Projet « Diaporama »

Le « Google Hash Code » est une compétition de programmation pour des équipes d'étudiants ou de non-étudiants. La compétition est basée sur un problème, le plus souvent en optimisation combinatoire, et sur des instances de taille importante à résoudre. Le gagnant de cette compétition est celui qui a le meilleur « score », c'est-à-dire une solution optimisant la fonction objectif sur ces instances.

Pour ce projet, je vous propose un problème tiré du « Qualification Round » du « Google Hash Code 2019 ».

On considère dans ce projet la résolution d'un problème de choix d'ordre de passage de photos dans une présentation vidéo.

Vous pouvez trouver la version du projet à la page **https://codingcompetitions.withgoogle.com/hashcode/archive** sous la forme d'un sujet **hashcode2019_qualification_task.pdf** et d'une archive de plusieurs instances appelée **qualification_round_2019.in.zip**.

1 Définition du problème

L'énoncé suivant reprend celui du sujet initial du Google HashCode qu'il est conseillé de consulter.

1.1 Enoncé du problème

Données :

On considère un ensemble de N photos. Chaque photo pour $i \in \{1, 2, \ldots, N\}$ est caractérisée par :

- une orientation : horizontale (H) ou verticale (V),
- un nombre M_i de mots-clé (tags en anglais) : $1 \leqslant M_i \leqslant 100$,
- une liste de M_i mots-clé. Un mot-clé est une chaîne de 1 à 10 caractères minucsules ASCII, sans espace.

Sortie :

On veut construire une présentation des photos en diaporama (photo slideshow, en anglais). Une présentation est définie comme une liste ordonnée d'au moins une diapositive.

Une diapositive (slide, en anglais) contient :

- soit une unique photo horizontale,
- soit exactement deux photos verticales (mises côte à côte).

Le fichier de sortie qui définit la présentation contiendra sur la première ligne un entier D égal au nombre de diapositives de la présentation. Les D lignes suivantes contiendront :

- soit un seul entier qui est le numéro (de 0 à $N-1$) d'une photo horizontale,
- soit deux entiers séparés par un espace qui sont les numéros de deux photos verticales (l'ordre des deux entiers n'est pas important).

Chaque photo ne peut être utilisée qu'au plus une fois (zéro ou une).

1.2 Score d'une solution

Toute la difficulté de ce problème repose sur la façon dont est calculé le «score» associé à une présentation. Le score mesure la qualité des transitions entre deux diapositives. Le but de ce score est d'inciter à ce que, à la fois, l'enchaînement de deux diapositives possède une certaine logique mais aussi à ce que deux diapositives successives ne soient pas trop similaires.

On peut imaginer qu'une telle présentation soit diffusée pour animer par exemple une salle d'attente (aéroport, hôpital,...) ou pour présenter les plus belles photos d'un artiste sur un site internet.

La liste des mots-clé d'une diapositive est :

- soit la liste des mots-clé de son unique photo horizontale,
- soit l'union (sans doublon) des deux listes de ses deux photos verticales.

Dans une présentation, $S = (S_1, S_2, \ldots, S_k)$, on appelle transition le passage d'une diapositive S_i à la diapositive suivante S_{i+1}. Le score d'une transition entre deux diapositives successives S_i et S_{i+1} est le minimum entre :

- le nombre de mots-clé communs entre S_i et S_{i+1},
- le nombre de mots-clé apparaissant dans S_i et non dans S_{i+1},
- le nombre de mots-clé apparaissant dans S_{i+1} et non dans S_i.

Le score d'une présentation de k diapositives est la somme des $k-1$ transitions entre ses diapositives. Une présentation d'une unique diapositive a donc un score nul.

1.3 Format des fichiers

Le fichier archive contient cinq instances du concours Google Hash Code : une instance de description : **a_example.txt**, une instance test : **c_memorable_moments.txt**, et trois instances de 80000 à 90000 photos (ces trois instances sont repérées par les lettres **b**, **d** et **e**).

Voici l'instance **a_example.txt** qui permet de décrire le format des instances :

```
4
H 3 cat beach sun
V 2 selfie smile
V 2 garden selfie
H 2 garden cat
```

On peut lire que cette instance contient 4 photos. La première est une photo horizontale avec une liste de 3 mots-clé qui sont l'ensemble **{cat, beach, sun}**.

Notez que pour la suite, cette première photo sera dite d'indice 0. La deuxième est une photo verticale qui sera d'indice 1, etc.

En retour, vous devrez produire une présentation dans un fichier de nom **solution_x.txt** où **x** est la lettre du problème, qui doit respecter le format très simple suivant. Ce format est indiqué ici par l'exemple d'une présentation qui est valide pour l'instance **a_example.txt** précédente.

```
3
0
3
2 1
```

Ce format correspond à une présentation de 3 diapositives. La première vignettre contient une photo H qui est celle de numéro 0 dans le fichier **a_example.txt**.

La deuxième diapositive contient une photo **H** de numéro 3. La troisième diapositive contient deux photos **V** de numéro 2 et 1 (rappelez-vous que l'ordre des deux photos **V** dans une diapositive n'a pas d'importance).

2 Entrées-Sorties

Récupérer le code C++ fourni pour le projet. Vous pouvez le compiler sous linux en lançant la commande **make** dans le répertoire. Il vous permettra d'obtenir le score pour vos fichiers solutions.

L'exécutable prend deux entrées : l'instance du problème et votre solution :

./Checker instance.txt ma_solution.txt.

3 Résolution :

Le projet consiste à essayer d'avoir les meilleurs résultats possible mais aussi à étudier différentes techniques de résolutions pour le problème. Vous pouvez utiliser le langage C ou Ocaml au choix.

Tout au long de ce projet, vous allez devoir faire des compromis entre tailles d'instances résolues et temps de calcul. En effet, les instances fournies par Google Hash Code sont très grandes.

Il peut être intéressant, pour chaque méthode utilisée, d'évaluer votre méthode sur des instances de tailles croissantes.

Vous ne pouvez pas réutiliser du code sur internet, mais vous pouvez par contre réutiliser le code utilisé en cours, en TD ou en TP lors de ces deux dernières années.

Si vous ne savez pas dans quelle direction partir, reprenez vos cours d'algorithmique de ces deux dernières années et essayez d'appliquer une des méthodes vues.

Le projet sera fait par groupe. Il vous faut donc former 6 groupes de 4 ou 5 personnes. Il ne doit pas y avoir plus de 2 MPI* par groupe. Il est fortement recommandé d'utiliser **git** pour pouvoir avancer sur le code en parallèle.

4 Rendu

Le rendu consiste à me renvoyer par courriel (avec dans le sujet **[projet_photos]**) une archive portant les noms des membres du groupe.

Cette archive contiendra :

- pour chaque instance **a**, **b**, **c**, **d** ,**e**, une solution sous forme d'un fichier texte (nommé **solution_a.txt** si c'est celle du problème **a**).
- le ou les algorithmes utilisés. Pour chaque algorithme, vous indiquerez (en commentaire dans le fichier, mais aussi dans le nom du fichier : **algorithmenaif_a_c.txt**) quelles solutions il a produites (instances **a** et **c**).

  Vous pouvez aussi rendre un algorithme qui n'a obtenu aucune meilleure solution mais qui présente un intérêt particulier.

  Vous commenterez chacun des algorithmes renvoyés. Vous serez très rigoureux sur les noms de variables et de fonctions : c'est indispensable quand on travaille en groupe.

  Pour chaque solution, l'algorithme correspondant doit renvoyer cette solution (si vous avez des algorithmes probabilistes, vous fixerez la graine pour le rendre déterministe).
- un rapport en pdf. Le rapport contiendra :
  - D'abord, pour chaque instance, une ligne de commande exacte pour utiliser votre code et produire la solution correspondante.
  - pour chaque algorithme, une explication brève de son fonctionnement. Les explications plus approfondies seront fait en commentaires dans le code.
  - ensuite, vous détaillerez et expliquerez vos choix d'algorithmes. Pourquoi vous avez choisi de résoudre le problème de cette manière : pourquoi pensez-vous que c'est une bonne méthode ? Quelle idée avez-vous eu qui a permis à votre algorithme d'être performant ?

    Vous pouvez (mais ce n'est pas obligatoire) présenter des analyses théoriques ou pratiques du problème et des algorithmes.
  - enfin, un des algorithmes rendus (ça peut être l'algorithme joker, celui qui ne sert pour aucune solution) devra utiliser une des techniques vu en cours cette année. Dans votre rapport, vous mentionnerez lequel et vous expliquerez comment vous avez choisi d'implémenter la méthode du cours.

5 Notation :

Le barème s'appuiera sur les quatre items suivants, avec une pondération à décider :

- la performance : un classement sera établi en calculant la somme des points obtenues sur chacune des 5 solutions. Cette somme et votre classement servira à calculer la note de performance.
- le code : la clarté, la lisibilité et les commentaires seront bien évidemment pris en compte.
- le rapport.
- l'algorithme qui utilise une méthode vu en cours.

Il y aura probablement aussi un classement intermédiaire qui ne comptera pas dans la note finale et je demanderais peut-être à l'équipe gagnante de faire une petite présentation de leur algorithme.