque trapèze de $\mathcal{T}$ pointe sur sa feuille correspondante dans $\mathcal{R}$ et réciproquement [1]. Les noeuds internes de $\mathcal{R}$ ont tous deux enfants et pointent soit sur un sommet soit sur un segment de la subdivision originale. Chaque noeud interne ν est la racine dans $\mathcal{R}$ d'un sous-graphe orienté sans cycle qui est une structure de recherche pour l'intersection d'une certaine zone Z_ν avec $\mathcal{T}$. Cette zone est définie récursivement comme suit. La zone de la racine de $\mathcal{R}$ est l'intérieur de la boîte englobante. Si ν pointe sur un sommet alors la zone de sa fille gauche (droite) est la partie de Z_ν à gauche (droite) de ce sommet. Si ν pointe sur un segment alors la zone de sa fille gauche (droite) est la partie de Z_ν au dessus (au dessous) de ce segment.

Pour effectuer une recherche d'un point p dans la structure de recherche $\mathcal{R}$, on part de la racine et on descend progressivement jusqu'à une feuille qui pointe sur le trapèze

1. On peut également construire la structure $\mathcal{T}$ directement sur les feuilles de $\mathcal{R}$

contenant p. À chaque noeud pointant sur un sommet on s'oriente vers la fille gauche ou droite selon que p est à gauche ou à droite de ce sommet. De même, à chaque noeud pointant sur un segment on s'oriente vers la fille gauche ou droite selon que p est au dessus ou au dessous de ce segment. La feuille atteinte correspond au trapèze contenant p. Ce trapèze peut lui-même être indéxé par la face le contenant dans la carte d'origine (via la face à gauche d'une demi-arête de son segment inférieur) fournissant ainsi le résultat cherché.

$\mathcal{T}$ et $\mathcal{R}$ sont construits de manière incrémentale randomisée en insérant successivement chaque segment s_i de la subdivision d'origine, pris dans un ordre aléatoire, dans les structures $\mathcal{T}_{i-1}$ et $\mathcal{R}_{i-1}$ associées aux $i-1$ premiers segments insérés.

L'algorithme de construction est le suivant :

Données : Une carte planaire formée de n segments $s_1, \ldots, s_n$ énumérés dans un ordre aléatoire.
Sorties : Les structures $\mathcal{T}$ et $\mathcal{R}$ associées.

1. Déterminer une boite englobante contenant les n segments (strictement) et initialiser $\mathcal{T}_0$ et $\mathcal{R}_0$

2. Pour $i := 1$ à n **faire**

3. Localiser grâce à $\mathcal{R}_{i-1}$ le trapèze T_1 de $\mathcal{T}_{i-1}$ contenant l'extrémité gauche de s_i.

4. En déduire pas simple parcours dans $\mathcal{T}_{i-1}$ les trapèzes $T_1, T_2, \ldots, T_k$ traversés par s_i.

5. Remplacer chaque trapèze $T_1, T_2, \ldots, T_k$ de $\mathcal{T}_{i-1}$ par sa subdivision, induite par s_i, et chacune composée de 2,3 ou 4 trapèzes.

6. Faire de même dans $\mathcal{R}_{i-1}$ en remplaçant cette fois chaque trapèze feuille par un sous-arbre de recherche élémentaire sur sa subdivision.

7. Fusionner les trapèzes des subdivisions pour rétablir la décomposition trapézoïdale induite par $s_1, \ldots, s_i$.

finpour

Pour l'étape 6, il y a essentiellement 3 types de sous-arbres élémentaires possibles selon qu'un trapèze est découpé en 2,3 ou 4 morceaux. Les noeuds internes des ces sous-arbres correspondent au segment s_i ou à ses extrémités. La figure suivante résume les étapes 6 et 7 sur deux exemples.

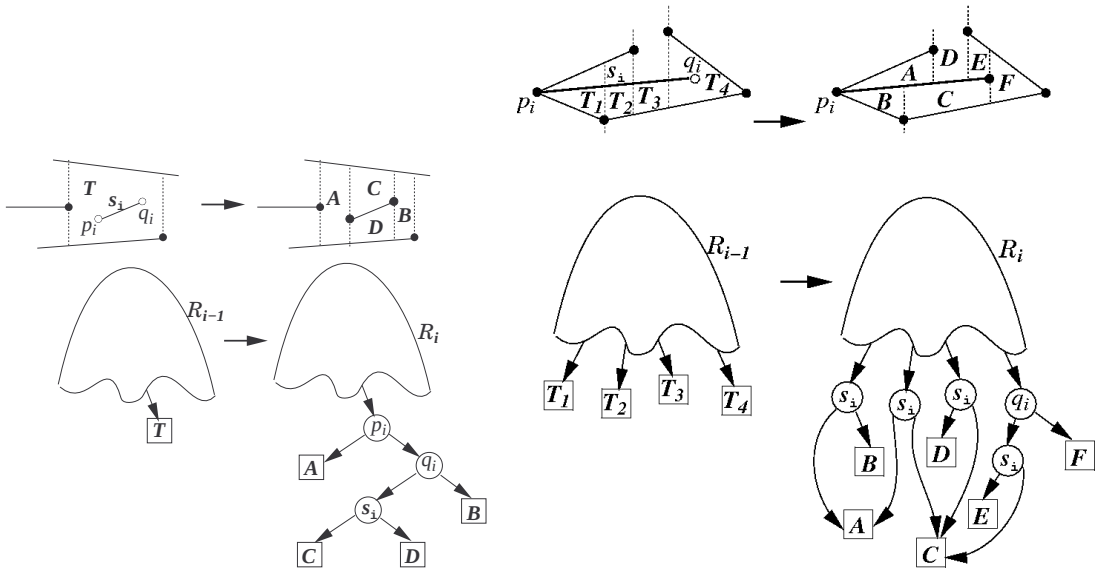
La structure de recherche $\mathcal{R}$ dépend évidemment de l'ordre d'insertion des n segments de la carte d'origine. On considère $\mathcal{R}$ comme une variable aléatoire sur l'espace des permutations (sur l'insertion) des segments muni de la loi uniforme.

Lemme 10.4 *Soit p un point fixé du plan. Le temps moyen de localisation de p dans $\mathcal{R}$ est un $O(\log n)$.*

Preuve : Le temps de localisation de p est clairement proportionnel à la longueur de son chemin de recherche $l(p)$ dans $\mathcal{R}$.

Remarque : à chaque insertion d'un segment dans la construction incrémentale la hauteur de $\mathcal{R}$ augmente d'au plus 3. Un chemin de recherche a donc une longueur bornée par $3n$.

On note X_i la variable aléatoire dénombrant les arcs de $l(p)$ apparus dans $\mathcal{R}$ lors de l'insertion de s_i, i.e. les arcs de $l(p)$ présents dans $\mathcal{R}_i$ mais pas dans $\mathcal{R}_{i-1}$. En notant Y_i



la variable aléatoire valant 1 si ce chemin est rallongé à l'étape i et 0 sinon, on a suivant la remarque précédente

$$|l(p)| = \sum_i X_i \leq 3 \sum_i Y_i.$$

D'où en prenant les espérances

$$E(|l(p)|) \leq 3 \sum_i E(Y_i) = 3 \sum_i P(Y_i = 1).$$

Or $l(p)$ est rallongé à l'étape i si et seulement si le segment s_i intersecte le trapèze contenant p dans la carte $\mathcal{T}_{i-1}$ c.-à-d. si et seulement si le trapèze Δ_i contenant p dans la carte $\mathcal{T}_i$ n'existe pas dans la carte $\mathcal{T}_{i-1}$. Mais $\Delta_i \notin \mathcal{T}_{i-1}$ si et seulement si l'un des quatre attributs qui le définissent, ($segh(\Delta_i)$, $segb(\Delta_i)$, $ptg(\Delta_i)$ ou $ptd(\Delta_i)$) n'est pas présent dans $\mathcal{T}_{i-1}$. Selon le principe de *l'analyse arrière* fixons le sous-ensemble S_i des i premiers segments insérés. Notons que $\mathcal{T}_i$ – et donc Δ_i – ne dépend que de S_i et non de l'ordre d'insertion des i premiers segments. On a alors

$$\begin{aligned} P(Y_i = 1 \mid S_i) \quad \leq \quad & P(segh(\Delta_i) \notin \mathcal{T}_{i-1} \mid S_i) + P(segb(\Delta_i) \notin \mathcal{T}_{i-1} \mid S_i) \\ & + P(ptg(\Delta_i) \notin \mathcal{T}_{i-1} \mid S_i) + P(ptd(\Delta_i) \notin \mathcal{T}_{i-1} \mid S_i) \end{aligned}$$

Puisque chaque segment de S_i a une probabilité $1/i$ d'être inséré le dernier, on a $P(segh(\Delta_i) \notin \mathcal{T}_{i-1} \mid S_i) = P(segh(\Delta_i) = s_i \mid S_i) = 1/i$ et de même $P(segb(\Delta_i) \notin \mathcal{T}_{i-1} \mid S_i) = 1/i$. Si plusieurs segments de S_i ont $ptg(\Delta_i)$ pour extrémité alors $P(ptg(\Delta_i) \notin \mathcal{T}_{i-1} \mid S_i)$ est nul car $ptg(\Delta_i)$ est alors à coup sûr dans $\mathcal{T}_{i-1}$. Sinon, en notant s l'unique segment de S_i d'extrémité $ptg(\Delta_i)$, on a $P(ptg(\Delta_i) \notin \mathcal{T}_{i-1} \mid S_i) = P(s = s_i \mid S_i) = 1/i$. D'où dans tous les cas $P(ptg(\Delta_i) \notin \mathcal{T}_{i-1} \mid S_i) \leq 1/i$. En résumé,

$$P(Y_i = 1 \mid S_i) \leq 4/i$$

D'où inconditionnellement $P(Y_i = 1) \leq 4/i$. On en déduit $E(|l(p)|) \leq 12H_n = O(\log n)$. □

Lemme 10.5 *La taille moyenne de la structure $\mathcal{R}$ de recherche est linéaire.*

Preuve : On vérifie que si k_i est le nombre de nouveaux trapèzes créés à l'étape i, alors le nombre de noeuds internes ajoutés à $\mathcal{R}_{i-1}$ est $k_i - 1$: soient $T_1, \ldots, T_k$ les trapèzes intersectés par s_i. Si T_j est subdivisé en $n(T_j)$ morceaux par l'introduction de s_i alors son sous-arbre élémentaire a $n(T_j) - 1$ noeuds internes. Soit un total de $N_i = \sum_j n(T_j) - k$ noeuds internes créés par l'introduction de s_i. Mais le nombre de nouveaux trapèzes est $k_i = \sum_j n(T_j) - (k-1) = N_i + 1$ car il faut effectuer $k-1$ fusions (le nombre de trapèzes intersectés moins 1) parmi les $\sum_j n(T_j)$ morceaux précédemment obtenus pour créer les nouveaux trapèzes.

Puisque la carte des trapèzes a une taille linéaire (lemme 10.2) $\mathcal{R}$ possède $O(n)$ feuilles plus $\sum_i (k_i - 1)$ noeuds internes. D'où $E(|\mathcal{R}|) = O(n) + \sum_i E(k_i)$. On utilise à nouveau une analyse arrière : pour S_i fixé, un trapèze de $\mathcal{T}_i$ a une probabilité au plus $4/i$ d'avoir été créé par l'insertion de s_i. Le nombre moyen $E(k_i)$ de trapèzes créés à l'étape i, qui est la somme de cette probabilité pour tous les trapèzes de $\mathcal{T}_i$, est donc majoré par $O(i)4/i = O(1)$ puisqu'il y a $O(i)$ trapèzes sans $\mathcal{T}_i$. □

En ce qui concerne le temps moyen de construction on remarque que le coût d'insertion de s_i est le temps de localisation de son extrémité gauche plus $O(k_i)$ pour parcourir les trapèzes traversés et mettre à jour $\mathcal{R}_{i-1}$ et $\mathcal{T}_{i-1}$. Le temps moyen de localisation est $O(\log i)$ par le lemme 10.4 et $E(k_i) = O(1)$ d'après ce qui précède. On en déduit un coût moyen total de $O(n \log n)$.

En résumé,

Théorème 10.6 *Soit S un ensemble de n segments formant une carte planaire. L'algorithme exposé calcule en temps moyen $O(n \log n)$ une structure de recherche pour la carte des trapèzes associée à S de taille moyenne $O(n)$. De plus pour tout point p du plan le temps moyen de localisation dans cette structure est $O(\log n)$.*

Notons que les moyennes sont ici prises relativement aux permutations sur les segments et que p est fixé une fois pour toute. En particulier, ce résultat ne dit rien sur le temps maximal moyen relativement à l'ensemble des points requêtes. Autrement dit il se pourrait que pour la plupart des permutations des segments il existe un point, dépendant de la permutation considérée, ayant un "mauvais" temps de localisation. En fait le temps maximal moyen est lui-même majoré par $O(\log n)$. Pour le voir on commence par remarquer que deux points appartenant à la même cellule de l'arrangement des droites supports des n segments et des $2n$ verticales aux sommets des segments partagent un même chemin de recherche dans $\mathcal{R}$ quelle que soit la permutation considérée. Pour chaque permutation, il y a donc $O(n^2)$ chemins de recherche distincts possibles [2] qui ont bien sûr chacun une longueur moyenne en $O(\log n)$. Si on montre que pour chaque chemin cette moyenne n'est dépassée que pour un sous-ensemble suffisamment petit de permutations alors ce sera encore le cas pour le maximum des longueurs de ces $O(n^2)$ chemins. C'est l'objet de ce qui suit.

2. Voir le théorème 9.3 sur les arrangements de droites.

Pour faire une analyse plus fine de la queue de distribution on utilise la technique de Chernoff. Le problème est ici que les variables aléatoires Y_i introduites plus haut ne sont pas indépendantes (exercice : le vérifier pour S formé des cotés d'un triangle et un point interne à ce triangle). On utilise une définition modifiée qui les rend indépendantes. Pour cela on considère le diagramme de Hasse du treillis (booléen) des parties de S pour l'inclusion. Ce treillis a $\binom{n}{i}$ noeuds de *niveau* i ayant chacun i arcs entrants et $n-i$ arcs sortants. Les permutations de S sont en bijection avec les chaînes maximales de ce treillis (joignant vide à plein) et chaque arc du treillis correspond à un segment de S (le segment ajouté). On fixe un point p et on marque un arc si la *suppression* du segment correspondant modifie le trapèze contenant p dans la carte associée à la tête de cet arc. On a vu que chaque noeud a au plus 4 arcs entrants marqués. L'astuce est de compléter systématiquement à 4 le nombre d'arcs entrants marqués (pour les noeuds de niveaux 1, 2, 3 on marque tous les arcs entrants). Le point p étant fixé, on note X_i la variable aléatoire définit sur les chaînes maximales du diagramme (donc les permutations) valant 1 si l'arc entre les niveaux $i-1$ et i est marqué et 0 sinon. Ainsi la longueur du chemin de recherche de p est majorée par $3\sum_i X_i$.

Lemme 10.7 *Les X_i sont mutuellement indépendantes.*

Preuve : Commençons par remarquer que si A est une condition portant sur des niveaux supérieurs ou égaux à i alors $P(X_i = \epsilon \mid A)$ ne dépend pas de A et vaut donc $P(X_i = \epsilon)$. Par exemple, si a est une arête du treillis entre les niveaux i et $i+1$, la probabilité que $X_i = 1$ restreinte aux permutations contenant a vaut $4/i$ si $i \geq 4$.

On raisonne ensuite par récurrence sur le nombre k de v.a. prises en compte. On suppose que les k premières variables aléatoires $X_1, \ldots, X_k$ sont mutuellement indépendantes et plus précisément que pour toute condition A_k portant sur des niveaux $\geq k$, les v.a. $X_1|A_k, \ldots, X_k|A_k$ le sont avec des probabilités uniformes par rapport à A_k. Dit autrement, pour tout $\epsilon_1, \ldots \epsilon_k \in \{0,1\}^k$, on a

$$P(X_1 = \epsilon_1 \wedge \ldots \wedge X_k = \epsilon_k \mid A_k) = \Pi_{j=1}^{k} P(X_j = \epsilon_j) \tag{10.1}$$

Notons B l'événement $(X_1 = \epsilon_1 \wedge \ldots \wedge X_k = \epsilon_k)$. On a alors (cf. exercice 1.20) pour toute condition A_{k+1} portant sur des niveaux $\geq k+1$ et pour tout $\epsilon_{k+1} \in \{0,1\}$:

$$P(B \wedge X_{k+1} = \epsilon_{k+1} \mid A_{k+1}) = P(B \mid X_{k+1} = \epsilon_{k+1} \wedge A_{k+1}) P(X_{k+1} = \epsilon_{k+1} \mid A_{k+1})$$

Ce qui vaut encore $\Pi_{j=1}^{k+1} P(X_j = \epsilon_j)$ d'après l'équation (10.1) et la remarque initiale. □

On peut maintenant appliquer la technique de Chernoff (cf. section 1.7.4) à la variable $X_p = \sum_i X_i$. Pour cela on regarde la probabilité que X_p dépasse $\lambda \ln(n+1)$. En utilisant l'inégalité de Markov on a

$$\forall t > 0,\ P(X_p \geq \lambda \ln(n+1)) = P((e^{tX_p}) > (n+1)^{t\lambda}) \leq E(e^{tX_p})/(n+1)^{t\lambda}$$

Or de part l'indépendance des X_i on a

$$E(e^{tX_p}) = \prod_i E(e^{tX_i})$$

Avec $t = \ln(5/4)$ on calcule pour $i \geq 4$, $E(e^{tX_i}) = (5/4)4/i + 1(1 - 4/i) = (i+1)/i$ et $E(e^{tX_i}) = 5/4$ pour $i = 1, 2, 3$. D'où $E(e^{tX_p}) \leq n + 1$. Finalement on trouve

$$P(X_p \geq \lambda \ln(n+1)) \leq 1/(n+1)^{\lambda \ln(5/4) - 1}$$

On a vu plus haut que X_p ne dépend que de la cellule d'un certain arrangement de taille $O(n^2)$. La probabilité que l'un des X_p définis pour chacune des cellules de l'arrangement dépasse $\lambda \ln(n+1)$ est donc majorée par $O(1/(n+1)^{\ln(5/4)\lambda - 3})$. Comme X_p est uniformément borné par $O(n)$ on en déduit que

$$\begin{aligned} E(\max_p X_p) &= \sum_k k P(\max_p X_p = k) \\ &\leq \lambda \ln(n+1) P(\max_p X_p < \lambda \ln(n+1)) + O(n) P(\max_p X_p \geq \lambda \ln(n+1)) \\ &\leq \lambda \ln(n+1) + O(1/(n+1)^{\lambda \ln(5/4) - 4}) \end{aligned}$$

Finalement, en choisissant λ tel que $\lambda \ln(5/4) \geq 4$, on obtient

Lemme 10.8 *L'espérance du temps* maximal *de recherche est* $O(\log n)$.

On montrerait de même que la probabilité que la taille (resp. le temps de construction) ne dépasse pas λn (resp. $\lambda n \log n$) est aussi proche de 1 que l'on veut si on choisit λ assez grand. Du coup la probabilité P que le temps maximal de recherche, la taille, et le temps de construction soient simultanément comme désirés est non nulle (et en fait P est aussi proche de 1 que l'on veut si on fait croître λ). On en déduit

Théorème 10.9 *Soit S l'ensemble des n segments d'une carte planaire. On peut construire en temps moyen $O(n \log n)$ une structure de recherche pour la carte des trapèzes associée à S de taille $O(n)$ et de temps de requête $O(\log n)$ dans le cas le pire.*

Pour cela on déroule l'algorithme précédent pour une permutation aléatoire jusqu'à s'apercevoir que la taille ou la profondeur de $\mathcal{R}$ dépasse λn ou $\lambda \log n$. Si c'est le cas on recommence avec une autre permutation aléatoire jusqu'à obtenir les taille et profondeur désirées. La moyenne du nombre d'essais à effectuer est $1/P$ et le temps moyen de construction est donc $O(n \log n)/P = O(n \log n)$.

Référence :

- Computational Geometry. Algorithms and applications. de Berg, van Kreveld, Overmars and Schwarzkopf. Springer 1997. pf. Springer 1997.

10.3 Localisation dans une triangulation

Une autre approche, proposée par Kirkpatrick [Kir83], consiste à partir d'une triangulation. Cette dernière peut être obtenue pour une carte planaire de taille n en temps $O(n \log n)$ suivant les algorithmes présentés au chapitre 3. La méthode de localisation de Kirkpatrick repose sur une représentation hiérarchique de la triangulation obtenue par

simplifications progressives. Pour passer d'une triangulation à une triangulation simplifiée on supprime un ensemble de sommets indépendants (deux à deux non-adjacents) et de degré borné puis on retriangule si nécessaire les "trous" laissés par les sommets supprimés. Pour que cet algorithme soit efficace on cherche à construire des ensembles de sommets indépendants, et de degrés bornés, les plus grands possibles. La borne sur les degrés assure que chaque triangle à un niveau de la hiérarchie intersecte un nombre borné de triangles du niveau suivant.

Dans ce qui suit le terme *triangulation* désigne une triangulation rectiligne du plan dont la face externe est elle-même un triangle (c'est donc combinatoirement une triangulation de la sphère). On peut toujours se ramener à une telle triangulation à partir d'une triangulation quelconque d'un domaine convexe en ajoutant trois sommets "à l'infini" et en joignant convenablement les sommets du bord du domaine à ces trois sommets. Les sommets d'une triangulation distincts des 3 sommets de la face externe sont dits *internes*.

Lemme 10.10 *Soit $\mathcal{T}$ une triangulation ayant n sommets internes et un entier $d \geq 6$. On peut sélectionner en temps $O(n)$ un nombre au moins αn de sommets internes, indépendants et de degrés au plus d avec $\alpha = \frac{d-5}{(d+1)(d-2)}$.*

Preuve : On considère l'algorithme glouton suivant : parcourir les sommets internes de $\mathcal{T}$ et sélectionner le sommet courant du parcours si son degré est au plus d et si aucun de ses voisins n'est sélectionné. L'algorithme glouton sélectionne clairement un ensemble de sommets internes, indépendants et de degrés au plus d en temps linéaire. Reste à minorer sa taille n_s.

Soit x et y les nombres de sommets internes de $\mathcal{T}$ respectivement de degré au plus d et au moins $d+1$. En particulier, $x+y=n$. On note a le nombre d'arêtes de $\mathcal{T}$. Par la relation d'Euler et par double comptage de la relation d'incidence arête/face, on obtient aisément $a = 3n+3$. De la relation d'incidence arête/sommet on déduit $2a = \sum_s \text{degré}(s)$, la somme portant sur tous les sommets de $\mathcal{T}$. En coupant cette somme en trois selon les sommets internes de degrés inférieurs ou supérieurs à d et les 3 sommets de la face externe, et en remarquant que tout sommet a degré 3 au moins, on obtient

$$2a \geq 3x + (d+1)y + 9$$

Ce qui, avec les relations précédentes implique

$$x \geq \frac{(d-5)n+3}{d-2} \geq \frac{(d-5)n}{d-2}$$

Mais l'algorithme glouton sélectionne un ensemble maximal de sommets. Dit autrement, l'ensemble des sommets sélectionnés et leurs voisins couvrent les sommets internes de degrés au plus d. On en déduit $(d+1)n_s \geq x$, soit encore

$$n_s \geq \frac{d-5}{(d+1)(d-2)} n$$

□

On construit une suite de triangulations $\mathcal{T} = \mathcal{T}_1, \ldots, \mathcal{T}_m$ ayant respectivement $n = n_1, \ldots, n_m$ sommets internes avec $n_m = 0$. On obtient $\mathcal{T}_{i+1}$ à partir de $\mathcal{T}_i$ en 2 étapes :

1. on sélectionne dans $\mathcal{T}_i$ un ensemble maximal de sommets internes, indépendants et de degré au plus d par l'algorithme glouton ci-dessus,
2. on supprime ces sommets de $\mathcal{T}_i$ et on retriangule l'étoile de ces sommets (de taille au plus d) en temps constant par étoile.

Le lemme précédent montre que $n_{i+1} \leq (1-\alpha)n_i$, d'où $m = O(\log n)$. On attache de plus à chaque triangle t de $\mathcal{T}_{i+1}$ au plus d pointeurs vers les triangles de $\mathcal{T}_i$ qui intersectent t. Clairement cette structure est de taille $O(\sum_i dn_i) = O(n)$ et peut être construite dans le même temps.

On détermine le triangle de $\mathcal{T}$ contenant un point p de requête par les étapes suivantes

1. déterminer si p est dans l'unique triangle de $\mathcal{T}_m$,
2. pour i allant de $m-1$ à 1, connaissant le triangle t de $\mathcal{T}_{i+1}$ contenant p, déduire celui de $\mathcal{T}_i$ contenant p en parcourant les triangles de $\mathcal{T}_i$ pointés par t.

Chacune des $m = O(\log n)$ étapes ci-dessus prend un temps constant puisque l'étoile à parcourir possède au plus d triangles.

Finalement,

Théorème 10.11 *Étant donné une triangulation $\mathcal{T}$ du plan, on peut construire en temps linéaire en la taille de $\mathcal{T}$ une structure de taille linéaire, permettant de localiser un point quelconque en temps logarithmique.*