Rapport projet programmation parallèle

Sommaire:

- Manuel d'utilisation du programme.
- Explications des différentes solutions implémentées (Le séquentielle, Le parallèle).
- Les résultats et temps d'exécution des différents plateaux du jeu.
- Interprétation.

1. Manuel d'utilisation:

Pour faire fonctionner mon programme et voir l'affichage d'une solution du jeu si elle existe, il suffit de donner en entrée un fichier qui décrit le plateau du jeu.

Ensuite si on utilise la solution parallèle on a la main pour modifier le nbr de thread pour trouver une solution.

2. Explication de la solution :

Je commence par expliquer comment j'ai implémenté le backtracking dans mon programme. Sur mon code on peut voir 3 grandes fonctions :

- estValideCouleurBord.
- estValideCouleurAdj.
- resoudreJeu.

Les deux premières fonctions vont nous permettre de vérifier les contraintes imposées par le jeu à savoir que les couleurs de bordure de notre plateau doivent être identiques et que chaque partie du carré doit avoir un coté adjacent d'un autre carré de même couleur. Et puis la dernière fonction c'est là ou le backtracking est vraiment implémenté, Dans cette fonction :

- -On parcours nos carrés du jeu possible et on place sur une case données un carré du jeu qui vérifie les conditions du jeu.
- -On ressort ce carré du vecteur et on le met sur une pile pour ne pas perdre la donnée dans le cas où cette route ne mène pas à une solution.

- -On continue à avancer, si on se trouve dans une situation de blocage (Aucun carré restant dans le vecteur ne peut être valide et mis sur le plateau) on fait un pas en arrière (On retire le carré du plateau et on le transféré de la pile au vecteur des carrés possibles).
- -Si on arrive remplit plateau, alors on a trouvé une solution et on l'affiche.
- -Si on essai toute les routes possibles sans trouver une solution, Alors on affiche « pas de solution possible pour ce jeu ».

Solution Séquentielle :

Dans cette solution tout ce fait simplement, on fait appel à la fonction resoudreJeu qui utilise a son tour les autres fonction, Si on cette fonction à une valeur de retour TRUE, alors on affiche notre plateau, Sinon on affiche un message indiquant que pas de solution trouvée.

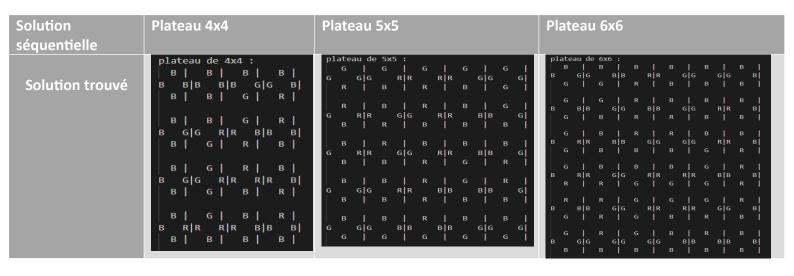
Solution parallèle :

Alors pour cette solution puisque la résolution du plateau commence toujours par la première case et puisque le parcours du vecteur se fait par ordre croissant, J'ai pensé à créer plusieurs threads qui commence par la première case du plateau mais avec un vecteur mélangé, Ceci implique que chaque thread ne va pas avoir le même premier carré qui satisfait les contraintes pour la première case, Donc chaque thread va partir sur une route différente que les autres threads. Chaque thread va essayer de résoudre le jeu, Le premier thread qui trouve une solution va informer le Main thread qui va demander aux autres threads de s'arrêter et après affiche la solution trouvée.

3. Les résultats des plateaux et temps d'exécution :

Pour le temps d'exécution je vais afficher les temps trouvé sur deux machines différentes parceque je pense que mon ordinateur portable ne donne pas de temps optimal pour les plateaux 4x4 et 5x5.

Pour la solution parallèle je vais afficher le temps d'exécution en agissant sur le nombre de thread



Temps d'exécution sur deux PCs	[Done] exited with code=0 in 1.547 seconds Process exited after 0.08561 seconds	exited with code=0 in 2.053 seconds Process exited after 1.089 seconds	exited with code=0 in 8.861 seconds Process exited after 12.27 seconds
Solution Parallèle			
Temps d'exécution avec nbr thread =8	exited with code=0 in 1.632 seconds	exited with code=0 in 1.71 seconds	exited with code=0 in 21.254 seconds
Temps d'exécution avec nbr thread =16	[Done] exited with code=0 in 1.716 seconds	[Done] exited with code=0 in 1.77 seconds	exited with code=0 in 11.334 seconds

4. Interprétation:

On remarque que les temps d'exécution varient selon la solution parallèle ou séquentielle. Les dimensions du plateau, ainsi que pour la solution parallèle le nombre de thread entre aussi en jeu.

Pour la solution parallèle on distingue que ça ne diffère pas trop avec la solution séquentielle pour les plateaux 4x4 et 5x5. Mais on remarque que pour le plateau 6x6 la solution parallèle présente un temps minimal par rapport aux autres dans le cas ou on a 16 threads pour traiter la solution. Cela montre que le nombre de thread entre en jeu lorsqu'on agrandie notre plateau. De plus, le temps varie d'une exécution à une autre parceque l'algorithme se base sur une fonction qui mélange le vecteur donc cela entre en jeu.

Sur ce projet ce qui a bloqué c'est la solution avec un thread pool je ne trouve pas d'idée pour implémenter le parallélisme sur ce programme avec un thread pool.