ALGO 01 : PROJET

Maxence Delacroix & Rémi Sezille

PART 1 :

L'algorithme que nous utilisons est très simple. On commence par calculer la position moyenne des maisons, ensuite, on calcule l'écart-type (sigma) de la distribution des maisons. Une fois cela fait, on sépare la liste des maisons en 4 listes différentes.

-positive MajorLlist : elle contient toutes les maisons positionnées entre notre point de départ et la valeur (moyenne + 0.96*sigma) définissant la limite haute où se situent le plus de maisons.

-negativeMajorLlist : elle contient toutes les maisons positionnées entre notre point de départ et la valeur (moyenne - 0.96*sigma) définissant la limite basse où se situent le plus de maisons.

-positiveRemainingLlist : elle contient les valeurs restantes positives

-negativeRemainingLlist : elle contient les valeurs restantes positives

On choisit ensuite en se fiant à la longueur de chaque **MajorLList** ou de la valeur la plus lointaine. On parcourt toujours les listes **MajorLlist** avant les **RemainingList**. Cette opération permet de définir dans quel ordre les listes seront parcourues.

De cette façon, on ne perd pas de temps à changer de direction pendant le passage du chasse-neige. Faire des demi-tours prend beaucoup de temps, c'est pourquoi nous allons systématiquement dans la même direction, tant qu'on ne s'éloigne pas trop des autres maisons (c'est la raison pour laquelle on calcule la moyenne et l'écart-type).

Cet algorithme n'a pas de boucles imbriquées et les boucles ne dépendent que de la taille des listes.

Il est composé de 5 boucles `for` et a un temps d'exécution de 5n

Il utilise 3 fois la fonction `sort()` de python qui a une complexité `O(n log(n))` (selon wikipédia, python utilise Timsort)

L'algorithme a donc un temps d'exécution de 5n + 3(log(n))

En simplifiant, l'algorithme a une complexité de O(n log(n)), donc, c'est un algorithme quasilinéaire, ce qui est mieux que polynomial.

Part 2

Option 1 :

Implémentation d'une heuristique pour le problème de coloration de graphes

Algorithme de Welsh-Powell :

L'algorithme se présente de la façon suivante :

- Attribuer un « poids » à chaque nœud en fonction du nombre de voisin
- Lister tous les nœuds selon leur poids dans l'ordre décroissant

- Colorer le nœuds le plus lourd, puis, suivant l'ordre décroissant, tous les points n'étant pas voisin de ce nœud (ni entre eux) d'une couleur
- Reprendre le nœud le plus lourd puis recommencer l'étape précédente avec une autre couleur, jusqu'à ce que tous les nœuds soient colorés

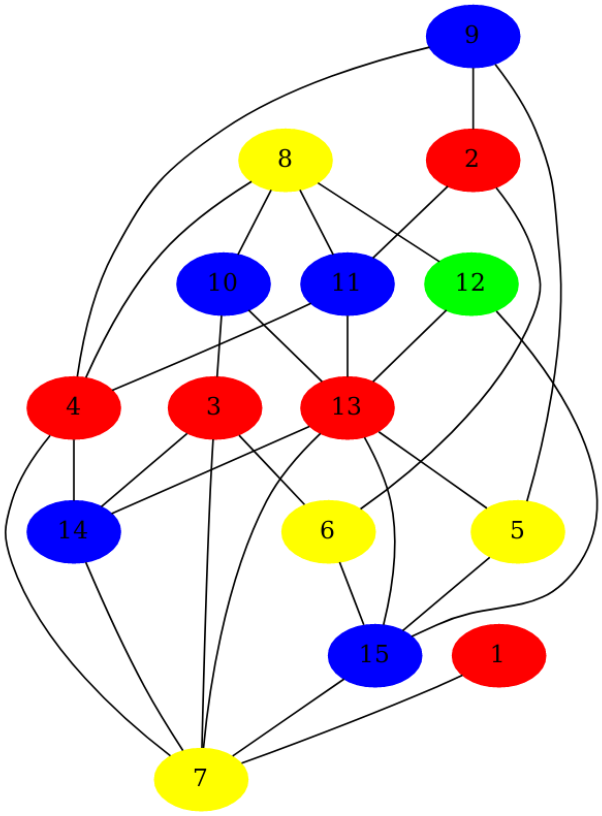

Figure 1Graphe comptant 15 nœuds et jusqu'à 30 arrêtes

Ici l'algorithme ne donne pas nécessairement une solution optimale (aussi appelé « nombre chromatique ») mais le résultat est nettement meilleur qu'une solution gloutonne ou pseudo-aléatoire.

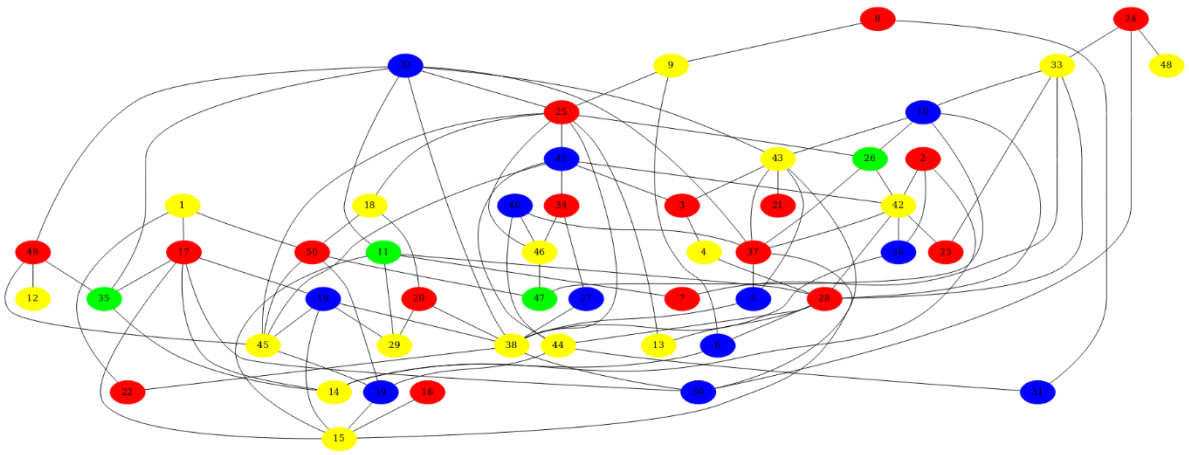

Figure 2Graphe comptant 50 nœuds et jusqu'à 100 arrêtes résolu avec 4 couleurs

Très souvent, ce problème peut être résolu en utilisant quatre couleurs ou moins.

Parfois, l'algorithme de Welsh-Powell utilise cependant 5 couleurs sur les certains graphes.

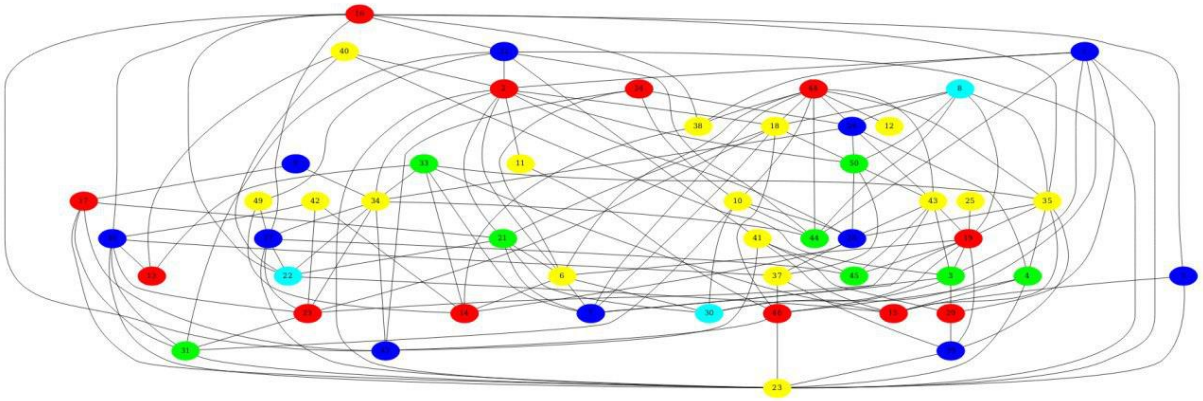

Figure 3Graphe comportant 50 nœuds et jusqu'à 150 arrêtes, résolu avec 5 couleurs