Projet Google Jam

Algorithmique et programmation fonctionnelle

David RIGAUX, Mehdi DALAA et Victor ROUX

Sommaire

Introduction

Le but du projet était de participer au concours annuel de programmation organisé par Google, le “Google Code Jam”. Comme le concours à lieu pendant le printemps, nous ne pouvions pas participer en direct, mais Google offre la possibilité de participer aux éditions précédente en temps non limité. Pour avancer dans le projet nous devions réaliser plusieurs exercices classés en différents rounds : Qualification, Round 1, Round 2, Round 3 et la Finale. Cependant, il faut savoir que chaque round n’est accessible que si le round précédent n'a été accompli. Pour les quatre premiers rounds il faut obtenir un minimum de 30 points pour accéder au round suivant, quant à la Finale nous n’avons besoin que de quinze points. Chaque exercices a deux niveaux différents, qui donne un nombre de points différents, Small et Large. Ce projet est écrit en OCaml et nous avons utiliser l’ensemble des connaissances acquises lors des quatre premières séances du cours (fonctions, récursivité, pattern matching, types et listes), mais dans certains cas nous sommes aller chercher d’autres notions pas encore vu en cours que nous avions besoin. Dans ce rapport nous allons expliquer nos démarches et notre algorithme de chaque exercice.

Étude du projet

Il est important ici de tout d’abord avoir un apriori des programmes que nous allions faire. C’est pour cela que pour chaque round nous avons commencé à regarder le nombre de points que touts les exercices donnés et nous concentrions sur les programmes qui nous offrait le plus de points et les plus simple possible. Avec cela, nous pouvions avoir une liste des exercices qu’on allait programmer. Pour chaque exercice, il nous fallait aller sur le lien de l’exercice en question et soit télécharger le fichier d’entrée du Small ou du Large. Il nous était fournis un extrait de code nous permettant de lire le fichier d’entré et de ressortir les résultats selon les critères imposé, ceci après avoir réussi l’algorithme permettant d’avoir le résultat. Pour chaque exercice, le procédé étude est complètement différent les uns des autres car chaque exercice est différent.

Qualifications

Ce *Round* est le tout premier du projet. Pour ce tour nous avons choisis de faire le problème *I/O Error* en Small qui vaut 9 points, ainsi que *Cody's Jam* également en Small qui nous rajoute 10 points et finalement *Captain Hammer* qui nous permets d'atteindre les 30 points valant 11 points. Nous allons maintenant essayer d'entrer un peu plus dans les détails de chaque exercices.

I/O Error Small

Le principe de ce problème est qu'il faut traduire un certain nombre de suite de "I" et de "O" sous forme de caractères qui nous donnera une certain chaîne caractères. Sachant que chaque caractère est codé sur 8bits il faut alors faire des paquets de 8. Mais avant de faire tout cela il faut lire le fichier texte. Nous avons donc écrits une fonction qui a le nom de *lecture* qui nous permets de lire une ligne du fichier d'entrée et la lis sous forme de String. Nous transformons ensuite touts les "I" en "1" et touts les "O" en "0". Nous avons alors un string de "1" et de "0". Maintenant il faut qu'on fasse des paquets de de 8 caractère, et pour cela nous formons des sous-strings de 8 caractère. Après avoir fais des recherches sur internet, nous avons trouvé un moyen de faire passer des strings directement sous forme binaire en ajoutant "0b" devant un string de "0" et de "1". Nous avons donc fais en sorte d'ajouter ces deux caractères devant chacun de nos strings et procéder par la suite à une conversion des strings en binaire puis du binaire en caractère, sous forme de liste de caractères. Il ne nous suffit qu'à faire de la liste de caractère, une chaîne de caractère. Ce qui nous permets donc de résoudre le problème.

Cody's Jam

Dans cet exercice il s'agit d'un vendeur qui compte bientôt mettre en réduction touts ces articles. Il a donc envoyé l'impression des nouveaux prix après réduction et ceux avant réduction à l'imprimeur. Cependant, en recevant ces étiquettes, l'imprimeur les a toutes mis dans le même paquet. Fort heureusement, touts les prix ont la même réduction de 75%. Le but de cet exercice est de trouver les prix après reduction. Pour faire cela, nous savons que les prix données sont dans l'ordre croissant donc les prix après réduction sont forcément avant ceux avant réduction. Nous multiplions alors la première valeur par 4/3 pour trouver la valeur du prix avant réduction et supprimons cette valeur de la liste et ainsi de suite jusqu'à arriver à la fin de la liste à l'aide d'un appelle récursif.

Captain Hammer

Dans cet exercice il s'agit de calculer un angle de lancement. Alors en effet, le Hamjet est un vaisseau super puissant. Si puissant qu'au décollage il épuise tout son réservoir d'essence au décollage. Il faut alors qu'on calcul l'angle de lancement, ayant comme données la vitesse de lancement ainsi que la distance que Captain Hammer veut parcourir. Il s'agit tout simplement d'appliquer des formules physiques pour résoudre ce problème. Ce programme est relativement court à faire mais le plus dur est de trouver la bonne formule à appliquer.

Round 1

C'est à partir de maintenant que nous avons remarqué que le niveau a augmenté. En effet ce round à été très dur.

Egg Drop

Dans ce problème on s'imagine dans un immeuble avec des un nombre d'œufs identiques en notre possessions. Chaque œuf est placé dans une boîte servant à le protéger. On veut savoir pour chaque étage si un œuf jeté de ce dernier se brisera ou non. On sait que si un œuf se brise en tombant d'un étage j, alors tous les œufs se briseront s'ils sont lancés d'un étage i tel que i plus grand ou égal à j. De même si un œuf ne se brise pas en tombant d'un étage j, alors tous les œufs jeté d'un étage i plus petit ou égal à j ne se briseront pas. Il faut alors calculer l'étage le plus haut au quel on peut laisser tomber l'oeuf avec le moins d'oeuf mis a notre disposition ainsi que le moins d'oeuf cassé. Pour résoudre ce problème il faut se baser sur la probabilité que l'oeuf se casse et appliquer une formule qui nous aide à la calculer. Cette formule est : $F(B,D) = F(D-1,B-1) + F(D-1,B)$
Il nous suffit juste de coder cette formule ainsi que quelques autres fonctions pour pouvoir le faire.

Round 2

Number Game

Ici il s'agit d'un jeux à deux joueurs, nous allons prendre comme prénom Justine et Tessa. Dans un premier temps, deux nombres entiers positifs A et B sont écrits sur un tableau noir. Les joueurs se relaient, en commençant par Tessa. À son tour, un joueur peut remplacer A avec A - k * B pour tout entier positif k, ou remplacer B avec B - k * A pour tout entier k positif. La première personne à faire un des numéros négatif ou égal à 0 perd. Par exemple, si les numéros de départ sont (12, 51), le jeu peut progresser comme suit:

> Tessa remplace 51 avec 51-3 * 12 = 15, laissant (12, 15) sur le tableau noir.
> Justine remplace 15 par 15-1 * 12 = 3, ce qui laisse (12, 3) sur le tableau.

Tessa remplace 12 avec 12-3 * 3 = 3, laissant (3, 3) sur le tableau noir.
Justine remplace le 3 avec 3 - 1 * 3 = 0, et perd.

Nous dirons que (A, B) est une position gagnante si Tessa peut toujours gagner un jeu qui commence par (A, B) sur le tableau noir, peu importe ce que Justine fait.

Il nous faut alors calculer les meilleurs cas ou Tessa gagne.

Box Factory

Dans l'exercice *Box Factory*, il faut sortir la valeur en fonction d'une suite de valeurs le maximum de jouets qu'une usine peut envoyer a ces clients. En effet, cette usine en question possède deux chaînes de production : une pour les jouets et la deuxième pour les boîtes. Pour représenter chaque ligne de production, on a deux lignes, une pour chaque unité. En fonction de la suite de valeurs qu'on a on sait qu'on a toujours une séquence de deux valeurs qui vont ensemble, par exemple a1 A1, ici on a un nombre a1 de jouets ou de boites A1. En fonction de cela on stock dans une liste le couple du nombre d'objets avec son type de jouet ou de boîte. Il faut alors faire des soustractions entre les listes pour pouvoir calculer le nombre de jouets qu'on peut mettre dans des boites pour envoyer.

Round 3

Mountain View

On part à l'aventure ! En effet, pour pouvoir comprendre ce problème il faut s'imaginer en montagne avec plusieurs sommets devant nous. Il faut savoir qu'un sommet peut en cacher un plus grand derrière, qui n'est donc pas dans notre angle de vue. Les données qu'on a en entré est la position du sommet qui nous semble le plus haut en fonction d'où on est. Sachant qu'on commence au sommet 1 et que l'on se dirige vers le sommet le plus haut. Il faut savoir qu'ici il existe une infinité de sortie possible mais il faut qu'elle soit cohérente pour pouvoir être considéré comme bonne.

Test Passing Probability

Parlons probabilité. Dans ce problème il y a un certain Dave qui souhaite passer un examen en ligne, il s'agit plus précisément d'un QCM. Il a un nombre limité de tentatives pour chaque test. Il mets alors une probabilité de bonne réponse à chaque choix. Il faut alors qu'on calcul en fonction de chaque probabilité de chaque choix et du nombre d'essais possible, quelle est sa probabilité de réussite. Nous n'avons pas réussis à finir ce problème, car nous avons un problème d'indice dans nos *Array*.

Evaluation

Passons maintenant à de la déclaration de fonctions. Ici, nous avons une liste de fonctions qui sont définies en fonctions de variables. Il se peut également qu'il y ait des variables qui sont définies. Notre but est alors de dire si une certaine séquence de fonctions est correcte ou pas. Pour quelle soit correcte il faut que toutes les fonctions soient correctement définies donc que toutes les variables de toutes les fonctions soient définies. Pour faire cela nous avons remarqué qu'il nous fallait en effet 3 listes. Une liste avec les fonctions et variables qui sont bien déclarés (***fb***), une autre avec le couple (nom_fonctions, [liste_vars]) où on mets les fonctions qui ne sont pas encore bien définies car ces variables ne sont pas définies (***fa***) et une dernière pour les variables qui ne sont pas encore définis (***va***). D'après nous il faut faire :

1. On lis une fonction ainsi : nom_fonct=x([liste_vars])
2. Si nom_fonct=(), on ajoute nom_fonct directement a la liste ***fb***
3. Si touts les elements de [liste_vars] sont dans ***fb*** alors mettre nom_fonct dans ***fb***
4. Si un ou plusieurs elements de [liste_vars] ne sont pas dans ***fb*** alors on ajoute [liste_vars] dans ***va***, mais ne faisant un test pour savoir si certaines de ces variables y sont deja pour ne pas en avoir en 2 fois. Et par la suite ajouter dans la liste ***fa*** (nom_fonct,[liste_vars]
5. Apres avoir lu chaque fonction de la liste, faire une operation qui permet de comparer les deux listes et voir si il n'y a pas des elements dans ***fb*** qui sont dans va et ainsi le supprimer dans va, et faire la meme opération dans la liste fa pour les variables des couples et au fur et à mesure vider les deux liste ***fa*** et ***va*** et remplir celle de ***fb***.
6. Finalement, après avoir lu toute la liste des fonctions, si la liste ***va*** n'est pas vide alors cette sequence de fonctions n'est pas bonne, sinon elle est bonne.

Cependant nous n'avons pas réussi à finir le codage car nous avons rencontré des difficultés.

Finale

Nous n'avons pas pu accéder à la finale.

Bilans personnel

Bilan personnel - David RIGAUX

Pour ma part j'ai trouvé ce projet très intéressant car il s'agit d'un concours de Google et étant un grand fan du géant Américain cela m'a beaucoup motivé. Mais après avoir commencé à codé je me suis directement rendu compte que ce projet aller être beaucoup moins simple que ce dont j'envisageait. En effet, je me débrouillais bien en programmation procédurale mais quand on a commencé la programmation fonctionnelle je me suis rapidement perdu dans les recursions etc, qui ne sont pas ma tasse de thé. J'ai dont trouvé ce projet relativement dur, si ce n'est pour moi l'un des plus dur dû à la programmation fonctionnelle. Mais nous avons réussis à travailler comme un groupe et j'ai personnellement réussi à apprendre des choses sur la programmation fonctionnelle et surtout sur OCaml. Mais j'ai personnellement hâte de passer au développement web.

Conclusion

Après avoir essayé jusqu'au bout et dû à la difficulté des problèmes et du fait que nous venons de commencer la programmation fonctionnelle avec OCaml, nous n'avons pas réussis à tout finir, car nous avons rencontré pas mal de difficultés. Cependant, nous sommes relativement content de ce dont nous avons réussis à faire. Mais on fera mieux la prochaine fois.