MARO 037 - Competitive Programming

Fabian Lecron - Arnaud Vandaele

Faculté Polytechnique - Services MIT et MARO

UMONS

Année académique 2024-2025

Séance 00 :
Description et organisation du cours

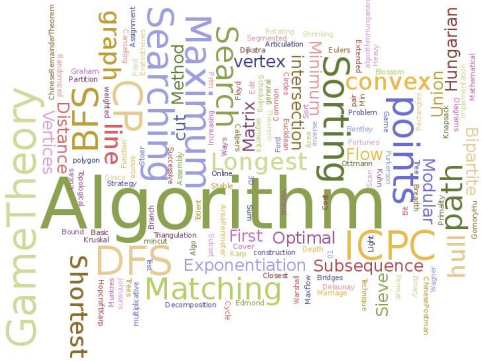
Séance 00 – Description et organisation du cours

Tables des matières

Plan

Qu'est-ce que c'est ?

- Programmer est un exercice intellectuel et, qui dit exercice dit compétition et sport. Le *Competitive Programming* peut donc être défini comme le sport consistant à participer à des compétitions de programmation.
- Les problèmes proposés peuvent varier fortement suivant les compétitions. Par contre, le déroulement est souvent identique : un ou plusieurs problèmes sont proposés et une solution doit être fournie en un temps limité. L'algorithmique, la logique et les maths sont des compétences essentielles pour ces compétitions.



Deux thèmes principaux : le design et l'implémentation d'algorithmes.

- Le coeur du *Competitive Programming* consiste à inventer des algorithmes efficaces résolvant des problèmes bien définis. Souvent, une solution à un problème est une combinaison de méthodes bien connues. Nous passerons sur les principales méthodes.
- Après avoir trouvé un algorithme qui résout un problème, l'étape suivante consiste à l'implémenter correctement. Le *Competitive Programming* diffère grandement du *software engineering* traditionnel : les programmes sont courts (généralement au plus quelques centaines de lignes), ils doivent être écrits rapidement et il n'est pas nécessaire de les maintenir après le concours.

Apprendre à penser et à mettre en oeuvre

Souvent, vous rencontrerez un problème que vous n'avez jamais résolu.
Cependant, on s'attend à ce que vous trouviez quand même une solution !
Comment se préparer ?

1. **Maitriser un langage de programmation** et être le plus à l'aise possible avec sa syntaxe. Lors de cours, ce sera Python car il est facile à apprendre et sa syntaxe est conviviale.
2. **Comprendre le concept de complexité.** Dans la plupart des cas, il existe plus d'une solution à un problème, et ces différentes solutions peuvent prendre plus ou moins de temps et d'espace. Pour résoudre un problème, il est donc important de se familiariser avec ce concept.
3. **Apprenez les principes fondamentaux des structures de données et des algorithmes** : Array, List, Stack, Queue, Tree, Tri, Graph, Recursion, Dynamic Programming. La chose la plus importante est de savoir quoi et quand les appliquer.
4. **Entrainez-vous**, et résolvez un maximum d'énoncés différents. Comme un sport, les réflexes s'acquièrent avec la pratique.

Plan

Objectifs

Deux compétences principales :

- Le design d'algorithmes
- L'implémentation d'algorithmes

A travers cet enseignement, nous comptons :

- vous faire découvrir le monde du Competitive Programming (et les différents types de problème)
- faire naître des vocations et initier une dynamique
- développer vos capacités algorithmiques
- vous faire découvrir Python, sa syntaxe et ses structures de données
- renforcer vos capacités d'implémentation (produire une solution en un temps limité)
- vous faire collaborer sur la résolution d'un problème donné

Plan

Fiche ECTS 2023-2024 pour MARO-037 Competitive Programming

- **Acquis d'apprentissage UE**

  Acquérir des connaissances algorithmiques et des compétences de programmation spécifiques afin d'être capable de résoudre des problèmes d'ingénierie en un temps limité et de participer à des compétitions externes. Une présence aux cours de théorie/exercices de minimum 80% est requise. La présence aux séances pratiques (et aux compétitions) et séminaires éventuels est obligatoire.

- **Contenu de l'UE**

  Partie algorithmique / Partie implémentation (voir description en ligne)

  Nous aborderons ces enseignements en faisant le parallèle avec les énoncés d'anciennes compétitions

  Une présence aux cours de théorie/exercices de minimum 80% est requise. La présence aux séances pratiques (et aux compétitions) et séminaires éventuels est obligatoire.

- **Commentaire sur les évaluations Q2 de l'UE**

  Evaluation sous la forme d'une compétition (implémenter une méthode afin de résoudre un problème en un temps limité). Une évaluation écrite complémentaire sera organisée pour les étudiants n'ayant pas assisté aux séances de travaux pratiques obligatoires et à au moins 80% des cours/exercices.

Epreuve et cotation : en pratique

Pour les étudiants respectant la règle
Une présence aux cours de théorie/exercices de minimum 80% est requise. La présence aux séances pratiques (et aux compétitions) et séminaires éventuels est obligatoire.,

- Les séances de travaux pratiques par groupe de deux
- Des exercices (éventuels) à déposer sur Moodle / autres plateformes
- L'examen de 4h consistera en la résolution individuelle d'un énoncé d'une difficulté similaire aux énoncés abordés durant le cours

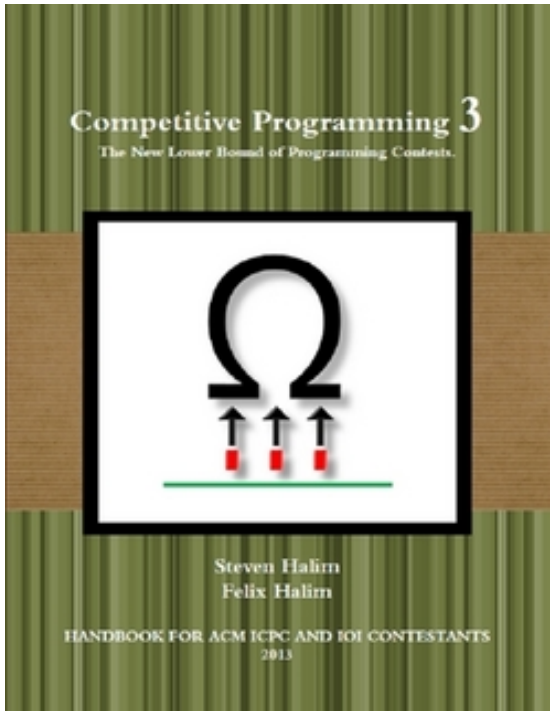
Pour la note (sur 20) :

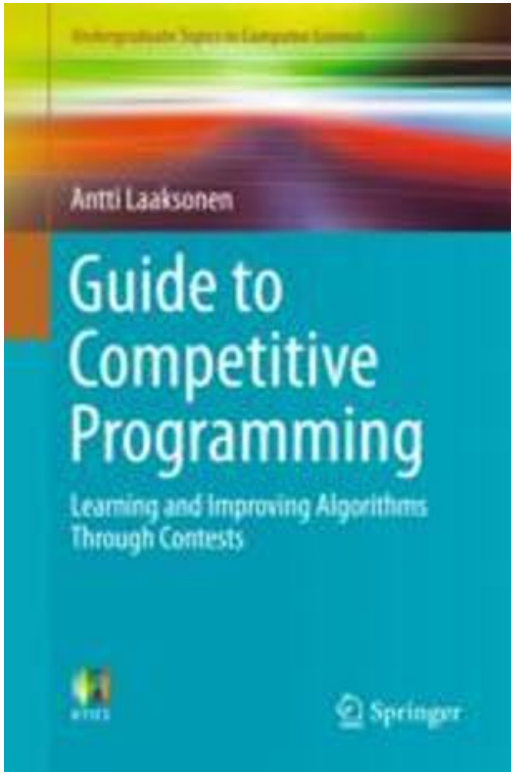
- Note A/20 - évaluation continue (devoirs éventuels et TP) : **acquis** $A = 20$ **ou non-acquis** $A = 0$
- Note B/20 - examen
- Pour ceux qui ont $A = 0$: Note finale $= 0.4 * A + 0.6 * B$

  Pour ceux qui ont $A = 20$: Note finale $= B$

Plan

Références


Competitive Programming 3
The New Lower Bound of Programming Contests.
Steven Halim
Felix Halim
HANDBOOK FOR ACM ICPC AND IOI CONTESTANTS
2013


Antti Laaksonen
Guide to Competitive Programming
Learning and Improving Algorithms Through Contests
Springer