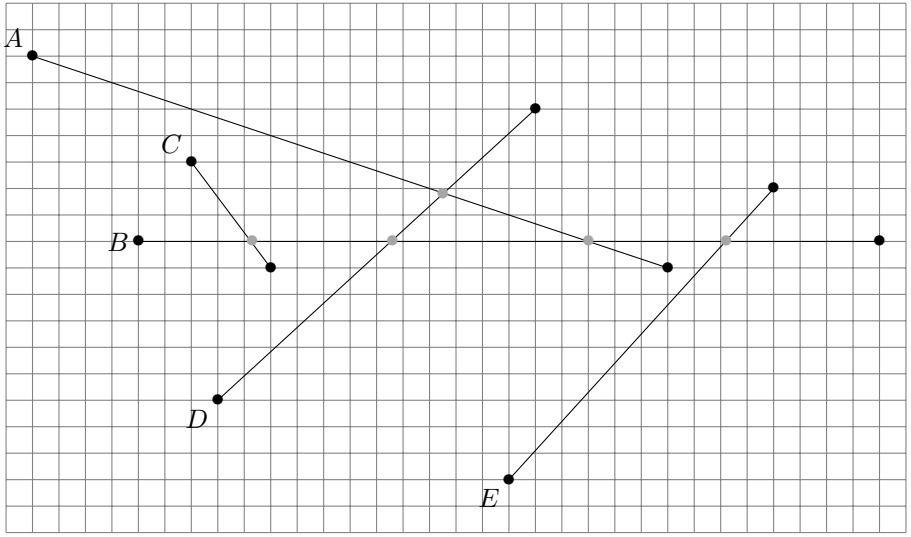
TD Infographie - Intersections

Dans tous les exercices, on se placera dans un repère orthonormé direct (O, i, j)

Exercice 1. [♡] Classification de Bentley-Ottmann

Cette méthode permet une classification efficace d'un ensemble de segments. Elle recherche dans une famille finie de segments, par balayage du plan, les paires qui se coupent.



1. Donner le contenu initial de la queue d'évènements Ω.
2. Pour chaque point évènement traité par l'algorithme, donner le contenu respectif de la queue d'évènements Ω et de la liste τ des segments intersectés par la ligne de balayage.

Algorithme ClassificationEnsembleSegments($lseg$)
données
$lseg$: liste de N segments
variables
Ω : liste de points (associés à un type D, F ou I) ▷ *triés par abs. croissantes*
τ : liste de liste d'identificateurs des segments ▷ *triés par ord. croissantes*

$\Omega \leftarrow$ les 2N sommets de $lseg$
$\tau \leftarrow$ liste vide
pt $\leftarrow$ 1er point de Ω

tantque (non fin(Ω)) **faire**
  cas (type(pt)) **de**
  `(D)EBUT` :
  insérer le segment de sommet pt dans τ (par ord. croissante)
    ▷ *par intersections des segments de τ avec droite x = pt.x*
  si (ce segment coupe ses voisins dans τ) **alors**
    insérer le(s) point(s) d'intersection dans Ω (par abs. croissantes)
  fsi

  `(F)IN` :
  si (les voisins de ce segment se coupent dans τ) **alors**
    insérer le point d'intersection dans Ω (par abs. croissantes)
  fsi
  enlever ce segment de τ

  `(I)NTERSECTION` :
  échanger les positions des segments concernés dans τ
  si (les segments nouvellement voisins se coupent dans τ) **alors**
    insérer les points d'intersection dans Ω (par abs. croissantes)
  fsi
  fcas

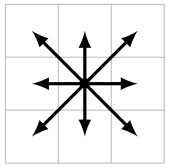
  pt $\leftarrow$ point suivant dans Ω
ftq

Exercice 2. [†] Remplissage récursif

On se propose d'écrire un algorithme équivalent à la « baguette magique » des logiciels de traitement d'images. L'outil « baguette magique » a pour but de sélectionner une région de pixels contigüs en cliquant sur un pixel de référence $p = (x, y)$ et en attribuant un seuil de luminosité $S \in [0, 255]$. Le principe de sélection est le suivant : on calcule la luminosité Y_p du pixel de référence p (avec un codage RVB sur 24 bits) :

$$Y := \lfloor 0.30R + 0.59V + 0.11B \rceil,$$

la valeur de chacune des trois composantes (R, V, B) appartenant à l'intervalle $[0, 255]$, et celle des huit pixels adjacents (le pixel p est au centre) :



On calcule alors la différence de luminosité entre p et ses huit voisins et on mémorise dans la région ceux des 8 pixels dont la luminosité est à une distance inférieure à S. La distance entre la luminosité Y_p et Y_q des pixels p et q est donnée simplement par :

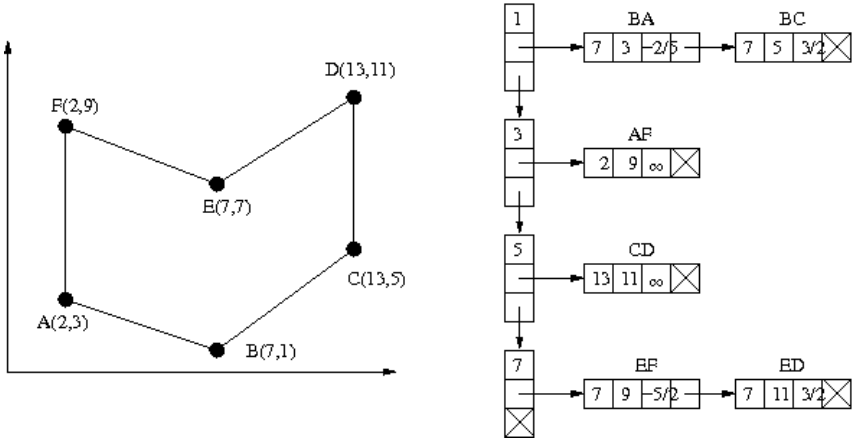
$$d(p, q) := |Y_p - Y_q|.$$

On recommence récursivement la même opération avec ceux des huit pixels qui auront été mémorisés, la valeur de luminosité de référence restant celle du pixel p initialement selectionné.

1. Décrivez un modèle de données adapté pour mémoriser la région, en supposant que l'image est un tableau bidimensionnel dont chaque terme est un enregistrement de trois champs (R, V, B) codé sur trois octets.
2. Écrivez l'algorithme $\mathsf{Y}(R, V, B)$ qui renvoie la luminosité d'un pixel à partir de son codage (R, V, B).
3. Écrivez l'algorithme récursif $\mathsf{BM}(p, S)$ qui renvoie la bonne région où $p = (x, y)$.

Exercice 3. [‡] Remplissage géométrique (*scanline*)

Pour remplir (par une couleur ou une texture) une figure géométrique de façon efficace, on s'appuie sur une structure de données adaptées : la table des arêtes actives (TAA). Au polygone défini par les sommets ABCDEF ci-dessous, correspond la table des arêtes (TA) à sa droite, sur laquelle s'appuie la mise à jour de la TAA au cours de l'algorithme dit *scanline*. Dans cet exercice, on écrira quelques algorithmes fondamentaux sur cette problématique.



1. Détailler les types de données `Sommet`, `Polygone`, `TA` (table des arêtes) et `TAA` (table des arêtes actives).
2. Ecrire l'algorithme de création de la TA. Vérifier son fonctionnement à partir de l'exemple ci-dessus. Rappels : les arêtes doivent être triées du bas vers le haut et de gauche à droite, on ne tient pas compte des arêtes horizontales, les pentes sont exprimées sous la forme d'un couple.
3. Ecrire l'algorithme principal *en faisant toute hypothèse utile quant à l'existence d'autres algorithmes* nécessaires à sa description (par exemple, le calcul de points d'intersection, la mise à jour de la TAA, ...).
4. Ecrire l'algorithme de mise à jour de la TAA (on supposera en antécédent de l'algorithme que la TA est bien constituée).