Colle 13 MP2I

Le travail se fait au tableau ou sur machine. Les étudiant.s.es doivent apporter une machine tournant sur Linux.

Un OCaml doit également être utilisable (éventuellement en ligne avec BetterOcaml).

Thèmes
- Complexité suites « diviser pour régner » ; Complexité en moyenne. Complexité amortie.
- Récursion terminale.
- OCaml traits impératifs.

Références
- Complexité.
- Terminaison et correction.
- Récursion terminale
- Complexité en moyenne et amortie
- OCAml : traits impératifs dont entrées-sorties et ligne de commande.

Cours Au tableau.

1. Fonction `tri_rapide` : Donner le code p 11 puis :
   (a) Difficulté faible : Donner le code de `partition`, justifier sa complexité (p7) et étudier la complexité de `tri_rapide l` lorsque ℓ est triée (p13).
   (b) Difficulté moyenne : donner la preuve de correction (par récurrence) du tri rapide (p12) en admettant que `partition` termine et est correct.
   (c) Difficulté élevée : montrer que la complexité T_n correspondant à l'appel `tri_rapide l` lorsque $|\ell| = n$ et ℓ est triée est la pire qu'on puisse avoir pour le tri rapide d'une liste de taille n (p14-15).
   (d) Difficulté élevée : complexité en moyenne du tri rapide (p16-18). Pour tous les étudiant.e.s volontaires mais Kosman, Gil, Laulan, Monnier, Millot, Paya, Trentin, Metayer, Pailhes et Lorne ne peuvent refuser cette preuve si on la leur demande.
2. Théorèe d'amortissement : preuve et corollaire p27-29.
3. Complexité amorti du compteur binaire p30-31.

Exercices Quelques thèmes (liste non exhaustive) :

- OCaml :Tous aspects de OCaml y compris types définis par l'utilisateur, entrées-sorties et passage d'arguments en ligne de commande (la.e colleu.se.r rappelera les interfaces pour les entrées-sorties et la ligne de commande).

— Complexité des algorithmes. Les étudiant.es.s doivent pouvoir fournir l'équation de récurrence de la complexité.

  Aucun exercice sur la complexité spatiale n'a été abordé. mais la.e colleu.se.r est libre de faire prendre conscience de l'accumulation des paramètres intermédiaires en cas de récursion non terminale.

  Complexité en moyenne ou amortie si l'étudiant.e est guidé.

Une obligation pour l'étudiant.e : **À toute fonction, sa fonction de tests**.

Dans tous les cas, si l'étudiant.e n'a pas le temps de terminer son code, une présentation de son algorithme et des principaux cas à traiter sera appréciée.

Déroulement Une question de cours ; entre 2 et 4 exercices de difficultés croissantes. Si la question de cours n'est pas maîtrisée, la moyenne n'est pas accordée.