Mini-projet – Géométrie Algorithmique

Réalisé par

KALAI Mohamed Hedi

SATOURI Khaireddine

L3 Informatique

Année Universitaire : 2021-2022

KALAI Mohamed Hedi – SATOURI Khaireddine
L3 Informatique

1. Introduction

L'idée est de créer un ensemble aléatoire de point et les stockés dans un tableau dynamique. Les rectangles sont représentés sous forme d'objet instanciées d'une classe **Rectangle** qui prend comme attributs des points du tableau des points et une largeur et hauteur aléatoire pour chaque rectangle.

L'ensemble de rectangles sont stockés dans un tableau dynamique et toutes les opérations à faire vont être basés sur ce tableau (accès aux points de chaque rectangle, tri, recherche de collisions…)

2. Approche Brute Force

1. Principe

Pour cette approche il faut juste penser sur la résolution de problème sans prendre en considération la complexité. Pour avoir tous les rectangles qui sont en collisions il suffit de parcourir l'ensemble de rectangle et chaque fois qu'on se pointe sur l'un d'eux, en boucle les autres pour vérifier s'il y'à des collisions.

Pour vérifier s'il y'a de collision entre 2 rectangles en compare les valeurs des points qui le constituent avec les autres points du rectangle ; le fait d'avoir une valeur supérieure à l'autre point de l'autre rectangle la plus proche en considère que les 2 rectangles sont en collisions (comparaison abscisse/abscisse ou ordonné/ordonné)

Les rectangles en collisions sont affichés en rouge.

2. Complexité

Dans notre algorithme, on a 2 boucles « for » l'une à l'intérieure de l'autre et après vérifications, nous avons réalisé que cela n'a pas fait pas de différence donc la complexité est de **O(n^2).**

3. Approche Sweep-Line

1. Principe

Cet algorithme nécessite une liste ordonnée d'éléments pour le respecter. L'idée est de commencer à trier l'ensemble de rectangles en utilisant une classe qui implémente la class générique Comparator (méthode du TP) responsable de tri des éléments.

Une fois on a la liste triée on applique le même principe que la BruteForce mais cette fois en bouclant à l'intérieure on a pas besoin de parcourir tous les rectangles de la listes une autre fois, puisque les rectangles sont triés on cherche les collisions à partir des prochains rectangles qui poursuit le rectangle actuel « i » et nous pouvons interrompre la boucle dès que l'abscisse x du premier point (point gauche en haut) du prochain rectangle que nous allons tester est plus grand que l'abscisse x du deuxième point(point droite en haut) de notre rectangle actuel « i » et il y'a une collision par rapport l'axe des ordonnées.

2. Complexité

L'algorithme nécessite un tri, donc on suppose que l'algorithme de tri prend **O(nLogn).** L'algorithme a 2 boucles « for » complexité de **O(n^2)** mais puisque chaque rectangle sera testé avec quelques rectangles qui le suivent, ça qui rend approximativement la complexité très proche à **O(n)**, donc en conclusion **O(nLogn)** en général incluant le tri.

4. Approche diviser pour régner

1. Principe

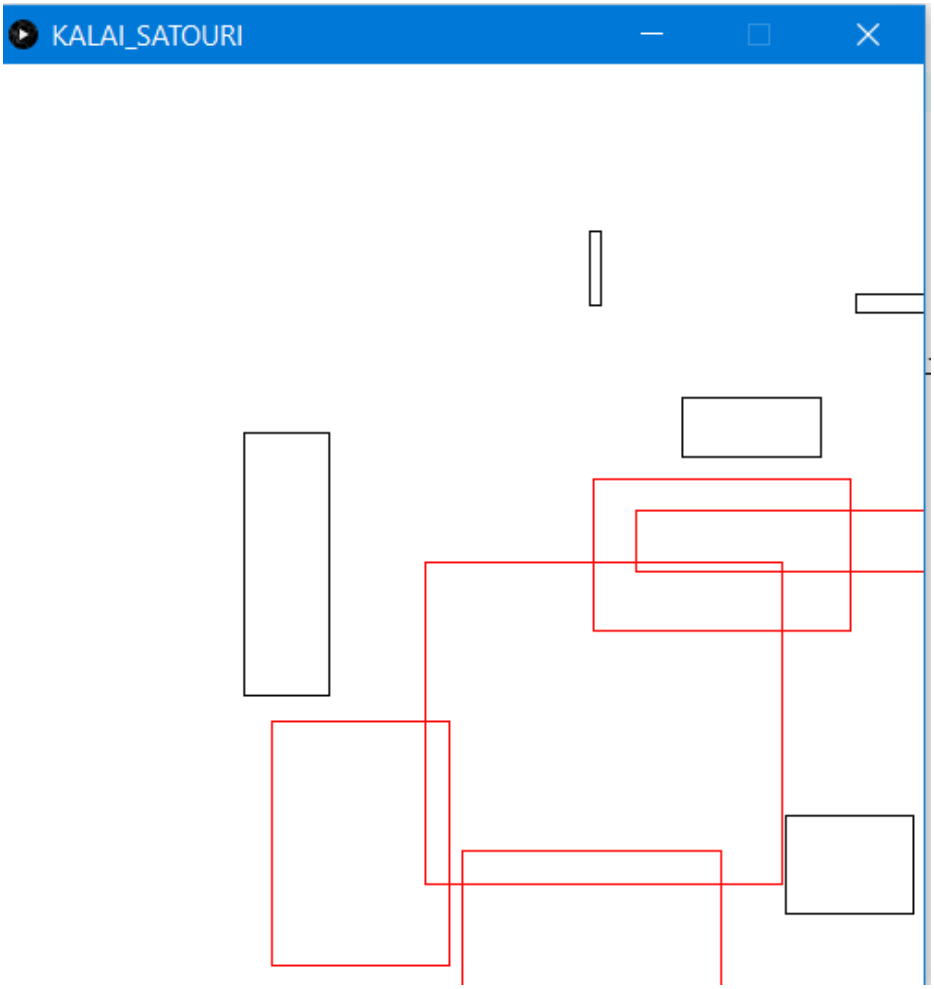
Cet algorithme nécessite un tri aussi. Une fois on a la liste triée on applique le principe de diviser pour régner, on fait des appels récursifs de la fonction pour découper la liste des rectangles jusqu'à nous avons suffisamment des rectangles puis on applique le même principe du brute force.

2. Complexité

L'algorithme nécessite un tri, donc on suppose que l'algorithme de tri prend **O(nLogn).** En conclusion l'algorithme prend **O(nLogn).**

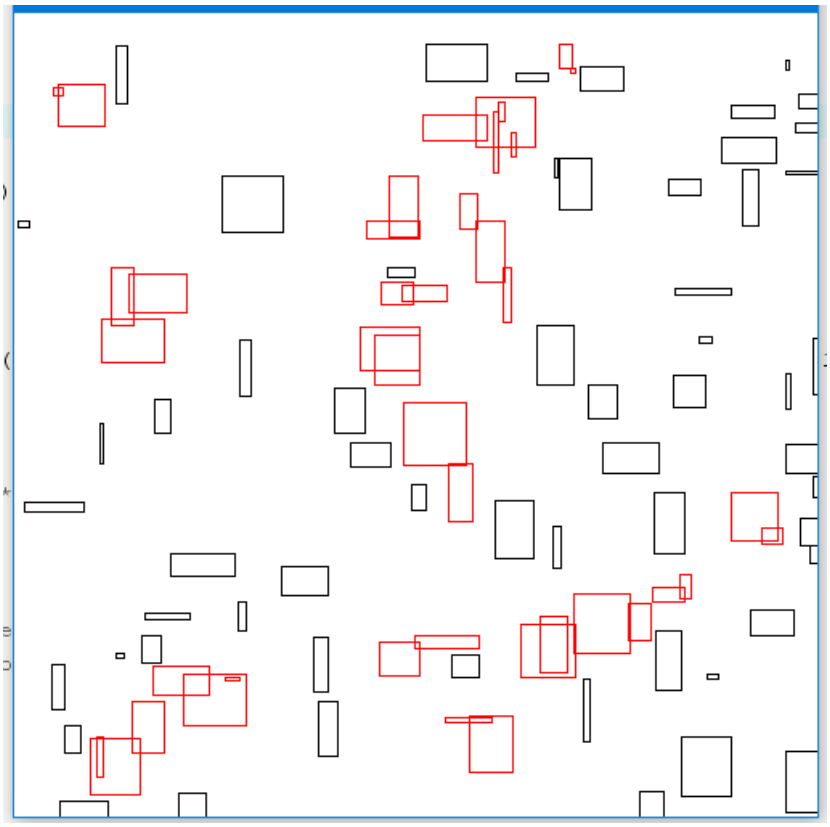
5. Captures d'écrans

N =10 Rectangles avec taille moyenne = 50

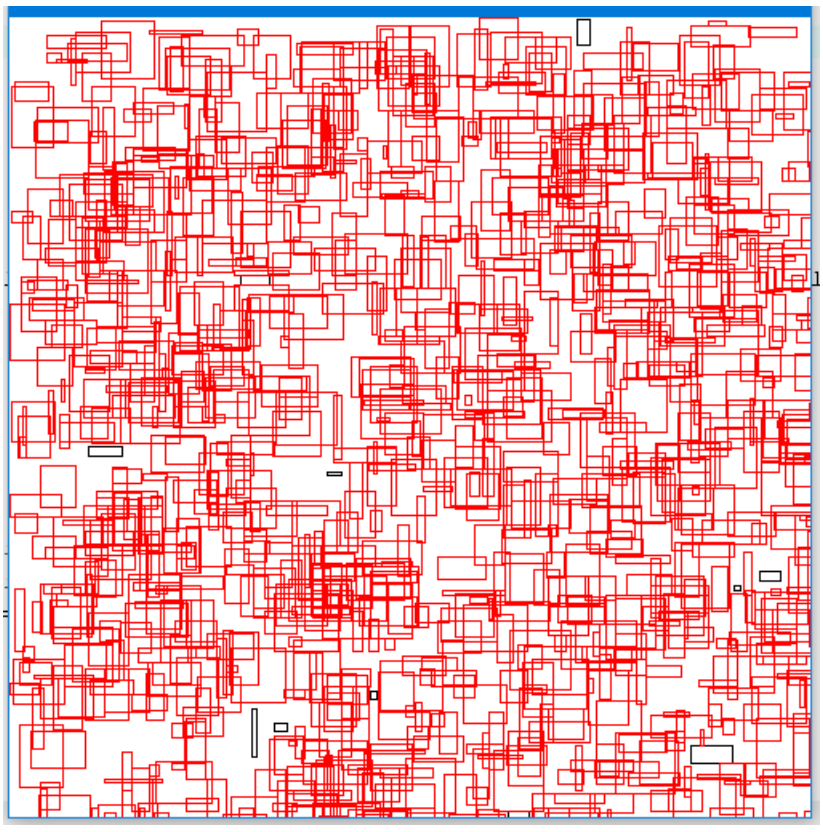


KALAI Mohamed Hedi – SATOURI Khaireddine
L3 Informatique

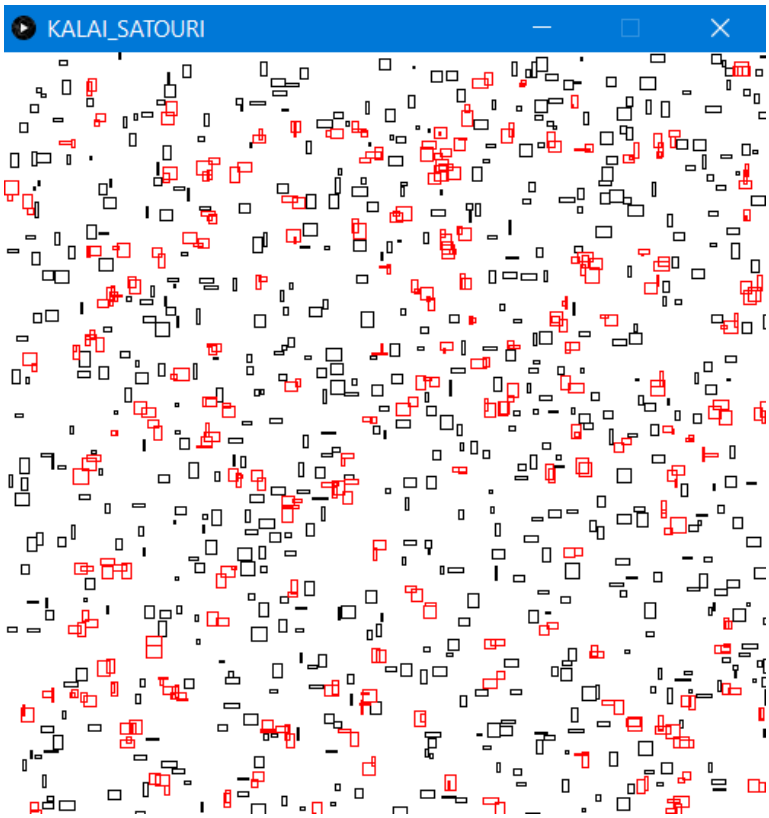
N =100 Rectangles avec taille moyenne = 20



N = 1000 Rectangles avec taille moyenne = 20



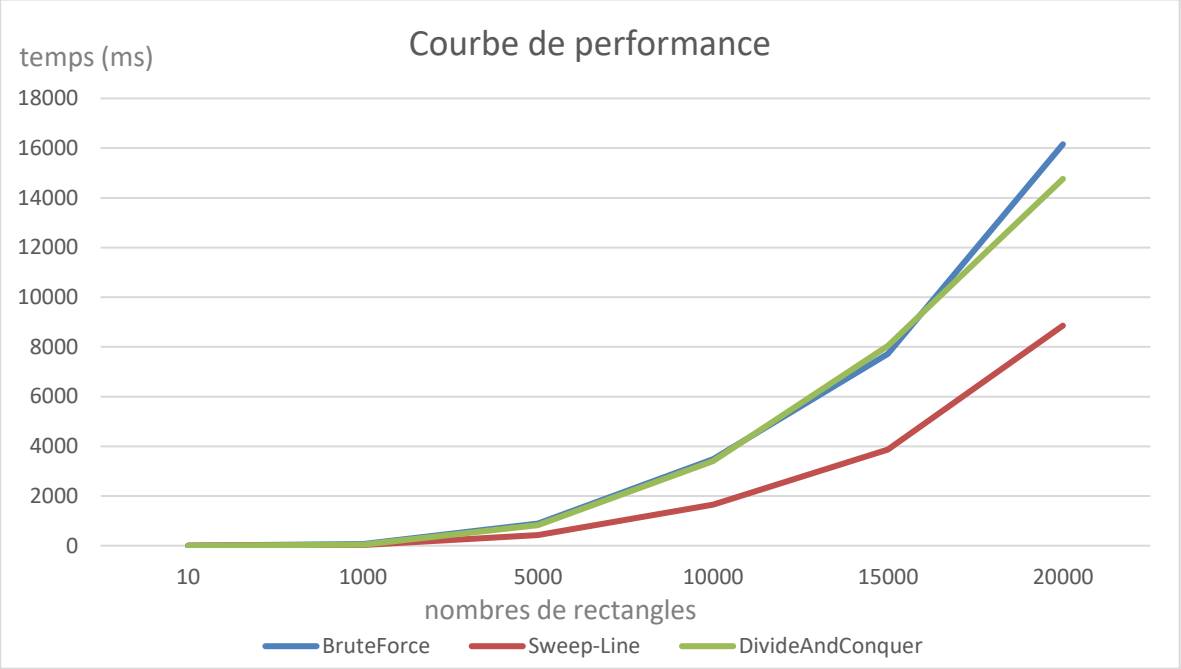
N =1000 Rectangles avec taille moyenne = 10



6. Résultats obtenus

	A	B	C	D	E
1		BruteForce	Sweep-Line	DivideAndConquer	
2	10	3	1	1	
3	1000	65	28	42	
4	5000	896	431	827	
5	10000	3481	1653	3407	
6	15000	7725	3868	8044	
7	20000	16158	8858	14763	

7. Courbe de performances



8. Conclusion

Le projet nous a pris 15h en total et semblé moyenne coté difficulté.

La partie la plus difficile est celle pour trouver la solution de diviser pour régner. Aussi pour débuter théoriquement les algorithmes et les structures utilisés.

Nous avons travaillé ensemble depuis le début dans le même local et nous avons chercher les solutions ensemble au fur et à mesure.

Merci de votre attention !