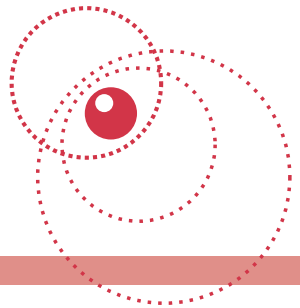

Entraînement au concours ACM-ICPC

Concours ACM-ICPC : format et stratégies

12 mai 2014

Pierre Senellart

TELECOM ParisTech

Plan

Qu'est-ce que c'est ?

- ACM-ICPC : International Collegiate Programming Contest de l'ACM
- ACM : Association for Computing Machinery, principale société savante en informatique ; organise des conférences, délivre des prix, publie des journaux, etc.
- 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)
- Concours prestigieux, auquel participent un très grand nombre d'universités et écoles. Première participation de Télécom ParisTech l'an dernier. `http://tinyurl.com/mc3k9s4`

Phases du concours

- 2 (ou parfois 3 voire 4) équipes de 3 candidats par école ou université ; à Télécom, sélection interne en fin de P4
- Compétition régionale (Southwestern Europe : ouest de l'Autriche, Espagne, France, Italie, Portugal, Suisse) en novembre (week-end), inscription en septembre
- La meilleure équipe (ou parfois les deux meilleurs équipes) de chaque région participent à la finale mondiale en juin – juillet de l'année suivante
- Télécom prend en charge les frais d'inscription, d'hébergement, de transport
- Les participants doivent être inscrits pédagogiquement à Télécom au moment du concours régional. La plupart des 3e année au moment de la compétition interne donc non éligibles :-(. Limites (peu contraignantes) d'âge et années après bac.

Règles du concours

- Temps limité (en général, 5 heures)
- Un nombre de problèmes de programmation fixé à résoudre (en général, une dizaine)
- Un seul ordinateur par équipe !
- Langages de programmation : C, C++ ou Java, version des compilateurs utilisées mentionnée
- Pas d'accès à Internet !
- Accès aux documentations des langages (JavaDoc, doc STL)
- Seul matériel autorisé : antisèche de 25 pages imprimées A4 (en trois exemplaires) par équipe
- cf `http://swerc.dsic.upv.es/rules.php`
  `http://icpc.baylor.edu/worldfinals/rules`
  `http://icpc.baylor.edu/worldfinals/programming-environment`

Format des problèmes

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

Soumettre un programme

- Entrée(s) sur lequel le programme sera testé gardée(s) secrète(s)
- Interface (Web ou ligne de commande) automatique d'évaluation
- Soumission du code source du programme
- Compilation, édition de liens, exécution sur le serveur de test
- Temps d'exécution et mémoire disponible limitées (e.g., quelques secondes, quelques méga-octets ou dizaine de méga-octets), contraintes parfois explicites, parfois non
- Notification quasi-instantanée de si la soumission est correcte ou non
- De tels serveurs librement disponibles pour s'entraîner : `https://icpcarchive.ecs.baylor.edu/`, `http://uva.onlinejudge.org/`

Score

- Critère le plus important : nombre de problèmes résolus dans la durée du concours
- Si égalité : somme, pour chaque problème résolu, du temps écoulé entre le début du concours et la soumission correcte. Pénalité de 20 minutes par soumission incorrecte (uniquement sur les problèmes finalement résolus)
- cf. `http://swerc.dsic.upv.es/resultados/public/index.html`

Plan

Présentation

Stratégies de base

Choix du langage

- Choix le plus courant : C/C++ impératif avec usage des collections de la STL
- Java impératif : possible, mais souvent moins rapide à écrire, et moins efficace à l'exécution. Bibliothèque standard plus riche
- C pur : pas d'intérêt
- C++ ou Java purement orienté objet (classes, héritage, interfaces, polymorphisme, etc.) : probablement pas assez de temps pour produire quelque chose d'utile dans le temps imparti – mais les exceptions existent !

Style de développement

- Le but n'est pas de produire du code réutilisable mais du code qui marche, et vite
- On oublie les principes de base de développement logiciel : commentaires, séparation du code en fichiers, organisation en classes multiples, héritage, accesseurs, etc.
- Mais on reste raisonnable : conserver des noms de variable explicites, une indentation correcte, une séparation en fonctions pour les parties les plus indépendantes. . . sinon relecture impossible même pendant la durée du concours
- Utiliser l'éditeur ou IDE (parmi ceux disponibles) avec lequel vous êtes le plus efficace, ne pas perdre de temps avec ça
- Compiler souvent, pour éviter les erreurs de compilation incompréhensibles, et tester petit à petit
- Mais, au final, l'algorithmique et les structures de données utilisées sont plus importantes que le code lui-même. . .

Que mettre dans l'antisèche ?

- Rappel des algorithmes et des structures de données classiques : parcours de graphe, tri, arbres de recherche, hachage, programmation dynamique, automates, etc. Résumé ultra-condensé d'un livre de cours d'algorithmique.
- Rappel de résultats mathématiques utiles : arithmétique, méthodes de résolution d'équations, etc.
- Mini-référence du langage et de sa bibliothèque
- Mini-référence de l'éditeur ou IDE

Choisir les problèmes

- Pas d'ordre particulier dans la présentation des problèmes d'un sujet
- Regarder rapidement tous les sujets pour identifier les plus faciles (et les plus difficiles)
- Surveiller les sujets traités par les autres équipes (si connus) pour repérer les plus faciles
- Paralléliser entre équipiers (voir plus loin)

Traiter un problème

- Identifier si le problème peut se ramener à un algorithme classique ou à l'utilisation d'une structure de donnée classique
- Rédiger les grandes lignes du pseudo-code
- Ne pas bloquer trop longtemps, éventuellement passer à un autre
- Coder
- Tester sur l'exemple donné
- Tester sur d'autres exemples – marcher sur un exemple n'est souvent pas suffisant
- Compromis entre soumettre tout de suite (et risquer 20 minutes de pénalité) et attendre d'avoir mieux testé

Stratégie d'équipe

- Plusieurs stratégies possibles :
  - Un code sur l'ordinateur, un résout les problèmes sur pseudo-code, le troisième fait l'interface
  - Partage équitable des problèmes entre les trois équipiers, une fois l'ordinateur disponible, un de ceux ayant fini d'élaborer le pseudo-code s'y met
  - N'importe où entre les deux
- Utiliser au mieux les compétences de chacun : même un non programmeur peut être utile dans une équipe
- Ne jamais être inactif : relecture du code au fur et à mesure qu'il est tapé, correction du code imprimé (si possible), réflexion sur d'autres problèmes, rédaction de pseudo-code voire de fragments de code sur papier

Optimisation du score

- Critère le plus important : nombre de problèmes résolus. Paralléliser au maximum !
- Traiter en priorité les problèmes faciles
- 10 minutes de test pour débuguer, c'est dix minutes de gagnées par rapport à une soumission ratée pénalisée
- On n'apprend rien d'autre d'une soumission que Correct/Incorrect. Beaucoup moins utile pour tester un programme que des tests en local
- On ne perd rien à faire des soumissions désespérées sur les problèmes non encore résolus dans les dernières minutes... pas de pénalité dans ce cas
- Si public, surveiller ce qu'ont fait les autres équipes

Licence de droits d'usage

Contexte public } avec modifications

Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après et à l'exclusion expresse de toute utilisation commerciale.
Le droit d'usage défini par la licence autorise un usage à destination de tout public qui comprend :
– le droit de reproduire tout ou partie du document sur support informatique ou papier,
– le droit de diffuser tout ou partie du document au public sur support papier ou informatique, y compris par la mise à la disposition du public sur un réseau numérique,
– le droit de modifier la forme ou la présentation du document,
– le droit d'intégrer tout ou partie du document dans un document composite et de le diffuser dans ce nouveau document, à condition que :
    – L'auteur soit informé.

Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel et non exclusif.
Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur : sitepedago@telecom-paristech.fr