1 Généralités

1. Que signifient langages compilés et langages interprétés ?
2. Savoir faire un graphe de flot de contrôle pour un programme simple et identifier un jeu de test par couverture des sommets et par couverture des arêtes (branches, chemin). (Notion facile à connaitre impérativement)
3. Savoir calculer la complexité d'un programme récursif (c'est-à-dire trouver la relation de récurrence puis la résoudre). Exemples typiques : $C_n = \alpha C_{n-1}$; $C_n = C_{\frac{n}{2}}$; $C_n = 2C_{\frac{n}{2}} + O(n)$.
4. Bien revoir les manipulations de pointeurs en C.
5. Connaitre les différentes parties de la mémoire et leurs rôles
6. bonne utilisation de malloc et free (il sera attendu que vous pensiez à libérer tout ce que vous allouez)
7. vérifier que vous connaissez toutes les commandes en C et en OCAML exigibles par le programme (annexe du programme en pièce jointe)

2 Des algos très simples que vous devez savoir écrire en C et en OCaml en moins d'une minute

8. Recherche du minimum dans un tableau.
9. Algorithme d'Euclide.
10. Horner.
11. Exponentiation rapide.
12. Décomposition en base b.
13. Suite récurrente linéaire avec autant de variables de sauvegarde que l'ordre de la relation de récurrence.
14. Tri bulle.
15. Insérer un élément dans une liste triée (en OCAML et en C pour une liste chainée codée avec des pointeurs).
16. Recherche par dichotomie dans un tableau.
17. parcours de graphe binaire en profondeur.
18. compter le nombre de noeuds ou la hauteur d'un arbre par forcément binaire.
19. un produit matriciel
20. Calculer le miroir d'une liste (en OCAML)

3 Algos classiques et au programme à savoir implémenter rapidement

21. tri rapide (de préférence en place)
22. tri fusion
23. parcours en largeur
24. parcours en profondeur
25. Dijkstra
26. Kosaraju
27. TRI TOPOLOGIQUE!!!
28. trouver les CC d'un graphe non orienté à l'aide d'un parcours (en profondeur par exemple)
29. algo min max avec élagage alpha-beta

4 Algos au programme dont il faut connaitre par coeur le principe et savoir dérouler sur un exemple

30. Floyd Warshall
31. Prim
32. Kruskal
33. Kosaraju
34. tri par tas
35. algorithme de Quine
36. Sélection d'activités (algo glouton)
37. construction de l'arbre de Huffman
38. Liv Zempel Welch
39. Boyer Moore (première optimisation seulement, oubliez la seconde !)
40. Rabin Karp
41. détection de cycles dans un graphe non orienté
42. A*
43. couplage maximal dans un graphe biparti

5 Programmation dynamique : exemples à retravailler càd savoir reconstruire la relation de récurrence et remplir un tableau à la main et écrire le code

44. Plus longue sous séquence commune
45. distance d'édition
46. Sac à dos
47. Cocke Younger Kasami
48. Floyd Warshall
49. produit matriciel

6 Jeux ?

7 IA ?

8 Algos gloutons

9 Piles et Files

50. Savoir implémenter toutes les opérations de file et de pile sur une structure de liste chainée en C.
51. tout connaitre su union find notamment l'implémentation

10 Algo probabilistes et d'approximation

52. definitions de Las Vegas, Monte Carlo, algo d'approximation (exemples que je recommande de connaitre -> tri rapide, kth element,
53. technique du backtracking (pb des N reines à connaitre avec la version Las Vegas)
54. technique branch and bound
55. dernaodmisation de l'akgo probabiliste de MAXSAT
56. 2-approximation du problème de couverture de sommets

11 Arbres

57. Savoir montrer le lien combinatoire entre hauteur d'un arbre et nombre de noeuds.
58. Connaitre et savoir montrer le nombre de feuilles en fonction du nombre de noeuds dans un arbre binaire strict.
59. savoir utiliser une file pour faire un parcours en largeur de graphe
60. définition de tas + représentation dans un tableau, insertio, suppression, construction, nm de feuilles, tri par tas.
61. insertion, recherche et suppression dans un arbre binaire de recherche : coût et intérêt de l'équilibrage.
62. définition d'un arbre rouge-noir , estimation de sa hauteur.
63. savoir comment fonctionne l'insertion dans un arbre rouge noir dans les grandes lignes.
64. transformation d'un arbre d'arité quelconque en arbre binaire (LCRS)

12 Logique

65. Savoir écrire un programme qui évalue une formule logique étant donné une valuation.
66. Connaitre le vocabulaire : satisfiable, tautologie, antilogie, modèle, formules équivalentes, conséquence sémantique,
67. Lois de Morgan
68. Table de vérité
69. Formes normales et méthode de mis en forme normale
70. Connaitre les règles de la déduction naturelle et s'entrainer sur les exemples vus en classe.

13 Algo des graphes

71. lemme de Konig
72. bien connaitre les différentes représentation de graphes et leur intérets.
73. connaitre le lien entre les puissances de la matrice d'adjacence et l'existence de chemins dans un graphe.
74. équivalences entre les différentes caractérisations des arbres.
75. connaitre et savoir montrer le lien entre le nombre d'arêtes et la somme des degrés
76. connaitre le lien entre nb de sommets et nb d'arêtes dans un arbre
77. Savoir justifier formellement l'existence d'un arbre couvrant dans un graphe connexe.
78. Savoir prouver qu'une arête sure est nécessairement une arête d'un arbre couvrant de poids minimal.
79. savoir quelle méthode permet de trouver un plus court chemin (parcours en largeur si on n'a pas de pondération, FW versus Dijkstra)
80. savoir prouver la validité du tri topologique

14 Calculabilité

81. Savoir montrer que le problème de l'arrêt est indécidable : se concentrer sur la version plus "haut niveau" plutIot que sur le code en C qui finalement n'est pas si intuitif.
82. comprendre l'importance du rôle de la machine universelle

15 Grammaires non contextuelles

83. Savoir montrer par double inclusion chacune nécessitant une récurrence propre le langage généré par une grammaire.
84. Savoir montrer qu'un langage reconnaissable est context-free.
85. Savoir trouver une arbre d'analyse.
86. Notion de grammaire ambigüe, dérivations gauches et droites.

16 Automates et langages rationnels

87. connaitre la définition inductive d'un langage régulier et savoir le distinguer d'une expression réguière
88. savoir utiliser le lemme de l'étoile
89. savoir éliminer les epsilon transitions
90. savoir compléter, émonder, standardiser un automate
91. savoir construire l'automate pour le complémentaire, l'union, la concaténation, l'intersection (automate produit)
92. savoir déterminiser un AFND.
93. savoir construire l'automate de Glushkov
94. connaitre le théorème de Kleene
95. savoir appliquer la méthode d'élimination des états.

17 Complexité

96. Définition d'une réduction polynomiale
97. Savoir montrer qu'un problème est dans la classe NP.
98. Savoir transformer un problème d'optimisation en un problème de décision à seuil
99. connaitre la définition de NP-complet (et notamment de NP-dur mais aussi de NP)
100. Savoir montrer que si $f_1 \leq_P f_2$ et que $f_2 \in P$ alors $f_1 \in P$; savoir montrer que $P \subset NP$.

18 Concurrence

101. savoir ce qu'est un fil d'éxécution
102. connaitre parfaitement l'algorithme de Peterson
103. connaitre le principe de l'algo de la boulangerie de Lamport
104. bien connaitre les différents enjeux
105. reprendre le sujet 0 des mines et le sujet de DS sur le problème lecteurs-rédacteurs pour s'entrainer sur les problèmes de concurrences et les différentes exécutions possibles d'un programme dues à l'entrelacement.
106. connaitre le rôle des mutex et des sémaphores
107. connaitre le diner des philosophes et producteur/consommateur

19 Bases de données

108. Tout revoir car je n'ai rien fait de trop !
109. Bien comprendre la différence entre WHERE et HAVING

20 Pas explicitement au programme mais je recommande pour ceux à qui il restera du temps

110. Analyse amortie des complexité des opérations sur les tableaux dynamiques (TP6) et sur les files implémentées avec deux piles (cours de sup)/.
111. couverture de segments (algo glouton)
112. caractérisation de l'existence d'un cycle eulérien
113. détection de cycles dans un graphe non orienté
114. transformée rapide de Fourier
115. savoir tirer uniformément un élément dans un ensemble de manière "online" (comme dans le TP sur les N reines)
116. l'approximation du problème de voyageur de commerce dans le cas métrique et le caractère NP-complet d'une telle approximation dans le cas général
117. essayer de s'approprier le principe de preuve de l'indécidabilité d'un problème de décision non trivial portant sur les programmes
118. travailler un maximum d'exemples de réductions polynomiales car c'est astucieux et difficile donc plus vous aurez vu d'exemples plus vous serez armés (et puis c'est marrant !)