DM - Arbres et Graphes - Master 1 Bio-informatique

2025 - 2026

Rev. 1 du 21/01/2025

L'objectif de ce projet est de créer une structure de donnée qui permet de calculer rapidement les collisions entres des billes 2D positionnées à l'intérieur d'une salle rectangulaire.

On se place donc dans le plan. Dans ce plan, une salle est modélisée par un rectangle de largeur L et de hauteur H. Les billes sont modélisées par des cercles de rayons et de centre quelconques. La figure 1 montre un exemple de salle contenant des billes. Dans cette salle les billes C1 et C2 sont en collision car les cercles qui les représentent se touchent. Les autres billes ne sont pas en collision entre elles.

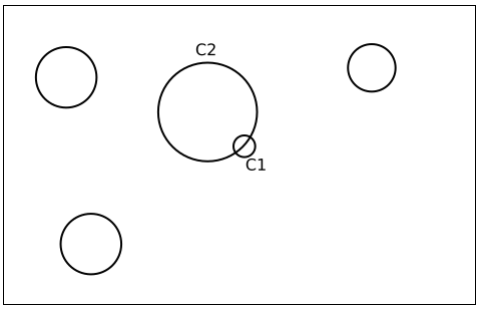

FIGURE 1 – Une salle contenant plusieurs billes. Les billes C1 et C2 sont en collision.

On souhaite construire une structure de donnée pour détecter rapidement si deux billes sont en collision.

Nous allons pour cela construire une représentation de la salle et des ses billes à l'aide de l'arbre binaire que nous allons définir maintenant.

Une boîte est un rectangle du plan. On appelle boîte horizontale (resp. verticale) une boîte qui, si elle est coupée, est coupée verticalement (resp. horizontalement) en deux boîtes verticales (resp. horizontales) de même tailles.

Par exemple, dans la figure 2, une boîte horizontale est coupée en deux, alors que dans la figure 3, une boîte verticale est coupée en deux.

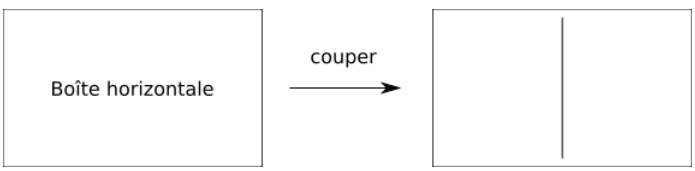

FIGURE 2 – Une boîte horizontale est coupée en deux

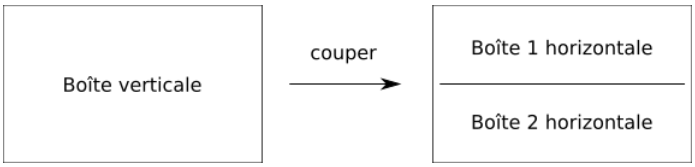

FIGURE 3 – Une boîte verticale est coupée en deux

Comme le montre les deux derniers exemples, un même rectangle peut être vu soit comme une boîte horizontale, soit comme une boîte verticale.

On décide maintenant de couper une salles en boîtes de la manière suivante.

Initialement, on place dans la salle, une boîte horizontale de la taille de la salle.

Ensuite, à chaque étape, on coupe toutes les boîtes présentent dans la salle.

Cette procédure est illustrée dans la figure 4 ou 3 étapes de découpes ont été réalisées.

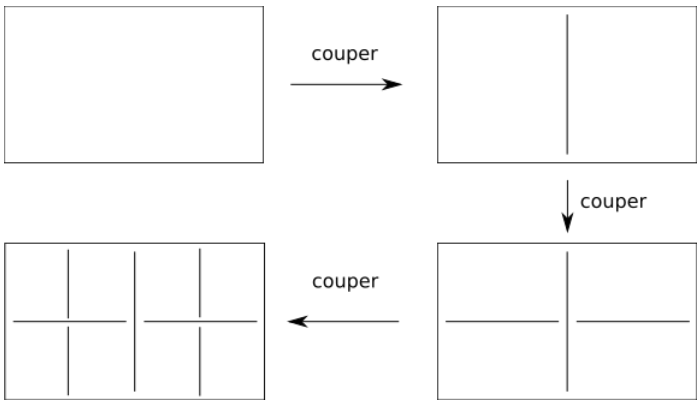

FIGURE 4 – Procédure à 3 étapes de découpe des boîtes d'une salle.

On cherche maintenant à modéliser par un arbre cette procédure. Vous pouvez déjà consulter le résultat de cette procédure en regardant les figures 5, 6 et 7.

Voici comment nous allons procéder. On construit l'arbre binaire de la façon suivante. Les boîtes représentent les noeuds de l'arbre binaire. La racine de l'arbre binaire est la première boîte de la procédure précédente, c'est à dire la boîte associée à la salle. Ensuite, si une boîte horizontale (resp. verticale) B est coupée en deux boîtes C_1 et C_2 alors C_1 et C_2 sont les deux fils de B. De plus, si C_1 est géométriquement situé à gauche (resp. en haut) de C_2 dans la salle, alors C_1 est le fils gauche et C_2 le fils droit.

La figure 5 montre la construction de l'arbre associé à une étape de découpe d'une boîte horizontale.

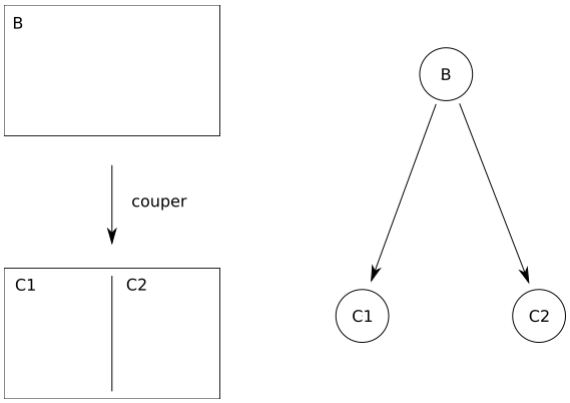

FIGURE 5 – L'arbre associé à la découpe d'une boîte horizontale

La figure 6 montre la construction de l'arbre associé à une étape de découpe d'une boîte verticale.

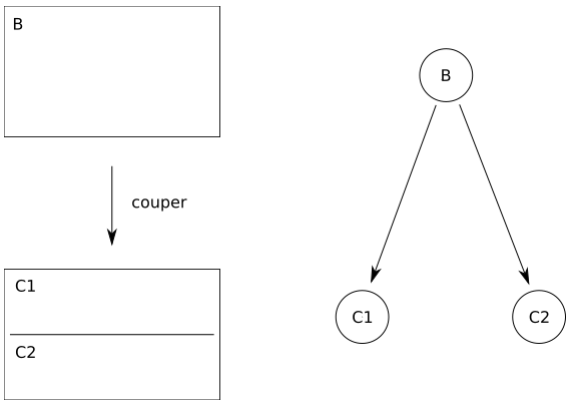

FIGURE 6 – L'arbre associé à la découpe d'une boîte verticale

Maintenant, si l'on applique une telle procédure, avec 3 étapes de découpe, on obtient les boîtes et l'arbre de la figure 7.

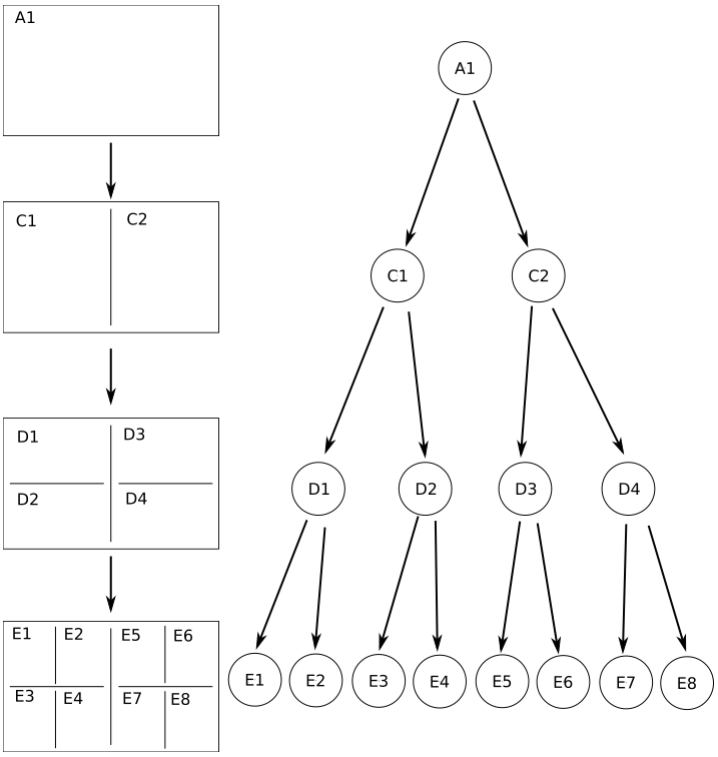

FIGURE 7 – L'arbre d'une procédure à 3 étapes de découpes.

Maintenant, nous allons ajouter à chaque sommet des étiquettes contenant les informations suivantes. Pour une boîte B, son étiquette est un dictionnaire contenant :

- la largeur de la boîte, de clef : "largeur" ;
- la hauteur de la boîte, de clef : "hauteur" ;
- l'orientation verticale ou horizontale, de clef : "orientation" ;
- une liste vide de billes, de clef : "billes" ;
- la position du coin Sud-Ouest de la boîte, de clef : "SO" ;
- la position du coin Nord-Est de la boîte, de clef : "NE".

Voici un exemple d'étiquette d'une boîte B d'un arbre A :

```
>> etiquette(A, B)
{
  'hauteur': 3, 'largeur': 1,
  'orientation': 'horizontale',
  'billes': [],
  'SO': (1, 0), 'NE': (2, 3)
}
```

1. Écrivez un programme `construire_arbre_collision(H,L,nb_etape)`, qui prends en paramètre la hauteur H et la largeur L de la salle, et nb_etape le nombre d'étapes du processus de coupure et qui renvoie l'arbre de collision, ne contenant aucune bille, présentée ci-dessus. Cet arbre doit être utilisable dans une biblothèque suivant l'API des arbre binaires suivants :

```
def filsGauche(A, p):
  # prends en paramètre un arbre A, un sommet p et renvoie un
  # sommet de A appelé fils gauche de p ou None si p n'a pas
  # de fils gauche.
def filsDroit(A, p):
  # prends en paramètre un arbre A, un sommet p et renvoie un
  # sommet de A appelé fils droit de p ou None si p n'a pas de
  # fils droit.
def pere(A,f)
  # prends en paramètre un arbre A, un sommet f et renvoie un
  # sommet de A ou None; ce sommet est appelé le père de f.
def racine(A) :
  # prends en paramètre un arbre A et renvoie un sommet A
  # appelé racine de l'arbre.
```

Nous allons maintenant coder les billes par un tuple dont la première valeur représente le centre du cercle et la deuxième valeur, son rayon. Par exemple, une bille de rayon 3 et de centre (2,5) est codée par : $((2,5),3)$.

On place maintenant le cercle dans la salle en l'insérant dans la plus petite boîte dans laquelle la bille entre entièrement.

La figure 8 montre un exemple de salles remplies de différentes billes. Dans ce graphe, par soucis de clarté, nous avons notés les billes par B_1, B_2, ..., B_7 à la place des tuples contenant centre et rayon de ces même billes. Nous n'avons pas aussi écrit le contenu de toutes les étiquettes de l'arbre.

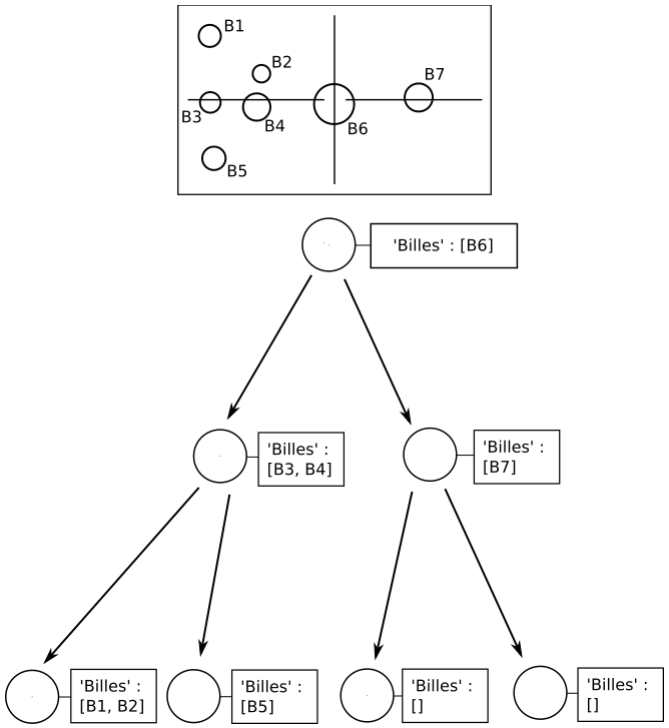

FIGURE 8 – Un exemple de salle remplies des billes B_1, B_2, $\cdots$, $B7$. En haut, le dessin de la salle et de ses billes. En bas, la structure arborescente qui lui est associée.

Nous allons utiliser cette structure arborescente pour manipuler et placer rapidement des billes dans la salle.

En vous inspirant des programmes de recherche dichotomique,

2. Écrivez un programme `placer_bille(salle, position, rayon)`, qui prend en paramètres une salle codé sous forme d'arbre, la position (x,y) du centre de la bille à ajouter et son rayon et qui ajoute la bille dans l'arbre à la position indiquée. En utilisant un générateur de nombre aléatoire, générez une grande salle avec plusieurs milliers de billes de toutes tailles.
3. Écrivez un programme `lister_bille(salle, position)`, qui renvoie la liste des billes de la salle qui contient un point situé à la position "position".
4. Écrivez un programme `lister_collisions_bille(salle, bille)`, qui prend en paramètre une salle et une bille et qui renvoie la liste des billes de la salle qui sont en collision avec la bille.
5. Écrivez un programme `liste_de_toutes_les_collisions(salle)` qui renvoie une liste de paires de billes qui sont actuellement en collision dans la salle. Appliquez votre programme avec l'exemple ci-dessus. Que se passe-t-il si vous utilisez à la place un programme simple qui passe en revue toutes les paires de billes et les comparent deux à deux ? Testez et comparez les résultats.