Rapport de projet de développement logiciel

KICHOU Yanis 3703169 & ALLOUCHE Yacine 3701222

14 Mai 2018

SOMMAIRE

Présentation

Le Robot Trieur :

Le principe est le suivant : une grille de dimension (n,m) et un nombre de couleurs est proposer par l'utilisateur en lançant jeu.

Le programme du jeu crée la grille de tels sort que chaque case de la grille contient une pièce avec un couleur et une couleur de fond de case , le robot est placé par défaut à la case (0,0) du jeu

Le but c'est d'utiliser le robot pour déplacer les pièce de tel sort que la couleur d'une pièce correspond à celle du fond de la case et automatiquement la case devient noir

Une fois que les (n*m) case de la grille sont noir le jeu est fini.

Les critères à prendre en considération durant notre Projet sont :

- Le nombre de pas que le robot effectue
- Le temps utilisé par le robot à fin de trouvé la solution
- Et vérifie si la solution est optimale

Problématique

En essayant le jeu plusieurs fois on s'est rendu compte qu'a chaque fois qu'un utilisateur joue au jeu rare ceux qui ont trouver la solution avec le même nombre de pas et avec la même durée , d'où on s'est posé la question comment trouvé la meilleur solution pour le jeu tel qu'elle sois minimal en nombre de pas et en temps d'exécution ? et qu'elle structure de données est plus adéquate à utiliser pour exécuter l'algorithme qui trouve la solution ?

Les proposition sont :

- Les Structures de dionées de type Matrice
- Les Structures de données de type Liste
- Les Structures de données de type Arbres
- Les Structures de données de type Graphes

Objectifs

Utilisé les différentes Structure de données et déterminé celle qui résout l'algorithme avec un temps d'exécution et une complexité minimale.

Solutions

1. Structure de donnés de type Matrice :

a) *Algorithme natif (Exercice 1) :*

Définition :

L'algorithme naïf consiste à créer une solutions qui sois la plus naïf possible , le principe est le suivant :

A Partir de la Case courant du robot chercher par ligne la case qui correspond à la couleur de la pièce et la déplacer jusqu'à cette case tout en choisissant la plus proche qui sois.

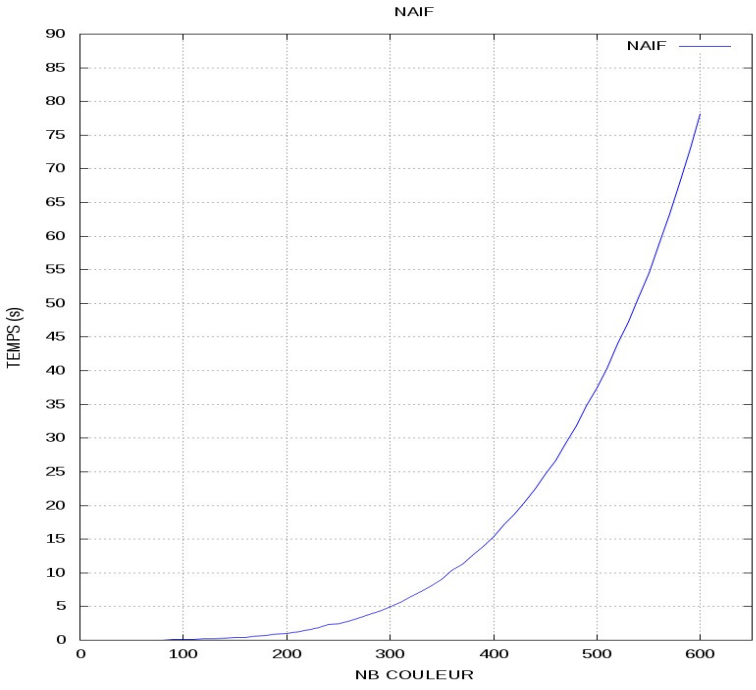
Cette algorithme propose une solution meilleur que celle de nos utilisateurs mais certes en ce qui concerne temps d'exécutions il est très lent

Et utilisable qu'avec des grilles de petite ou moyenne taille car la recherche se faire par ligne alors si la ligne est de taille n alors la recherche se ferra en O(n).

D'où sois la grille à dimension n m alors cette algorithme utilise une complexité en O(n*m) .

Graphe :

On exécutant le programme plusieurs fois avec des valeurs croissantes on a obtenu les résultats figurant les fichier .dat et ainsi on a pu dessiner la courbe suivant



Cette courbes représente le temps d'exécution du programme en fonction de la taille de la grille (n*m) tels que pendant nos tests on a considéré que la grille soit carré pour une meilleur précision.

Comme le montre le graphe on déduit que la fonction tracé par la

courbe est hyperbolique : $f'(x) = ax^1$ Qui prouve que la complexité de l'algorithme est en O(n*m) vu

<u>TEST :</u>

Pour :

n=10 m=10 nombre de couleur =10 graine =10 ➔ 0.000243 s

n=100 m=100 nombre de couleur =100 graine =10 ➔ 0.095 s

n=1 m=100 nombre de couleur =1000 graine =10 ➔ 0.034s

b) *Algorithme CIRCULAIRE (Exercice 2) :*

Définition :

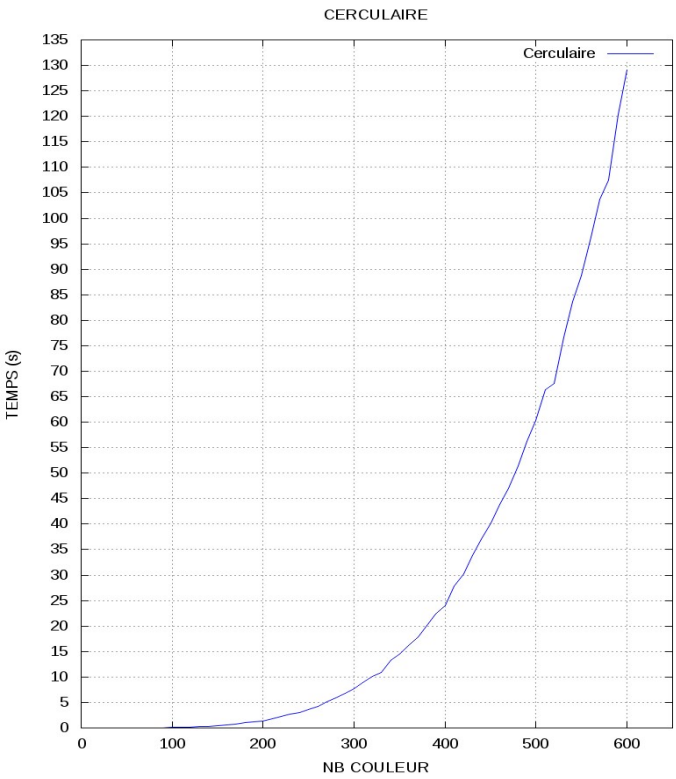
Cette Algorithme ajoute une petite touche d'intelligence à l'algorithme naïf , quelques améliorations qui permet de gagné en temps d'exécution .

L'algorithme Circulaire cherche la case la plus proche du robot et qui à la même couleur de fond que sa pièce, et pour cela il cherche dans les 4 direction vers le Haut ,le Bas , à Droite et à gauche et renvoie la première case rencontré ,et en cas ou y a deux case qui ont la même distance il renvoie la plus haute à gauche.

Cette algorithme permet de gagner du temps d'exécutions de la recherche car l'algorithme naïf parcoure toute la grille pour trouvé la case la plus proche.

Graphe :

On utilisant un script qui exécute le programme avec l'algorithme Circulaire et enregistre dans un fichier .dat on a pu obtenir ce graphe qui représente les différents temps d'exécution en fonction du nombre de couleur



Remarque :

Pour avoir des résultat précis on a utilisé des matrices carrer.

On note que le graphe ne différé pas trop de celui du naïf mais juste on gagne quelque temps d'exécutions

D'où on déduit la dérivé : $h'(x)=bx$
Tel que a > b

Si l'on trace les deux courbes dans un seul graphe on notre que la courbe de L'algorithme naïf est en dessous de celle du Circulaire. Mais certes en ce qui concerne la complexité au pire des cas on note que reste toujours en O(n*m)

TEST :

N=10 m=10 nbCouleur=10 graine=10 ➔0.00225

N=100 m=100 nbCouleur=100 graine=0 ➔0.128 s

N=100 m=100 nbCouleur=1000 graine=5 ➔ 0.224s

N=50 m=50 nbCouleur=2500 graine=5 ➔ 0.0216

N=200 m=200 nbCouleur=10 graine=0 ➔ 1.03s

II. Structure de Liste Doublement chaînées :

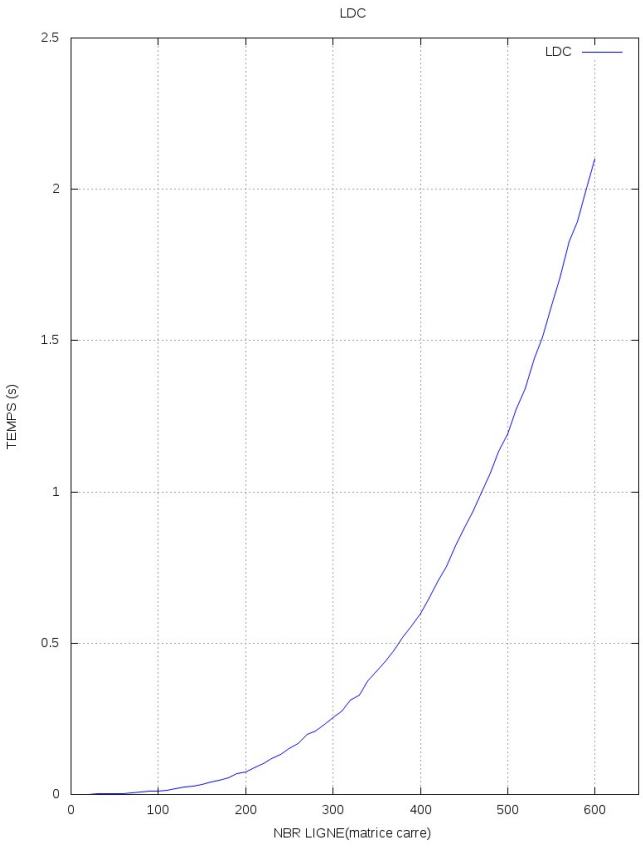
Définition :

Cette Algorithme utilise une structue de données de type liste doublement chaînées cette structure est très coûteuse en mémoire mais elle permet d'améliorer la complexité de cette algorithme et ainsi accéléré la rapidité d'exécution du programme.

Sa complexité est en o(n*m) tel que n est la longueur d'un file et cela réduit le coût de la recherche de la case la plus proche correspondante à la pièce du robot

Graphe :

En utilisant le même principe vu précédemment dans la conception des courbes précédentes à l'aide d'un script qui le le programme qui utilise l'algorithme qui se base sur la structure de Liste Doublement chaînées on a pu dessiné cette courbe qui représente les temps d'exécution en fonction nombre de cases



Par défaut on utilise toujours des matrice

On note que la fonctions est plus rapide que celle des deux algorithme vue précédemment (Naïf et Circulaire) et on peut la modéliser par la fonction {

$ldc'(x) = lx$ Ou l < b < a }

Remarque :

On a pu déduire que cette algorithme est au maximum de sa performance si le nombre de case correspond au nombre de case tel que la

recherche de case n’ai pas d’autre qu’un accès direct a la file qui contient qu’un seul élément.

TEST :

N=100 m=100 nbCouleur=20 graine=0 ➔ 0.0215 s

N=250 m=255 nbCouleur=15 graine=0 ➔ 0.596 s

N=500 m=500 nbCouleur=25000 graine=0 ➔2.263 s

N=500 m=500 nbCouleur=10 graine=10 ➔11.747 s

2- Structure de Graphe :

A. *Structure d'arbre (AVL) :*

Définition :

un arbre est une structure de données récursive générale, représentant un arbre au sens mathématique. C'est un cas particulier de graphe qui n'a qu'une seule source et aucun cycle

Un Arbre AVL (Adelson-Velskii et Landis) : est un arbre binaire de fouille équilibré. Ainsi, pour chaque nœud de l'arbre, la différence de hauteur de ses sous-arbres est au maximum un.

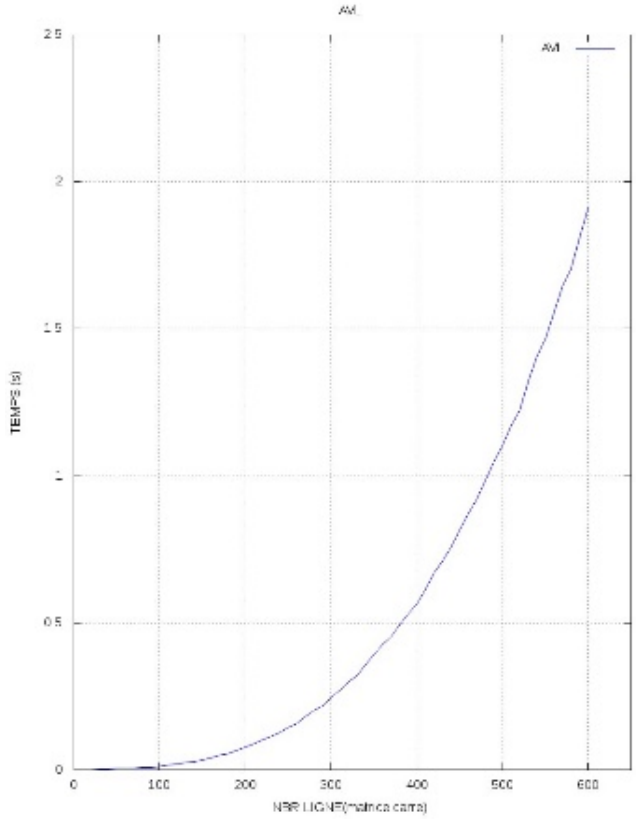
L'équilibrage de l'arbre est le point fort de cette algorithme , car on utilisant pour la rechercher un élément , il utilise la recherche dichotomique ce qui permet de réduire la complexité de recherche sur une ligne en O(log (n)) et d'avoir une complexité générale en O(m * (log (n)) tel que n est le nombre de colonne et m est le nombre de ligne.

GRAPHE :

Cette courbe représente les temps d'exécution en fonction du nombre de cases

D'après cette courbe on a déduit qu'elle représente un fonction polynomiale.

On annoté que le temps d'exécution de l'algorithme utilisant un



AVL est plus rapide que l'algorithme Naïf Circulaire et LDC certe si on peut modélisé les fonctions on pourra écrire :

$AVL'(x) = vx$ | v< l < b < a

Test :

N=10 m=10 nbCouleur=10 graine=10 ➔0.00038 s

N=100 m=100 nbCouleur=100 graine=10 ➔0.0265 s

N=350 m=400 nbCouleur=100 graine=10 ➔0.996 s

N=500 m=500 nbCouleur=250000 graine=10 ➔ 15.94 s

N=500 m=500 nbCouleur=10 graine=0 ➔ 2.89 s

2-Structure de données de types GRAPHES :

Définition :

Dans cette structure on utilise le principe des graphes , ou dans chaque sommet on enregistre sa liste d'adjacence qui sont les coordonnées des cases ayant la même couleurs de fond.

Cette algorithme utilise une autre structure comme les fils afin de pouvoir parcourir les différents graphes et la recherche de sommet.

Le principe de cette algorithme c'est :

A partir d'un sommet donné on parcoure sa liste d'adjacence et on extrait tout les cycles existant.

et à partir de cette liste on utilise deux type d'algorithme

1- l'algorithme Daniel :

Qui permet d'avoir la meilleur exécution et qui minimise le nombre de pas produit par le robot, mais certe il est utilisé dans le cas ou le nombre de ligne égale a 1 (vecteur) en parcourant les cycle trier par rapport au coordonnée des colonnes et donne une solution qui combine le parcoure de chaque cycle afin d'avoir comme un cycle générale .

2- L'algorithme Daniel Modifier :

Une petite amélioration de l'algorithme Daniel a fin de pouvoir L'utiliser dans les cas générale ou le nombre de ligne n'ai pas limité a 1.

6-Conclusion :

Durant ce Projet on a pu améliorer nos connaissances des différentes structures de données et comment les exploité au mieux a fin de résoudre un problème complexe avec la prise en charge de la complexité des différents algorithme et leurs temps d’exécution.