TP : Coloration de graphes d'intervalles

MPI, Lycée Leconte de Lisle

Vous vous occupez de l'attribution des emplacements d'un camping pendant le mois de juillet 2022. On vous a communiqué le planning de réservation suivant :

Numéro	Nom	Jour d'arrivée	Jour de départ
0	Fabien	20	28
1	Dylan	14	22
2	Brice	3	7
3	Aurore	2	15
4	Corentin	5	25
5	Edouard	16	18
6	Gregory	26	30

Vous devez attribuer à chaque personne un emplacement de camping numéroté de 0 à $C-1$ de telle sorte à ce que deux personnes ne se retrouvent pas en même temps sur le même emplacement (le jour de départ, l'emplacement est considéré toujours occupé). Afin d'optimiser la gestion du camping, on cherche également à minimiser le nombre d'emplacements utilisés pendant la saison, c'est-à-dire la quantité C.

1 Formalisation du problème

Pour tout numéro de réservation k, on associe à la réservation numéro k un intervalle

$$I_k = [\text{jour d'arrivée}, \text{jour de départ}].$$

On définit également le *graphe d'intervalle* associé au problème, qui est un graphe non orienté $G = (S, A)$, défini ainsi :

- $S = \{I_0, \dots, I_{n-1}\}$ est l'ensemble des intervalles de réservation
- $\{I, J\} \in A$ lorsque $I \cap J \neq \varnothing$.

Question 1

Représenter le graphe d'intervalle G_{ex} correspondant à l'exemple traité.

Une *coloration* consiste à attribuer une couleur à chaque sommet du graphe. Formellement, il s'agit donc d'une application $c : S \to [|0, C-1|]$ où C est le nombre de couleurs maximal autorisé. De plus une coloration vérifie toujours la propriété suivante :

$$\forall x \in S,\ \forall y \in S,\ \{x, y\} \in A \Rightarrow c(x) \neq c(y)$$

signifiant que deux sommets adjacents du graphe ne peuvent avoir la même couleur. Une coloration est optimale si C est minimal. La valeur C optimale pour un graphe donné est appelé *nombre chromatique* du graphe et notée $\chi(G)$.

Question 2

Déterminer une coloration optimale du graphe G_{ex} en justifiant pourquoi elle est optimale.

Dans un graphe on appelle *clique* un sous-ensemble de sommets $K \subset S$ tel que

$$\forall x \in K,\ \forall y \in K,\ \{x, y\} \in A.$$

La taille de la plus grande clique d'un graphe est appelée *nombre clique* et notée $\omega(G)$.

Question 3

Déterminer les cliques de G_{ex} et en déduire $\omega(G_{ex})$.

Question 4

Justifier que pour tout graphe G, $\omega(G) \leq \chi(G)$.

2 Implémentation en C

On représentera un graphe par *listes d'adjacence* ainsi

```
struct sgraphe {
    int n; // Nombre de sommets
    int voisins[999][1000];
};
typedef struct sgraphe graphe;
```

On considérera que le graphe contient $n \leq 1000$ sommets numérotés de 0 à 999 (au maximum). Le tableau `voisins[i]` contiendra en première case le nombre de voisins sortants du sommet numéro i et dans les cases restantes, la liste de ses voisins sortants.

Question 5

Estimer l'espace mémoire utilisé pour représenter un graphe. Est-il préférable d'allouer les graphes sur le tas ?

Question 6

Écrire une fonction

```
bool conflit(int a, int b, int c, int d)
```

qui détecte si deux intervalles $[a, b]$ et $[c, d]$ sont en conflit. On fournira un jeu de test bien choisi et une fonction de test pour cette fonction.

Question 7

Dans la fonction `main`, déclarer puis initialiser deux tableaux d'entiers `debut` et `fin` contenant 1000 cases entières représentant respectivement les jours d'arrivées et de départ, ainsi qu'un entier n représentant le nombre de réservations dans le planning.

Question 8

Écrire une fonction
`void construit_graphe(graphe* g, int n, int debut[1000], int fin[1000])`

qui construit le graphe d'intervalles associé au problème de réservation fourni. Déterminer la complexité de cette fonction.

Question 9

Écrire une fonction
`int degre(graphe* g, int x)`

retournant le degré (sortant) d'un sommet donné. Utiliser cette fonction pour vérifier la bonne construction de G_{ex}.

Question 10

Écrire une fonction
`bool est_clique(graphe* g, int K[], int n)`

qui teste si un ensemble de sommets donné sous forme de tableau `K` de longueur `n` est une clique de `g`. Vérifier cette fonction sur G_{ex}.

3 Résolution glouton

Un algorithme de résolution utilisant une *stratégie glouton* est la suivante :

- Traiter les sommets dans l'ordre de 0 à $n-1$.
- Initialement, les sommets n'ont pas de couleur, on dit alors que leur couleur est -1.
- Pour chaque sommet, choisir la plus petite couleur disponible qui n'a pas déjà été choisie par un de ses voisins.

Question 11

Appliquer l'algorithme de coloration glouton à G_{ex}. Remarquer que la coloration obtenue n'est pas optimale.

On représentera une coloration par un tableau `couleurs` contenant la couleur de chaque sommet (éventuellement -1 en cas d'absence de couleur).

Question 12

Écrire une fonction
`int couleur_disponible(graphe* g, int couleurs[], int x)`

retournant la première couleur disponible pour le sommet x supposé non coloré.

Question 13

En déduire une fonction

```
void coloration_glouton(graphe* g, int couleurs[])
```

qui remplit le tableau de couleurs avec les couleurs choisies par l'algorithme glouton.

Question 14

En déduire un programme qui affiche à l'écran une solution (non nécessairement optimale) au problème de réservation.

4 Coloration optimale

La coloration optimale d'un graphe quelconque est un problème algorithmique difficile. On ignore aujourd'hui s'il existe un algorithme en temps polynomial pour le résoudre. Toutefois, dans le cas des *graphes d'intervalle* on sait résoudre efficacement le problème.

Pour obtenir une coloration optimale il suffit d'appliquer l'algorithme glouton mais en traitant les sommets par ordre croissant selon l'ordre suivant

$$[a, b] \leq [c, d] \Leftrightarrow a \leq c$$

Question 15

Appliquer cet algorithme de coloration à G_{ex}. Remarquer que la coloration obtenue est optimale.

Question 16

Modifier votre programme pour qu'il effectue ce tri en prétraitement des données. On pourra utiliser un nouveau tableau pour se rappeler l'ordre initial.

Question 17

Démontrer que l'algorithme glouton avec tri des intervalles fournit une solution optimale. Indication : montrer que pour chaque sommet, le numéro de couleur choisi est inférieur ou égal à $\omega(G) - 1$.

5 Compléments

Question 18

Adapter le programme pour qu'il puisse lire les données du problème à partir d'un fichier texte. Le fichier pourra ressembler à :

```
7
Fabien,20,28
Dylan,14,22 (etc)
```