À SAVOIR POUR LES CONCOURS

A Algorithmes à savoir écrire en OCaml en moins de 5 minutes

- recherche du minimum d'un tableau,
- algorithme d'Euclide,
- produit matriciel naïf,
- évaluation de polynôme via la méthode de Horner,
- exponentiation rapide,
- décomposition en base b,
- suite récurrente linéaire en temps constant (impératif et récursif) par exemple Fibonacci,
- insertion d'un élément dans une liste triée,
- tri par insertion,
- coder le miroir d'une liste en temps linéaire en sa taille,
- recherche par dichotomie dans un tableau trié,
- calculer la taille et la hauteur d'un arbre.

B Algorithmes classiques à savoir implémenter en 10 minutes

- tri rapide,
- tri fusion,
- parcours en largeur et en profondeur d'un graphe,
- parcours préfixe, infixe, postfixe d'un arbre binaire,
- recherche, insertion et suppression dans un ABR,
- création, extraction et ajout dans un tas min ou tas max,
- tri par tas,
- les opérations sur une file de priorités implémentées par un tas,
- les opérations sur une structure union-find.

C Algorithmes dont la connaissance aide à finir plus vite

- algorithme de Quine,
- algorithme de Huffman,
- algorithme de Floyd-Warshall,
- algorithme de Dijkstra,
- algorithme A*,
- algorithme de Kruskal,
- algorithme de Prim,
- tri topologique,
- détection de cycles dans un graphe orienté,
- calcul de couplage maximum dans graphe biparti (chemin augmentant),
- les algorithmes classiques de programmation dynamique : rendu de monnaie, plus longue sous séquence commune, distance d'édition, sac à dos.

D Automates et langages réguliers

Les éléments théoriques à connaître :

- définition inductive des mots,
- lemme de Lévi,
- définition inductive des expressions régulières,
- sémantique des expressions régulières,
- définition inductives des langages réguliers,
- théorème de Kleene,
- stabilité des langages réguliers (union, concaténation, fermeture de Kleene),
- stabilité des automates finis (intersection (produit), complémentation),
- constructions de Thompson,
- lemme de l'étoile.

À savoir faire :

1. compléter, complémenter ou émonder un automate,
2. construire un automate pour l'union, le complémentaire, l'intersection, la concaténation,
3. exhiber un automate ou une expression régulière associés à un langage simple,
4. déterminiser un automate fini non déterministe (via l'automate des parties),
5. trouver l'automate correspondant à une expression régulière en utilisant l'algorithme de Berry-Sethi (automate local, automate de Glushkov),

6. trouver l'expression régulière correspondant à un automate par élimination des états (automates généralisés),
7. éliminer les ε-transitions d'un automate,
8. utiliser le lemme de l'étoile pour montrer qu'un langage n'est pas rationnel,

E Arbres et graphes

- maîtriser les différentes représentations des graphes et leurs intérêts (listes et matrices d'adjacence),
- connaître le lien entre les puissances de la matrice d'adjacence et l'existence de chemins dans un graphe,
- savoir démontrer la correction et la terminaison du parcours en largeur d'un graphe,
- connaître et savoir montrer les différentes caractérisations d'un arbre,
- savoir montrer les encadrements classiques entre hauteur et taille d'un arbre binaire,
- savoir transformer un arbre d'arité quelconque en arbre binaire.

F Logique

- savoir modéliser des propositions en langage naturel par des propositions logiques,
- savoir établir rapidement une table de vérité,
- savoir mettre une formule sous forme normale disjonctive ou conjonctive, d'après un table de vérité ou en calculant,
- connaître les lois de la logique des propositions,
- savoir évaluer une formule logique étant donnée une valuation entière binaire,
- maîtriser le vocabulaire : satisfaisable, tautologie, antilogie, valuation, modèle, équivalence, conséquence sémantique, séquent, prémisse, conclusion, règle d'inférence.
- connaître les règles de la déduction naturelle et s'entraîner à les appliquer sur des exemples.

G Techniques algorithmiques

- Savoir calculer la complexité d'un programme impératif,
- Savoir calculer la complexité d'un programme récursif d'après la relation de récurrence,
- Connaître les complexités associées aux relations :
  - $T(n) = 1 - T(n-1)$
  - $T(n) = 1 + T(n/2)$
  - $T(n) = n + 2C(n/2)$
  - $T(n) = 1 + 2T(n-1)$
- savoir démontrer la terminaison d'un algorithme impératif ou récursif,
- savoir démontrer la correction d'un algorithme,

H Techniques mathématiques

Savoir démontrer une propriété :

- par récurrence,
- par induction structurelle,
- par double inclusion,
- par contraposition,
- par l'absurde,
- par analyse-synthèse.