Commentaires DS4

Remarques générales :

- objectif concours : marquer des points !
- certaines questions doivent commencer à vous sembler "classique", voire à être carrément "du cours" (d'où l'intérêt de réviser !!). Surtout, ne pas chercher à innover !
- il y a des choses classiques dans toutes les parties : penser à la gestion du temps !
- penser à regarder l'annexe : ça donne des pistes pour la programmation ! (List.iter, Liste.rev)
- beaucoup d'erreurs dans les complexités à connaître ou à calculer
- les questions qui demandent une démonstration doivent être traitées comme telles (récurrence, implication, raisonnement par l'absurde ... voir corrigé), pas simplement en indiquant une vague idée.

Rédiger une démonstration, ça se travaille !

Les erreurs courantes sont :

- erreur sur les bases de la logique (confondre une implication et une réciproque ...)
- erreur sur les techniques de démonstration (mise en place d'une récurrence, d'un raisonnement par l'absurde)
- rédaction maladroite ou incompréhensible
- erreur sur l'appropriation des notions qui sont à utiliser dans le raisonnement
- manque d'esprit critique : tendance à affirmer des propositions fausses.

Pour savoir démontrer, il faut bien connaître son cours (définitions, propriétés ...) pour ne pas affirmer de choses fausses, et acquérir un savoir-faire : comprendre une démonstration, ça n'est pas savoir démontrer

Pour démontrer, il faut être capable, de manière totalement autonome, de :

- trouver une stratégie de démonstration (il y en a souvent plusieurs)
- mettre en place la structure de la démonstration : quel est le point de départ ? quel est le point à atteindre ?
- raisonner correctement du début jusqu'à la conclusion

Tout ceci peut se faire en quelques secondes pour les questions les plus faciles (ou pour les plus aguerris), ou peut demander un travail de recherche au brouillon.

Parfois il faut un peu d'intuition, mais 90% des démonstrations reprennent des idées déjà vues en cours

Lorsqu'il y a une démonstration à connaître (ce qui n'est pas vraiment le cas en informatique), on ne la retient pas telle quelle : on retient la stratégie de démonstration, et le ou les arguments clé. Le savoir-faire fait le reste !

Généralités

Q1. OK

Q2. propriétés "classiques", mais plusieurs démonstrations possibles (aucune n'est facile ...). Au minimum il faut penser à une récurrence sur le nombre de sommets ou le nombre d'arêtes. Attention aux fausses évidences : retirer un sommet quelconque d'un graphe connexe ne le laisse pas forcément connexe

Q3. "classique" et beaucoup plus facile. Il y a au minimum 4 implications à démontrer

Q4. Q5. moins classique, de difficulté moyenne

Q6. Q7. Q8. tri_fusion en OCaml. "classique" : essayer de faire simple (sans opération coûteuse comme List.length ou @, sans fonction auxiliaire ...). Dans le tri fusion, la liste à un élément est un cas de base (indispensable sinon appels infinis)

Algorithme de Kruskal inversé

Q9. Cours

Q10. OK

Q11. tri_aretes. il faut se creuser un peu la tête pour trouver une solution élégante ... il faut passer de couples à des triplets + attention aux arêtes en double dans le graphe + utiliser tri_fusion et List.rev

Q12. connectes. C'est un parcours dans lequel on ne peut suivre que certaines arêtes. Ne pas inventer un parcours ! Faire au plus simple : un parcours en profondeur implémenté récursivement avec un tableau des sommets vus.

Q13. kruskal_inverse. il faut se creuser un peu la tête pour trouver une solution élégante ... retirer une arête du graphe n'est pas si difficiles. ou à l'inverse ajouter les arêtes sélectionnés dans un graphe vide

Q14. démonstration assez facile (mais rarement convaincante)

Q15. question difficiles

Difficulté de l'arbre optimal

Q17. notion d'arbre de Steiner, sommets de Steiner, qu'il faut comprendre correctement

Q18. notion de stable (classique). Ne pas pouvoir rajouter de sommet à un stable n'en fait pas un stable de cardinal maximal.

Q19. Q20. Q21. transformation assez classique (et déjà vue en exercice). Les raisonnements sont très précis

Q22. Q23. transformation moins classique, mais raisonnement tout aussi précis

Q24. C'est la démarche habituelle : NP + NP difficile (par réduction)

Approximation du problème

Q25 programmation assez facile

Q26 à Q30. Floyd-Warhall (cours). le rôle du tableau pred n'est pas très clair dans l'énoncé

Q31. Algorithme assez difficile à appréhender ... mais la réponse est simple

Q32. Facile

Q33. Difficile

Q34. Facile

Q35. Q36. Long et difficile !