Algorithmique

Travaux dirigés : feuille 3

Objectifs pédagogiques et méthode de travail. Cette feuille comporte deux parties.

— La partie 1 "Objectif 10" propose des exercices très proches du cours. Avec les QCM en ligne (ENT), cette partie a pour objectif l'assimilation des connaissances de base. Ces notions pleinement acquises permettent d'obtenir la moyenne lors des contrôles de connaissances (contrôle continu et sessions d'examens).

— La partie 2 "Objectif 20" propose des exercices de difficulté croissante sur des notions plus avancées ou des formulations plus abstraites. Les étudiant-e-s informaticien-e-s, et les étudiant-e-s mathématicien-e-s qui souhaitent continuer au delà de la Licence, doivent impérativement traiter cette partie 2.

1 Connaissances

— recherches séquentielle et dichotomique
— récursivité
— complexité en temps et en espace
— algorithme récursifs de tri : tri fusion, tri rapide

2 Objectif 10

Exercice 1.

1. Écrire un algorithme itératif qui calcule un booléen indiquant si un tableau d'entiers **donné trié** par ordre croissant contient une valeur donnée. La recherche sera effectuée par dichotomie.
2. Écrire une fonction récursive `trouve` (c'est-à-dire son en-tête, puis l'appel puis le corps) qui retourne le booléen précédent.

Exercice 2.

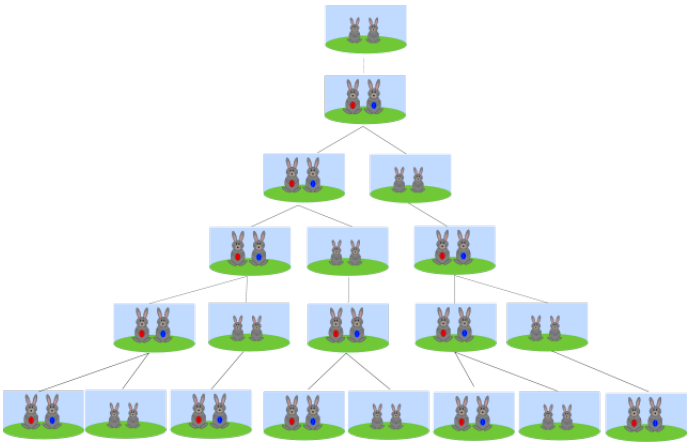
1. Dérouler l'algorithme de recherche dichotomique (version itérative) pour rechercher la valeur 4 dans la tableau [0,1,2,3,4,5,6,7,8,9].
2. Donner une valeur à rechercher qui minimise le nombre d'itérations de la recherche.
3. Même question pour celle qui maximise ce nombre.

Exercice 3. Écrire un algorithme qui effectue la recherche séquentielle d'une valeur arbitraire dans un tableau 2D de taille (n, p). On suppose que la valeur cherchée est présente dans le tableau. L'algorithme retournera la position de la première occurrence de cette valeur. Quelle est la complexité en temps de cet algorithme ? et la complexité en espace ?

Exercice 4. Écrire $s(n)$ une version récursive du calcul de la somme des n premiers entiers. Expliciter l'arbre des appels et les calculs associés lors de l'évaluation de $s(5)$. Combien d'appels récursifs à s sont nécessaires ? Que penser de la complexité de ce calcul ?

Exercice 5. On rappelle la suite de Fibonacci : $F(0) = 0, F(1) = 1$ et $F(n+1) = F(n) + F(n-1)$.

1. Calculer les premières valeurs de F pour se convaincre qu'elles augmentent rapidement.
2. Rappeler l'algorithme récursif du calcul de $F(n)$.
3. Expliciter l'arbre des appels récursifs du calcul de $F(4)$ puis de $F(5)$ — prévoir une feuille A4 en largeur. Dénombrer ces nombre d'appels.
4. Détailler l'exécution du calcul de $F(3)$ en explicitant les environnements successifs.



source : `https://commons.wikimedia.org/w/index.php?curid=11104228`

Exercice 6. On souhaite calculer la valeur de x^n où n et un entier positif, et x un flottant. On rappelle que $x^0 = 1.0$.

1. (a) Écrire un algorithme itératif qui calcule x^n.
   (b) Quelle est la complexité en temps de cet algorithme ?
2. (a) Utiliser la propriété $x^p = x \times x^{p-1}$ pour écrire un algorithme récursif qui calcule x^n.
   (b) Utiliser la propriété $x^p \times x^p = x^{2p}$ pour écrire un algorithme récursif qui calcule x^n.
   (c) Que pensez vous de ces deux versions récursives ?
   (d) Dans ces deux cas, compter le nombre d'appels récursifs pour les valeurs suivantes de $n = 4, 7, 8, 9, 63, 64, 65$.

Exercice 7. Donner la forme récursive d'une boucle pour qui (parcourt et) affiche les indices successifs de 11 à 1. Même question pour afficher 1, 2, . . ., 10, 11.

3 Objectif 20

Exercice 8. Compléter l'exercice 1 (recherche dichotomique) en indiquant quel est le nombre maximal d'itérations nécessaires à la terminaison de l'algorithme.

Exercice 9. *Jeu de la devinette : je pense à un nombre compris entre a et b. La machine me pose des questions où je réponds par oui ou par non, jusqu'à ce qu'elle trouve le nombre choisi.* Écrire l'algorithme suivi par la machine et qui devine le plus rapidement possible ce nombre pour a et b arbitraires. Bien expliciter la question posée. Coder cet algorithme en C et s'amuser.

Exercice 10. La suite de Syracuse s d'un nombre N donné est définie comme suit.
On commence avec $s(0) = N$, puis :
$s(n+1) = s(n)/2$, si $s(n)$ est pair,
$s(n+1) = 3s(n) + 1$, si $s(n)$ est impair.

1. Calculer les valeurs de la suite pour $N = 4$, $N = 2$ et $N = 1$. Qu'en déduire et en particulier que penser de la terminaison de cette suite ?

2. La coder (en C) et observer son comportement pour différentes valeurs de N. Attention ! ! !
3. On ajoute la condition d'arrêt $s(n) = 1$ qui est rencontrée quelque soit le terme de départ N (d'après la conjecture de Collatz). Écrire des fonctions qui calculent les notions suivantes (source Wikipédia), s'amuser avec et proposer des graphiques instructifs.
   (a) Le temps de vol est le plus petit indice n tel que $s(n) = 1$.
   (b) Le temps de vol en altitude est le plus petit indice n tel que $s(n) \leq N$.
   (c) L'altitude maximale est la valeur maximale $s(n)$ de la suite.

Exercice 11. Wikipédia décrit l'algorithme d'Euclide pour calculer le pgcd de deux entiers ($a > b$) comme suit.

Soient deux entiers naturels a et b, dont on cherche le PGCD. Le cas où a ou b est nul ne nécessite aucun algorithme ; on l'exclut. Une suite d'entiers $(a_n)_n$ est définie par récurrence de pas 2, plus précisément par divisions euclidiennes successives ; la suite est initialisée par $a_0 = a, a_1 = b$, puis propagée par la règle de récurrence : tant que a_{n+1} est non nul, a_{n+2} est défini comme le reste de la division euclidienne de a_n par a_{n+1}.

On commence donc par calculer le reste de la division de a par b, qu'on note r ; puis on remplace a par b, puis b par r, et on ré-applique le procédé depuis le début.

On obtient ainsi une suite, qui vaut 0 à un certain rang ; le PGCD cherché est le terme précédent de la suite.

1. Écrire un algorithme itératif qui calcule ce pgcd.
2. Écrire un algorithme récursif qui calcule ce pgcd.
3. Proposer une formulation synthétique de l'algorithme d'Euclide (qui remplacerait avantageusement l'extrait de Wikipédia).
4. Profiter de la page Wikipédia pour lire comment sont prouvées la correction et la terminaison l'algorithme d'Euclide, ainsi que sa complexité (problème beaucoup plus difficile).

Exercice 12. On souhaite additionner deux entiers de p chiffres décimaux. On dispose pour cela d'une fonction `add(c1,c2)` qui calcule et retourne un couple `(r,s)` où :
— `c1, c2, s` sont des entiers compris entre 0 et 9,
— `r` est égal à 0 ou à 1, et
— `c1 + c2 = 10 r + c`.
Ainsi `r` est la retenue de l'addition des chiffres `c1` et `c2` et `s` est la valeur "des unités" de cette somme. Par exemple : `add(2,3)` retourne `(0,5)` et `add(8,5)` retourne `(1,3)`.

On représente un entier n à p chiffres décimaux par un tableau N de longueur p où $N[i] = n\%10^i, 0 \leq i < n$. Dit plus simplement, le i-ème chiffre de n est stocké en position i dans le tableau N — et on notera qu'il est ainsi plaisant que les tableaux soient indicés à partir de 0. Par exemple :

1. Utiliser la fonction `add` pour écrire un algorithme qui calcule la somme de deux entiers $n1$ et $n2$ respectivement représentés par les tableaux $N1$ et $N2$. Le résultat $s = n1 + n2$ sera aussi représenté par un tableau adapté.
2. Soient $n1 = 1234$, $n2 = 4567$ et $n3 = 9876$. Dérouler l'algorithme pour les deux calculs $n1+n2$ et $n1+n3$.

Exercice 13. Tri fusion
Le tri fusion (*Merge Sort*) est l'application directe du principe diviser pour régner.

On part d'une liste de n valeurs, par exemple entières et stockées dans un tableau 1D (de longueur n).

Diviser : diviser la suite de n valeurs à trier en 2 sous-suites de $n/2$ valeurs chacune.

Régner : trier les deux sous-suites de manière récursive en utilisant le tri fusion.

Combiner : fusionner les 2 sous-suites triées pour produire la réponse triée.

La récursivité s'arrête quand la suite à trier est de longueur 1 : elle est triée et il n'y a donc plus rien à faire.

1. Appliquer ce principe pour trier le tableau $t = [3, 4, 1, 2, 6, 5]$.
2. Écrire une procédure `fusion` dont les paramètres sont décrits dans l'en-tête suivante :
   `procedure fusion(A[p,r] : in out tableau d'entiers; p,q,r : in entiers)`
   et qui fusionne les 2 sous-tableaux $A[p..q]$ et $A[q+1..r]$ en un tableau trié $A[p..r]$. Cette fusion suppose :
   — les indices p, q, r vérifient : $p \leq q < r$,
   — les sous-tableaux $A[p..q]$ et $A[q+1..r]$ sont triés,

— le tableau ainsi fusionné remplace le tableau $A[p..r]$.
La procédure `fusion` pourra créer et utiliser des tableaux locaux.

3. Valider la procédure `fusion` dans les étapes correspondant au tri du tableau T de la question 1.
4. Expliquer le principe d'une fonction récursive `triFusion` qui effectue le tri fusion d'un (sous-)tableau $A[p, r]$. Bien préciser la condition de terminaison de cette fonction.
5. Ecrire l'en-tête de `triFusion`.
6. Ecrire le corps de `triFusion`.
7. Ecrire l'algorithme qui trie le tableau T de la question 1 avec `triFusion`.
8. Combien d'appels à `triFusion` sont nécessaires pour trier T ? Justifier la réponse à l'aide d'une représentation sous la forme d'un arbre des appels.
9. On veut trier un tableau de longueur $n = 2^p$. Combien d'appels à `triFusion` sont nécessaires.
10. $\star$ On note $C(n)$ le coût pour construire avec `fusion` un tableau de longueur n à partir de 2 tableaux de longueurs adéquates. Justifier qu'on peut écrire $C(n) = c_1 \times n$, où c_1 est une constante strictement positive.
11. On note $D(n)$ le coût d'une division en 2 d'un tableau de longueur n. Expliciter le coût $T(n)$ (la complexité) de `triFusion` sous la forme d'une relation de récurrence qui dépend de $D(n)$ et $C(n)$.
12. Justifier qu'on peut supposer (par exemple) pour $n > 1$:
    (a) $T(1) = c$, où c est une constante strictement positive ;
    (b) $D(n) = c_2$, où c_2 est une constante strictement positive.
13. En déduire une expression plus simple de la relation de récurrence sur $T(n)$ et préciser $T(1)$.
14. $\star$ Expliciter à l'aide d'une représentation avec des arbres (binaires), l'évolution de $T(n)$ pour le tri d'un tableau de longueur $n = 2^p$.
15. En déduire une expression de la complexité $T(n)$ du tri fusion. Quelle est sa complexité asymptotique ?