Algorithmes de coloriage de graphes

(sujet proposé par M. Szusterman - maud.szusterman@gmail.com)

Les trois sujets proposés font intervenir des algorithmes classiques de théorie des graphes, qu'il s'agit d'animer dans un contexte concret qu'il vous faudra coder. Le premier sujet est le grand classique de la carte de géographie à colorier, le second est aussi une affaire de couleurs (coloration des aretes, cette fois). Le troisième sujet est une variante du problème Vertex Cover, souvent appelée Matrix Domination.

Selon le sujet choisi, des références seront fournies aux étudiants pour les algorithmes concernés.

0.1 Theoreme des 4 couleurs

Considérons la carte de l'Europe et supposons que tout point du plan est à la frontière d'au plus 3 pays, et qu'il n'y a pas d'enclave (par exemple, pas de petit morceau d'Angleterre ou d'URSS dans Berlin...). Autrement dit, supposons avoir affaire à une carte planaire simple. De combien de couleurs a-t-on besoin pour colorier les pays de la carte, de sorte que deux pays voisins n'aient pas la meme couleur ?

Etant donné un graphe planaire G, on peut toujours colorier ses faces avec au plus 6 couleurs : c'est une conséquence assez simple de la forumule d'Euler. Avec la notion de chaine de Kempe, cet argument conduit au théorème dit des 5 couleurs (5 couleurs suffisent pour colorier une carte planaire). Le théorème des 4 couleurs est bien plus difficile à démontrer : la première démonstration correcte est due à Appel et Haken (1977), et est en partie une preuve assistée par ordinateur.

Dans ce projet, on demande de proposer une interface qui laisse l'utilisateur dessiner une carte planaire (ie, tracer des frontières), et choisir quatre ou cinq couleurs. Ensuite vous devez colorier la carte géographique avec les couleurs données. On pourra dans un second temps proposer à l'utilisateur de colorier quelques pays au départ : il faudra alors compléter le coloriage en respectant les contraintes (si possible).

(on pourra envisager d'autres interfaces utilisateur, à l'initiative des étudiants)

0.2 Theoreme de Vizing

Considérons un graphe G simple (pas de boucle, ni de multi-aretes). Soit $\Delta = \max_{v \in V(G)} deg(v)$ le degré maximal du graphe. On veut colorier les aretes du graphe, de sorte qu'en chaque sommet v, toutes les aretes quittant v aient chacune une couleur différente. Le théorème de Vizing énonce qu'on peut toujours y arriver, avec au plus $\Delta + 1$ couleurs.

Dans ce projet, on demande de proposer une interface qui laisse l'utilisateur générer un graphe simple (éventuellement on lui fait d'abord choisir un degré maximal). Il s'agit ensuite de trouver un coloriage des aretes, avec au plus $\Delta + 1$ couleurs, en suivant l'algorithme de Vizing. Dans un second temps, on pourra proposer une coloration en au plus Δ couleurs, si celle-ci existe.

0.3 Jeu de lumières

On offre à l'utilisateur une grille de taille $N \times n$, représentant les rues d'une ville (plan de ville à la Manhattan), et il place des voitures à certaines intersections. Une première variante est d'illuminer les voitures : le jeu consiste à choisir certaines voitures, leur faire allumer leurs phares (qui seront truqués pour éclairer dans les deux directions, et non une seule). Le but du jeu est d'allumer un minimum de voitures, mais assez pour que toutes les voitures placées au départ soient éclairées.

A partir d'une solution proposée par le joueur, s'il n'a pas choisi un nombre minimal de voitures, vous devez passer "continument" de sa solution à une solution minimale (une video qui fasse s'animer les voitures serait bien). On pourra ensuite créer un jeu s'inspirant de cette premiere variante : par exemple avec des lumières variées plus ou moins lumineuses, le but pour le joueur étant qu'un maximum de voitures reçoivent un éclairage supérieur à une valeur minimale donnée. Une courte animation video entre deux niveaux du jeu, est souhaitable.

0.4 Algorithme de Buss pour Vertex Cover

Considérons un graphe $G = (V, E)$ simple. (on pourra proposer à l'utilisateur de créer un graphe entièrement, ou bien de modifier un peu un graphe existant qui lui est proposé). Au départ tous les sommets sont colorés en noir. Il y a un paramètre k en entrée, plus petit que le nombre total de sommets. On voudrait sélectionner certains sommets de G, les colorier en blanc, de sorte que n'importe quelle arete du graphe ait au moins une de ses deux extrémités coloriée en blanc. Un tel sous-ensemble de V, est appelé ensemble couvrant.

Votre programme doit déterminer s'il existe ou non, un sous-ensemble couvrant $W \subset V$, qui soit de taille au plus k. Lorsqu'un tel sous-ensemble existe, on demande une animation qui dévoile pas à pas un W qui marche. On pourra suivre l'algorithme de Buss pour exécuter cette partie.

Variante : Vous proposez à l'utilisateur un graphe G. Il a la possibilité de retirer des sommets (ce qui enlève aussi les aretes qui en partent). Il a le droit de retirer au plus k sommets. Le but du jeu est que le graphe obtenu à la fin, ne comporte plus aucun triangle. Pour cette variante : vous devez au préalable effectuer une vérification algorithmique qui vous dise si votre graphe admet effectivement (ou pas) une solution en au plus k sommets (à retirer). Si le joueur échoue, proposer, à partir de ses choix, comment continuer pour n'avoir plus de triangles (sans tout effacer).