EPFL - Information, Calcul et Communication - Partie théorique

Série 3

1 Quel est le bon algorithme?

Lequel des quatre algorithmes récursifs suivants permet de calculer la somme des n premiers nombres pairs? (ex: si $n = 4$, alors la sortie doit valoir $2 + 4 + 6 + 8 = 20$). Expliquez pourquoi les autres ne fonctionnent pas.

algo1
entrée : *nombre entier positif ou nul n*
sortie : *nombre entier positif ou nul s*

Si $n = 0$
  Sortir: 0
Sortir: **algo1**$(n-2) + n$

algo2
entrée : *nombre entier positif ou nul n*
sortie : *nombre entier positif ou nul s*

Si $n = 0$
  Sortir: 0
Sortir: **algo2**$(2n-2) + 2n$

algo3
entrée : *nombre entier positif ou nul n*
sortie : *nombre entier positif ou nul s*

Si $n = 0$
  Sortir: 0
Sortir: **algo3**$(n-1) + 2n$

algo4
entrée : *nombre entier positif ou nul n*
sortie : *nombre entier positif ou nul s*

Si $n = 0$
  Sortir: 0
Sortir: $2 \times$ (**algo4**$(n-1) + n$)

2 Algorithme mystère

mystère
entrée : *liste ordonnée L de nombres entiers, n taille de la liste, x nombre entier*
sortie : *valeur binaire oui/non*

$a \longleftarrow 1$
$b \longleftarrow n$
Tant que $b \geq a$
  $j \longleftarrow \lfloor \frac{a+b}{2} \rfloor$
  Si $x = L(j)$
    Sortir : *oui*
  Sinon
    Si $L(j) < x$
      $a \longleftarrow j + 1$
    Sinon
      $b \longleftarrow j - 1$
Sortir : *non*

a) Si $L = (2, 2, 2, 2, 2, 2, 2, 2)$, $n = 8$ et $x = 1$, quelle est la sortie de l'algorithme et combien de fois la boucle "Tant que" est-elle exécutée?

b) En général, quelle est la sortie de l'algorithme et quelle est sa complexité temporelle en fonction de la taille n de la liste L? (utiliser la notation $\Theta(\cdot)$)

3 Au temps des Grecs

Proposez une version récursive de l'algorithme d'Euclide, qui calcule le plus grand diviseur commun de deux entiers naturels a et b: (*Note:* a mod b désigne le reste de la division euclidienne de a par b)

algorithme d'Euclide

entrée : *a, b deux entiers positifs*
sortie : *pgcd(a,b)*

Tant que $b \neq 0$
  $temp \longleftarrow a \bmod b$
  $a \longleftarrow b$
  $b \longleftarrow temp$
Sortir : a

4 Recherche de clé

Vous avez un carton qui contient potentiellement une clé. Sauf que le carton contient possiblement d'autres cartons imbriqués qui eux même peuvent contenir d'autres cartons etc... Pour trouver la clé, ou être sûr qu'elle n'existe pas dans le carton initial, l'algorithm récursif suivant vous est donné de manière sommaire.

recherche de clé

entrée : *carton*
sortie : *rien*

Si *je vois la clé dans le carton:*
  Sortir *(avec la clé en main)*
Sinon
  Si *le carton contient au moins un autre carton:*
    parcourir la liste de ces cartons et
    effectuer une recherche de clé dans
    chacun de ceux-ci
  Sinon
    Sortir *(sans avoir trouvé la clé)*

Ecrivez une version plus élaborée (toujours récursive) de cet algorithme dont la sortie soit une variable binaire s valant "oui" si la clé a été trouvée et "non" dans le cas contraire.

Note: Le problème est moins facile qu'il n'y paraît peut-être au premier abord !

5 Pour le plaisir: trouver une personne de confiance*

Vous dirigez une équipe de cinq personnes, mais depuis quelque temps, vous soupçonnez que deux d'entres elles (au plus) mentent *systématiquement* (sans avoir réussi à identifier lesquelles), tandis que les autres disent toujours la vérité. Votre but est d'identifier une personne de confiance dans l'équipe. Pour cela, vous ne pouvez poser que des questions du type:

(à la personne A) "Est-ce que la personne B dit toujours la vérité?"

Quelle est la meilleure stratégie (i.e., le meilleur algorithme) qui vous permettra d'atteindre votre but en posant le moins de questions possibles?

Indication: Pour vous échauffer, vous pouvez commencer à étudier la situation plus simple où l'équipe comporte trois personnes et au plus une d'elles ment systématiquement. Il est même fortement conseillé de le faire! Du résultat obtenu pour trois personnes, vous pourrez obtenir de manière *récursive* la solution du problème avec cinq personnes.