Mise en place d'une plateforme de programmation compétitive comme support de TDs

Thibault Raffaillac, Francis Chavanon, Romain Vuillemot

LIRIS, École Centrale de Lyon, Université de Lyon, France

Abstract. Dans cet atelier, nous présentons l'utilisation d'une plateforme Web de programmation compétitive (DMOJ), pour la validation automatique des programmes des étudiants dans le cadre de travaux pratiques de programmation. Nous fournissons une image de machine virtuelle (au format OVA) sur laquelle la plateforme est déjà installée et paramétrée. Son utilisation est illustrée avec la résolution de divers types problèmes en ligne. Les participants sont invités à adapter des sujets existants (ou en créer de nouveaux) pour permettre leur validation en ligne avec DMOJ.

Keywords: programmation compétitive, algorithmique, structures de données, outil numérique, autonomie, conception de TP

1 Introduction

Dans le cadre de travaux pratiques en programmation, la mise en place d'outils numériques pour l'évaluation automatique de programmes écrits par les étudiants est notoirement complexe. Il s'agit d'exécuter du code inconnu, dont on souhaite contrôler les entrées et sorties, interdire les effets de bord et garantir une exécution égalitaire et reproductible entre tous les programmes. En plus de cela il faut permettre la récupération des programmes des étudiants, leur envoi à un évaluateur automatique et la transmission de sa réponse aux étudiants, ce qui implique généralement un site Web.

Nous proposons l'utilisation d'un outil issu de la programmation compétitive pour le support des travaux pratiques de programmation (voir figure 1 et [1]), ce qui est une pratique déjà éprouvée dans l'enseignement supérieur [4, 3, 2]. Une *compétition de programmation* consiste en un ensemble de problèmes logiques ou mathématiques à résoudre en temps limité (de 1h à 24h), seul ou en équipe. La résolution d'un problème se fait en écrivant un programme qui pour toute donnée d'entrée donne une réponse en sortie. La majorité des compétitions utilise communément les entrées/sorties standard `stdin` et `stdout` d'Unix. Les formats des données sont toujours rigoureusement spécifiés (nombres et types des valeurs, bornes maximales et minimales). Les participants soumettent chaque programme sur une plateforme (*juge en ligne*) qui l'exécute plusieurs fois avec des données d'entrée secrètes, et compare chaque sortie à celle attendue. Enfin une soumission

est jugée valide si elle compile correctement, s'exécute sans erreur, n'est pas interrompue par limite de temps ou de mémoire, et donne la bonne réponse pour chaque test d'entrée.

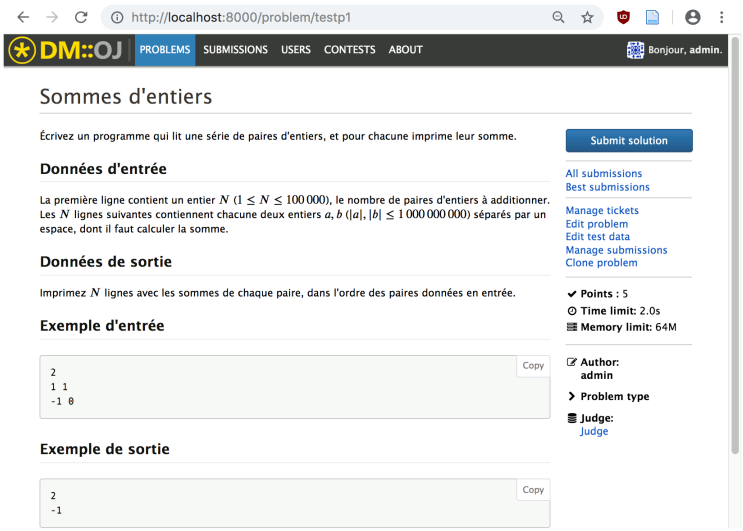

Fig. 1. Illustration de la plateforme DMOJ avec un problème simple.

Le milieu de la programmation compétitive est aujourd'hui soutenu activement par de nombreuses grandes entreprises d'informatique, pour les compétences qu'elles enseignent aux étudiants :

- l'écriture de code rapide, fiable et compact
- la planification et la gestion du temps dans la résolution de problèmes
- la maîtrise et l'application de nombreux algorithmes et méthodes de programmation

2 Conception d'exercices

Les plateformes issues des compétitions de programmation sont conçues pour exécuter les programmes dans des environnements isolés et contrôlés. Leur contrainte est que les données ne soient passées *que* par `stdin` et `stdout`. Il faut donc concevoir des problèmes qui spécifient les entrées et sorties en texte (DMOJ permet aussi des données binaires). En particulier, ce type de plateforme ne permet pas a priori la validation automatique d'interfaces graphiques, pour lesquelles les sorties à évaluer seraient difficiles à quantifier en données.

Nous distinguons typiquement deux types de problèmes :

– Les problèmes avec une ou plusieurs solutions exactes connues pour chaque entrée. Chaque programme est exécuté sur la plateforme, et un éventuel programme supplémentaire peut calculer la validité de la solution produite si plusieurs sont possibles.
– Les problèmes d'optimisation (NP) dont les solutions optimales ne sont pas connues étant donnée la taille des données d'entrée. On peut demander à l'étudiant d'envoyer un programme (généralement avec une limite de temps plus importante), ou directement sa solution calculée pour une entrée connue. Un programme supplémentaire calcule un score pour chaque solution, et un classement des meilleures solutions soumises est établi.

Le but d'un sujet de travaux pratiques est de guider les étudiants dans l'écriture d'un programme qui implémente une ou plusieurs fonctionnalités. La plateforme d'évaluation en ligne est là pour valider les résultats finaux, voire des étapes intermédiaires. Lors de la conception ou l'adaptation d'un sujet qui l'utilise, il faudra se poser les questions suivantes :

– *Quelles étapes/fonctions sont les plus sources d'erreurs pour les étudiants ?* On en sélectionne alors une poignée pour validation en ligne (principalement celles qui pourraient impacter négativement le reste du travail).
– *Comment formuler le résultat de chaque étape comme une fonction qui transforme des données ?* Il s'agit de rédiger le sujet pour demander l'écriture d'une fonction qui transforme des entrées en sorties (sans effets de bord).
– *Comment intégrer l'évaluation automatique du programme à moindre effort pour les étudiants ?* On choisit de préférence des données faciles à lire en entrée et à produire en sortie (ex. entiers séparés par des espaces).
– *Quelle est la diversité des données d'entrée que les programmes doivent pouvoir gérer ?* Il s'agit par exemple des bornes des valeurs, des nombres d'éléments à traiter, ou des cas de pire complexité pour l'algorithme attendu. On crée alors des jeux de tests pour chacun, et on formule le sujet en ligne pour lever les ambiguïtés.

Enfin, durant les travaux pratiques enseignés dans notre université nous utilisons le langage Python avec Spyder, or celui-ci ne permet pas d'exécuter un programme avec la redirection d'entrées (le chevron `<` dans un terminal). Pour tester leur programme avant envoi à DMOJ, les étudiants sont obligés d'écrire les entrées à la main dans le terminal intégré à Spyder. Lorsque leur programme doit lire beaucoup de données en entrée, nous leur demandons de lire depuis un fichier, et de remplacer une ligne `f=open(...)` par `f=sys.stdin` lors de l'envoi de leur programme.

3 Informations pratiques

Les participants devront disposer d'un ordinateur portable avec VirtualBox installé **au préalable** dessus (c'est la configuration que nous aurons testée). Il est recommandé également d'avoir quelques années d'expérience dans l'administration de Linux, car l'utilisation de DMOJ nécessite parfois une intervention directe sur la machine virtuelle.

References

1. DMOJ: DMOJ, https://github.com/DMOJ
2. Gárcia-Mateos, G., Fernández-Alemán, J.L.: A course on algorithms and data structures using on-line judging. In: Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education. pp. 45–49. ITiCSE '09, ACM, http://doi.acm.org/10.1145/1562877.1562897, event-place: Paris, France
3. Kosowski, A., Małafiejski, M., Noiński, T.: Application of an online judge & contester system in academic tuition. In: Leung, H., Li, F., Lau, R., Li, Q. (eds.) Advances in Web Based Learning – ICWL 2007. pp. 343–354. Lecture Notes in Computer Science, Springer
4. Kurnia, A., Lim, A., Cheang, B.: Online judge 36(4), 299–315, http://www.sciencedirect.com/science/article/pii/S0360131501000185