

INF280 : Préparation aux concours de programmation

Présentation du cours et du concours ICPC

Antoine Amarilli

Programmation compétitive

- **Qu'est-ce que c'est ?**

- **Matériel** : participants individuels ou en équipe
- **Entrée** : spécification d'un problème de programmation

Alice veut tondre sa pelouse rectangulaire avec un chemin aussi court que possible qui passe par chaque case et évite des obstacles...

- **Sortie** : un programme qui répond au problème

Programmation compétitive

- **Qu'est-ce que c'est ?**

- **Matériel** : participants individuels ou en équipe
- **Entrée** : spécification d'un problème de programmation

Alice veut tondre sa pelouse rectangulaire avec un chemin aussi court que possible qui passe par chaque case et évite des obstacles...

- **Sortie** : un programme qui répond au problème

- **Pourquoi le faire ?**

- Pour le **fun** !
- Pour apprendre à mieux **programmer** !
- Pour la **gloire** ! (si on gagne...)
- Pour trouver du **travail** !

Programmation compétitive

- Qu'est-ce que c'est ?

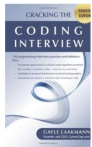
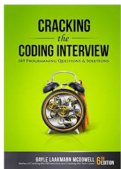
- **Matériel** : participants individuels ou en équipe
- **Entrée** : spécification d'un problème de programmation

Alice veut tondre sa pelouse rectangulaire avec un chemin aussi court que possible qui passe par chaque case et évite des obstacles...

- **Sortie** : un programme qui répond au problème

- Pourquoi le faire ?

- Pour le **fun** !
- Pour apprendre à mieux **programmer** !
- Pour la **gloire** ! (si on gagne...)
- Pour trouver du **travail** !



- Préparation au concours de programmation **ICPC**

- Préparation au concours de programmation **ICPC**
- **Déroulement des séances :**
 - Bref **cours magistral** sur un thème algorithmique
 - Trois **problèmes** d'annales ICPC donnés sur un concours sur le site **Virtual Judge**
 - À soumettre sur **Virtual Judge** au plus tard **la veille** de la séance suivante

- Préparation au concours de programmation **ICPC**
- **Déroulement des séances :**
 - Bref **cours magistral** sur un thème algorithmique
 - Trois **problèmes** d'annales ICPC donnés sur un concours sur le site **Virtual Judge**
 - À soumettre sur **Virtual Judge** au plus tard **la veille** de la séance suivante
- **Évaluation :** moyenne de deux notes
 - **Contrôle continu :** exercices à rendre (noté sur 21)
 - **Concours de programmation interne :** épreuve sur machine, présence obligatoire (jeudi 18 juin aprèm)

- Préparation au concours de programmation **ICPC**
- **Déroulement des séances :**
 - Bref **cours magistral** sur un thème algorithmique
 - Trois **problèmes** d'annales ICPC donnés sur un concours sur le site **Virtual Judge**
 - À soumettre sur **Virtual Judge** au plus tard **la veille** de la séance suivante
- **Évaluation :** moyenne de deux notes
 - **Contrôle continu :** exercices à rendre (noté sur 21)
 - **Concours de programmation interne :** épreuve sur machine, présence obligatoire (jeudi 18 juin aprèm)
- Langage : **C++** uniquement, i.e., C avec STL

- **ICPC** : International Collegiate Programming Contest

Concours ICPC

- **ICPC** : International Collegiate Programming Contest
- **Collegiate** : les candidats (étudiants) sont organisés en **équipe** (de 3 membres) qui viennent tous de la même université

Concours ICPC

- **ICPC** : International Collegiate Programming Contest
- **Collegiate** : les candidats (étudiants) sont organisés en **équipe** (de 3 membres) qui viennent tous de la même université
- **International** : équipes du monde entier, compétition en **deux phases** (régionale et mondiale)

- **ICPC** : International Collegiate Programming Contest
- **Collegiate** : les candidats (étudiants) sont organisés en **équipe** (de 3 membres) qui viennent tous de la même université
- **International** : équipes du monde entier, compétition en **deux phases** (régionale et mondiale)
- **SWERC** : Phase régionale de l'Europe du sud-ouest. Télécom ParisTech participe depuis 2013. À Télécom en **2017-2019**.
<http://swerc.up.pt/2014/reports/ranking.html> (13e et 33e sur 49)
<http://swerc.up.pt/2015/reports/ranking.html> (14e et 45e sur 52)
<http://swerc.up.pt/2016/reports/ranking.html> (20e et 36e sur 60)
<https://swerc.eu/2017/theme/results/official/public/> (30e et 42e sur 75)
<https://swerc.eu/2018/theme/scoreboard/public/> (18e et 69e sur 89)
<https://swerc.eu/2019/theme/scoreboard/public/> (30e et 48e sur 95)

- **Équipes de 3 candidats** pour chaque école ou université
- Télécom y envoie **deux équipes** d'étudiants sélectionnés
- Les participants doivent être **inscrits pédagogiquement** à Télécom au moment du SWERC en 2020–2021.
- Les quelques meilleures équipes du SWERC participeront à la **finale mondiale** en 2021

Règles du concours SWERC

- Temps limité, généralement à **5 heures**
- Une **dizaine** de problèmes de programmation à résoudre
- **Un seul ordinateur** par équipe!
- Langages de programmation : **C, C++ ou Java**
- **Pas d'accès à Internet!**
- Accès aux **documentations des langages** (JavaDoc, doc STL)
- Seul matériel autorisé : **antisèche de 25 pages** imprimées A4

Détails :

- <https://swerc.eu/2019/regulations/>
- <http://icpc.baylor.edu/worldfinals/rules>
- <http://icpc.baylor.edu/worldfinals/programming-environment>

Règles du concours interne

- Temps limité, **3 heures** (plus préparation et résultats)
- Environ **six** problèmes de programmation à résoudre
- **Concours individuel**
- Langage de programmation : **C, C++ ou Java**
- **Pas d'accès à Internet**, excepté le juge en ligne, la documentation des langages, et parfois Wikipédia.
- **Antisèche de 25 pages** autorisée

Date : **le jeudi 18 juin 2020 de 13h15 à 17h.**

Format des problèmes

- Description en anglais d'un **problème concret** à résoudre
- Description du **format** des entrées et sorties
- **Exemple** d'entrée et de sortie correspondante fournie
- Le programme à implémenter prend sur son **entrée standard** (`stdin`, `cin`) une instance du problème et doit produire sur sa **sortie standard** (`stdout`, `cout`) la sortie correspondante

Soumettre un programme

- Soumission du code source sur une **interface Web**

Soumettre un programme

- Soumission du code source sur une **interface Web**
- Évaluation **automatique** sur des entrées secrètes.

Soumettre un programme

- Soumission du code source sur une **interface Web**
- Évaluation **automatique** sur des entrées secrètes.
- Compilation, édition de liens, exécution **sur le serveur de test**

Soumettre un programme

- Soumission du code source sur une **interface Web**
- Évaluation **automatique** sur des entrées secrètes.
- Compilation, édition de liens, exécution **sur le serveur de test**
- Temps d'exécution et mémoire disponible **limités** (e.g., quelques secondes, quelques méga-octets)

Soumettre un programme

- Soumission du code source sur une **interface Web**
- Évaluation **automatique** sur des entrées secrètes.
- Compilation, édition de liens, exécution **sur le serveur de test**
- Temps d'exécution et mémoire disponible **limités** (e.g., quelques secondes, quelques méga-octets)
- **Verdict** sur la soumission, les plus courants sont :
 - **Accepted** : La soumission passe les tests
 - **Time limit exceeded** : Trop lent (ou boucle infinie, bug...)
 - **Runtime error** : Erreur d'exécution (segfault, etc.)
 - **Wrong answer** : Résultats faux
 - **Presentation error** : (parfois) Mauvais formatage des résultats
 - **Memory limit exceeded, Compilation error**, etc.

- **Objectif** : résoudre le plus de problèmes le plus vite possible

Scores et stratégies

- **Objectif** : résoudre le plus de problèmes le plus vite possible
- **Aucun point** si la soumission n'est pas parfaite (**Accepted**)
(sauf dans le contrôle continu et concours interne)

- **Objectif** : résoudre le plus de problèmes le plus vite possible
- **Aucun point** si la soumission n'est pas parfaite (**Accepted**)
(sauf dans le contrôle continu et concours interne)
- **Temps de pénalité** pour les soumissions fausses
(sauf dans le contrôle continu)

Scores et stratégies

- **Objectif** : résoudre le plus de problèmes le plus vite possible
- **Aucun point** si la soumission n'est pas parfaite (**Accepted**)
(sauf dans le contrôle continu et concours interne)
- **Temps de pénalité** pour les soumissions fausses
(sauf dans le contrôle continu)
- **Stratégie** :
 - attaquer les **problèmes faciles** d'abord
(problèmes dans un ordre aléatoire, sauf dans le contrôle continu)
 - ne pas **rester bloqué** en cas de bug

- Le but est de produire **vite** du code qui **marche**

Style de développement

- Le but est de produire **vite** du code qui **marche**
- Inutile d'être **réutilisable**, et nécessité d'être **concis** :
 - Pas besoin de **commentaires**
 - **Un seul fichier** par problème
 - Pas de programmation **orientée objet** (sauf STL)
 - Pas de **vérification des entrées**, sécurité mémoire, etc.

Style de développement

- Le but est de produire **vite** du code qui **marche**
- Inutile d'être **réutilisable**, et nécessité d'être **concis** :
 - Pas besoin de **commentaires**
 - **Un seul fichier** par problème
 - Pas de programmation **orientée objet** (sauf STL)
 - Pas de **vérification des entrées**, sécurité mémoire, etc.
- Mais il ne faut pas faire d'**erreur** :
 - Noms de variables **brefs** mais **logiques**
 - Conserver l'**indentation**

Traiter un problème

- Souvent un **algorithme classique** caché derrière le programme (tri, graphe, géométrie, file de priorité, etc.; cf le cours)
- Réfléchir **sur papier** à l'algorithme et au pseudo-code :
Ne pas se précipiter pour coder!
- Tester sur l'**exemple fourni** et autres exemples simples
- Faire attention au **format de sortie** (retours à la ligne, etc.)
- Tester sur le juge en ligne, trouver les bugs s'il y en a...

Pour le contrôle continu...

- **uDebug : AUTORISÉ**
 - Vous fournissez un **fichier d'entrée**
 - Il fournit la **sortie correcte**
 - Permet de chercher les **bugs** dans votre programme

Pour le contrôle continu...

- **uDebug : AUTORISÉ**

- Vous fournissez un **fichier d'entrée**
- Il fournit la **sortie correcte**
- Permet de chercher les **bugs** dans votre programme

- **Forums, blogs, etc. : TOLÉRÉ**

- Si vous êtes bloqué, il y a souvent des éléments de solutions, idées (parfois fausses...) sur Internet
- C'est OK aussi de **discuter entre vous** à propos du problème

Pour le contrôle continu...

- **uDebug : AUTORISÉ**
 - Vous fournissez un **fichier d'entrée**
 - Il fournit la **sortie correcte**
 - Permet de chercher les **bugs** dans votre programme
- **Forums, blogs, etc. : TOLÉRÉ**
 - Si vous êtes bloqué, il y a souvent des éléments de solutions, idées (parfois fausses...) sur Internet
 - C'est OK aussi de **discuter entre vous** à propos du problème
- **Plagiat, copie de solutions en ligne : INTERDIT**
 - **Interdit** de recopier le code d'une solution en ligne!
 - **Interdit** de recopier le code de vos camarades!
 - Tout **emprunt** d'un algorithme classique (Dijkstra, etc.) doit être **clairement indiqué** avec l'URL de sa source en commentaire

Exemple de résolution de problème

S'inscrire sur Virtual Judge et traiter le problème “Identifying tea” :

<https://vjudge.net/contest/355192>

- Version initiale de ces transparents par Pierre Senellart.
- Transparent 2 : Google Images, droit de citation.