Informatique
Automatismes Essentiels pour les Concours Ipesup

1 Structures de données

Tableau : accès $O(1)$; parcours $O(n)$

Liste chaînée : accès $O(n)$; insertion tête $O(1)$

Pile / File : opérations $O(1)$; LIFO / FIFO

1.1 Arbres

— DFS : pré / infixe / post
— BFS : par niveaux

1.2 Graphes

— BFS = plus court chemin non pondéré
— DFS = exploration profonde
— Toujours marquer les sommets visités

2 Complexitè

2.1 Ordres de grandeur

Hiérarchie fondamentale : $1 \ll \log n \ll n \ll n \log n \ll n^2 \ll 2^n$

Réflexes immédiats :

— Double boucle imbriquée $\Rightarrow O(n^2)$
— Division par 2 répétée $\Rightarrow O(\log n)$
— Parcours complet structure $\Rightarrow O(n)$
— BFS / DFS $\Rightarrow O(n+m)$

Sommes classiques :

— $\sum_{k=1}^{n} k = O(n^2)$
— $\sum_{k=1}^{n} \log k = O(n \log n)$

2.2 Notations asymptotiques

$O(g(n))$: majoration asymptotique (borne supérieure).

$o(g(n))$: négligeable devant $g(n)$: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

$\Theta(g(n))$: même ordre de grandeur (borne supérieure et inférieure).

Réflexes immédiats :

- $n \in o(n^2)$
- $n \log n \in o(n^2)$
- Si $f \in \Theta(g)$ alors même croissance dominante.

2.3 Récurrences classiques

- $T(n) = T(n-1) + 1 \Rightarrow O(n)$
- $T(n) = T(n-1) + n \Rightarrow O(n^2)$
- $T(n) = T(n/2) + 1 \Rightarrow O(\log n)$
- $T(n) = T(n/2) + 1 \Rightarrow O(\log n)$
- $T(n) = 2T(n/2) + 1 \Rightarrow O(n)$
- $T(n) = 2T(n/2) + n \Rightarrow O(n \log n)$

2.4 Récursion

Toujours identifier immédiatement :

- Profondeur pile :
  - Appel $n-1 \Rightarrow$ profondeur n
  - Division par $2 \Rightarrow$ profondeur $\log n$
- Coût total :
  - Travail constant par niveau + profondeur $n \Rightarrow O(n)$
  - Travail n par niveau + profondeur $\log n \Rightarrow O(n \log n)$
  - Deux appels taille $n/2 \Rightarrow$ souvent $O(n \log n)$
  - Deux appels taille $n-1 \Rightarrow O(2^n)$

3 Démonstration standard

3.1 Invariant de boucle

1. Initialisation
2. Conservation
3. Terminaison
4. Correction

3.2 Schémas

- Preuve de correction par invariant
- Preuve par induction sur la taille
- Preuve de terminaison par variant

4 Algorithmes

1. Recherche
   - Recherche linéaire
   - Recherche dichotomique (tableau trié)
2. Tri
   - Tri par insertion
   - Tri par sélection
   - Tri fusion
3. Parcours
   - Parcours tableau
   - Parcours arbre (DFS)
   - BFS (avec file)
   - DFS (récursif ou pile)
4. Récursifs Classiques
   - Calcul factoriel
   - Fibonacci naïf
   - Divide and conquer standard

Attention

Les démonstrations susceptibles d'être demandées à l'examen oral portent exactement sur les mêmes thèmes que ceux mentionnés dans ce PDF (complexité des algorithmes, recursion, invariants).
Cependant, après discussion avec plusieurs collègues, la manière précise dont ces démonstrations sont attendues reste encore peu claire. En d'autres termes, il n'est pas certain que l'on exige une démonstration formelle et complète des théorèmes — ce qui impliquerait de maîtriser rigoureusement chaque étape — ou bien seulement l'idée générale de la démonstration, auquel cas il suffirait de bien connaître les principales méthodes de preuve : raisonnement par l'absurde, induction (sur la taille des données ou sur des fonctions récursives) et preuve par invariant.
Afin d'éviter de vous transmettre des indications inexactes — soit en vous poussant à apprendre des démonstrations trop lourdes et peut-être inutiles, soit en vous préparant insuffisamment — je préfère approfondir la question et vous donner des informations plus précises ultérieurement.
En attendant, vous pouvez néanmoins consolider la théorie relative à tous les points présentés dans ce PDF, à condition bien sûr qu'ils aient déjà été abordés en cours. Comme vous le savez, le programme n'est pas encore terminé et certains éléments seront expliqués plus tard.