Semaine 2 : Série d'exercices sur les algorithmes

Introduction

Ces séries d'exercices comprennent plusieurs parties :

- une première partie que nous considérons comme « obligatoire », qui devrait être faite exhaustivement ; elle devrait consistuer une charge de travail d'environ 1h00 à 2h30 (dont 45 min en classe avec nous) ;
- une seconde partie optionnelle, « pour aller plus loin » pour celles et ceux qui sont intéressé(e)s ; nous conseillons néanmoins à la grande majorité d'entre vous d'au moins en regarder le corrigé ;
- une troisième partie, elle aussi optionnelle, propose des sujets plus amusants, plus « fun », liés à la matière ; pour celles et ceux que cela amuse, justement...
- une quatrième partie, enfin, fait le lien avec le cours de programmation (et relève donc des exercices du cours de programmation) ; nous vous encourageons à les faire dès que possible comme entraînement sur les *deux* parties du cours (programmation et théorie), mais cela doit attendre la semaine 6 de programmation afin d'avoir assez de notions dans ce domaine pour être profitable.

Ces exercices ne sont pas des exemples de ce qui sera donné à l'examen, mais des moyens pédagogiques complémentaires pour vous faire passer du cours à l'examen : ils sont donc une étape *intermédiaire*, qui se veut être un *complément* du cours et non pas une préparation en tant que telle à l'examen.

Les exercices sont proposés par niveau de difficulté croissante (indiquée, de 1 à 3). La difficulté « niveau 1 » devrait correspondre de 5 à 10 minutes de travail, la « niveau 2 » de 10 à 15 minutes et la « niveau 3 » au dela. Les exercices (ou parties d'exercices) marqué(e)s d'une étoile sont optionnel(le)s.

Faites moi savoir si un exercice ne correspond vraiment pas à cette estimation de niveau (dans un sens ou dans l'autre).

Ensuite, pour vous préparer à l'examen, nous fournissons sur le Moodle du cours les sujets des examens des deux années précédentes. Les corrigés seront également disponibles un peu plus tard (afin de vous laisser le temps de vraiment vous entraîner) ; mais vous pouvez, bien sûr, poser des questions à leur sujet sur le forum !

Vous trouverez au dos la correspondance des différentes questions avec les semaines du cours.

Un dernier conseil enfin : n'attendez pas la semaine de l'examen (ni même celle d'avant) pour vous y préparer : la charge de travail serait alors trop importante ; mais préparez vous dès maintenant à l'examen en faisant, après la série d'exercice, les parties correspondantes, par exemple dans l'examen d'il y a deux ans (voir ci-dessous pour le détail des liens entre ces examens et le cours) et gardez par exemple l'examen de l'année passée pour un dernier entraînement, complet en une fois.

Nous en profitons pour rappeler qu'un cours à « 3 crédits » (partie correspondant à la partie théorique du cours ICC) suppose en moyenne un travail de **6 heures** par semaine : 3 ensemble en classe (cours + exercices) et 3 par vous-même « à la maison » ; essayez de répartir cette tâche au mieux sur les semaines plutôt que de la concentrer uniquement sur les semaines précédent les examens, ce qui ferait trop.

En espérant que ces conseils vous seront utiles !

Correspondance, semaine de cours d'ICC par semaine de cours, avec les questions du premier examen de quelques années précédentes :

- **semaine 2 (I.1 : algorithmes, complexité des algorithmes) :**
  - examen 1 2022 : Q7 ;
  - examen 1 2023 : Q1.3, Q2.1–4, Q4.7–8, Q5.4 ;
  - examen 1 2024 : Q1.2, Q1.6, Q3.1 ;
  - examen 1 2025 : Q2.4 (mais nécessite de connaître l'écriture binaire, vue en leçon I.4) ;
- **semaine 3 (I.2 : stratégies, récursivité) :**
  - examen 1 2022 : Q2, Q3.4, Q8 ;
  - examen 1 2023 : Q2.5–6 ;
  - examen 1 2024 : Q1.4, Q3.3 ;
  - examen 1 2025 : Q3 ;
- **semaine 4 (I.3 : calculabilité, complexité des problèmes) :**
  - examen 1 2022 : Q1.4, Q1.6, Q6 ;
  - examen 1 2023 : Q3, Q4.1–6 ;
  - examen 1 2024 : Q1.3, Q1.5, Q1.7, Q2, Q3.2, Q3.4, Q4 ;
  - examen 1 2025 : Q2.1 et 2.2, Q3.2, 3.4 et 3.5 ;
- **semaine 5 (I.4 : représentation des nombres) :**
  - examen 1 2022 : Q1.1–3, Q1.5 ;
  - examen 1 2023 : Q7 ;
  - examen 1 2024 : Q1.1, Q1.8, Q1.9 ,Q1.10 ;
  - examen 1 2025 : Q2.3 et 2.4, Q4.

1 [N1] Quel est le bon algorithme ?

Lequel des quatre algorithmes suivants permet de calculer la somme des n premiers nombres pairs ? Par exemple : si $n = 4$, alors s doit valoir $2 + 4 + 6 + 8 = 20$.

Expliquez pourquoi les autres ne fonctionnent pas.

a)

algorithme 1.A

entrée : n *entier naturel non-nul*
sortie : s

$s \longleftarrow 0$
Pour i allant de 1 à n
  Si i est pair
    $s \longleftarrow s + i$
Sortir : s

b)

algorithme 1.B

entrée : n *entier naturel non-nul*
sortie : s

$s \longleftarrow 0$
Pour i allant de 1 à n
  $s \longleftarrow s + 2i$
Sortir : s

c)

algorithme 1.C

entrée : n *entier naturel non-nul*
sortie : s

$s \longleftarrow 0$
Pour i allant de 1 à $2n$
  $s \longleftarrow s + i$
Sortir : s

d)

algorithme 1.D

entrée : n *entier naturel non-nul*
sortie : s

Pour i allant de 1 à n
  $s \longleftarrow s + i$
Sortir : s

Note : Pour identifier concrètement si un nombre b est un multiple de a, on vérifie si le reste de sa division par a est nul, c.-à-d. si « b modulo a », c.-à-d. « b mod a », vaut 0.
Dans les exercices (et examens) de ce cours, vous pourrez vous contenter d'écrire « b est pair », « b est divisible par 5 », etc.

2 [N1] Que font ces algorithmes ?

Pour chacun des algorithmes suivants, indiquer (en mots) quelle est la sortie de l'algorithme **et** (en notation Θ) quelle est sa complexité (temporelle pire cas).

a)

algorithme 2.A

entrée : n *entier naturel*
sortie : *? ?*

$m \longleftarrow n$
$i \longleftarrow 1$
Tant que $m > 0$
  $i \longleftarrow 2i$
  $m \longleftarrow m - 1$
Sortir : i

b)

algorithme 2.B

entrée : a, b *entiers naturels non-nuls*
sortie : *? ?*

$s \longleftarrow 0$
Si $a < b$
  Pour i allant de 1 à a
    $s \longleftarrow s + b$
Sinon
  Pour i allant de 1 à b
    $s \longleftarrow s + a$
Sortir : s

3 Au temps des Egyptiens

3.1 [N1] Comprendre

Que fait l'algorithme suivant ? Essayez par exemple $a = 5,\ b = 7$, et si vous ne voyez pas essayez $a = 10,\ b = 9$.

algorithme 3

entrée : *a, b deux entiers naturels non nuls*
sortie : *? ?*

```
x ⟵ a
y ⟵ b
z ⟵ 0
Tant que y ≥ 1
    Si y est pair
        x ⟵ 2 x
        y ⟵ y/2
    Sinon
        z ⟵ z + x
        y ⟵ y − 1
Sortir : z
```

Note : la « science du calcul » (*Computer Science*), en tout cas les algorithmes, existe(nt) depuis bien avant les ordinateurs. L'algorithme ci-dessus nous vient de l'Égypte ancienne, où il était bien pratique pour le commerce (ou les comptables ?...).

3.2 * Démontrer (N3, difficile)

Sauriez vous ensuite démontrer formellement votre réponse (= prouver que l'algorithme fait bien toujours ce que vous prétendez) ?

4 [N2 sans d) ; N3 avec d)] Et que fait celui-ci ?

Considérez l'algorithme suivant [1] :

algorithme 4

entrée : *L liste de nombres entiers*
sortie : *? ?*

```
n ⟵ taille(L)
x ⟵ |L(2) − L(1)|
Pour i allant de 1 à n − 1
    Pour j allant de i + 1 à n
        Si |L(j) − L(i)| > x
            x ⟵ |L(j) − L(i)|
Sortir : x
```

a) Quelle est la sortie de cet algorithme pour $L = (17, -4, 22, 19)$?

b) Quelle est la sortie de cet algorithme (en général) ?

c) Quelle est la complexité (temporelle pire cas) de cet algorithme ?

d) * Pouvez-vous penser à un algorithme plus efficace [2] dont la sortie soit identique ?

1. $L(i)$ réfère au i^{e} élément de la liste L ; ainsi, dans la liste du point **a)** ci-dessous, $L(2)$ vaut -4.
2. Par « plus efficace », on entend ici que sa complexité, c.-à-d. le nombre d'opérations effectuées par le nouvel algorithme, soit *considérablement moindre* pour des grandes valeurs de n.

5 [N2] Réparez-moi ces algorithmes !

Un enseignant du cours ICC a demandé à Jeanne et à Jean d'écrire un algorithme qui calcule la somme des n premiers éléments de la liste des multiples de 5 ou de 7. Voici ce que Jeanne et Jean ont écrit :

a)

algorithme de Jeanne

entrée : n *entier positif*
sortie : s

$s \longleftarrow 0$
$j \longleftarrow 1$
Pour i allant de 1 à n
  Tant que j n'est ni un multiple de 5 ni un multiple de 7
    $j \longleftarrow j + 1$
  $s \longleftarrow s + j$
Sortir : s

b)

algorithme de Jean

entrée : n *entier positif*
sortie : s

$s \longleftarrow 0$
$i \longleftarrow 0$
$j \longleftarrow 0$
Tant que $i < n$
  $j \longleftarrow j + 1$
  Si j est un multiple de 5
    $s \longleftarrow s + j$
    $i \longleftarrow i + 1$
  Si j est un multiple de 7
    $s \longleftarrow s + j$
    $i \longleftarrow i + 1$
Sortir : s

Malheureusement, chacun des algorithmes précédents a un problème (différent). Dans chacun des cas, voyez-vous lequel ? Et pouvez-vous aider Jeanne et Jean à réparer chacun leur algorithme ?

6 [N3] Création d'algorithmes

Soit `L` une liste d'entiers (pas forcément triée et avec de possibles répétitions), comme par exemple `(19, 31, 15, 21)`.

6.1 La plus petite valeur

Ecrivez un algorithme qui permet de trouver la plus petite valeur de la liste. Par exemple avec la liste précédente, l'algorithme doit retourner la valeur `15`.

Votre algorithme fonctionne-t-il avec des valeurs *ex aequo*, comme par exemple avec la liste `(15, 31, 15, 21)` ?

Donnez la complexité temporelle au pire cas de votre algorithme (en utilisant la notation $\Theta(.)$).

6.2 La plus petite différence

On souhaite maintenant trouver deux valeurs de la liste qui ont la plus petite différence entre elles.

Par exemple, si l'algorithme prend en entrée la liste de départ `(19, 31, 15, 21)`, il doit afficher la paire de nombres `(19, 21)` car toutes les autres paires possibles ont entre elles une différence plus grande que 2.

1. Écrivez un algorithme réalisant ce traitement.
2. Donnez la complexité temporelle au pire cas de votre algorithme (en utilisant la notation $\Theta(.)$). Expliquez comment vous êtes parvenu(e) à ce résultat.

Pour aller plus loin (optionnel)

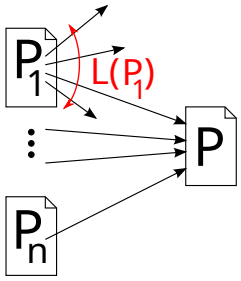
7 [N3] PageRank

7.1 Contexte

Les algorithmes ne sont pas que des abstractions de théoriciens mais peuvent avoir une réelle valeur économique. Par exemple, PageRank® est un algorithme, somme toute assez simple, qui a permis à Google de devenir si célèbre.

Le but de PageRank est de donner une « note » à chaque page Web en fonction de la note de chacune des pages qui la citent (c.-à-d. qui contiennent un lien vers la page en question.)

7.2 Algorithme



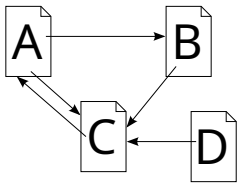
Plus précisément [3], le « PageRank » PR d'une page P ayant n autres pages P_1, ..., P_n qui la citent, est défini par :

$$\mathrm{PR}(P) = 0.15 + 0.85 \left(\frac{\mathrm{PR}(P_1)}{L(P_1)} + ... + \frac{\mathrm{PR}(P_n)}{L(P_n)} \right)$$

où $L(x)$ est le nombre de liens sortant de la page x vers d'autres pages.

Par exemple, si :

- la page A cite les pages B et C,
- la page B cite la page C,
- la page C cite la page A,
- la page D cite la page C,



alors on a $L(A) = 2$ et $L(B) = L(C) = L(D) = 1$ (nombre de liens sortant) et :

$$\begin{aligned}
\mathrm{PR}(A) &= 0.15 + 0.85\,\mathrm{PR}(C) \\
\mathrm{PR}(B) &= 0.15 + 0.85\,\frac{\mathrm{PR}(A)}{2} \\
\mathrm{PR}(C) &= 0.15 + 0.85 \left(\frac{\mathrm{PR}(A)}{2} + \mathrm{PR}(B) + \mathrm{PR}(D) \right) \\
\mathrm{PR}(D) &= 0.15 + 0.85 \times 0 = 0.15
\end{aligned}$$

Sur un exemple aussi simple, on peut très facilement résoudre les équations ; mais dans la réalité du Web le nombre de variables est bien trop grand ! PageRank calcule donc ces scores de façon approchée et itérative. Il fonctionne de la façon suivante :

- au départ, les notes PageRank de toutes les pages sont égales à l'inverse du nombre total de pages (0.25 dans l'exemple ci-dessus),
- ensuite on recalcule la note de chaque page en utilisant les notes précédentes,

3. Référence : `http://infolab.stanford.edu/pub/papers/google.pdf`.
Voir aussi la vidéo à l'adresse `https://www.youtube.com/watch?v=wR0wVxK3m_o`.

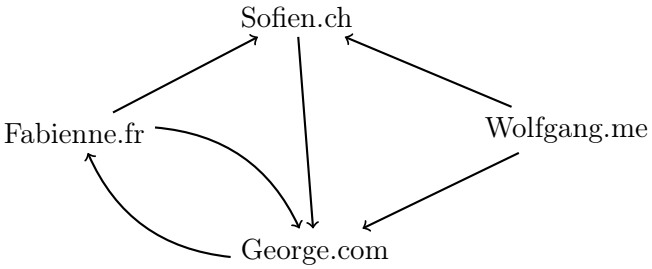


FIGURE 1 – PageRank : liens entre les pages des amis de Rosa.

— et on itère comme cela sans arrêt (les notes sont constamment recalculées, le Web évoluant tout le temps).

Pour l'exemple précédent cela donne :

	étape 1	étape 2	étape 3	étape 4	...		...
PR(A)	0.25	0.36250	0.72906	0.70197	...	1.49011	...
PR(B)	0.25	0.25625	0.30406	0.45985	...	0.78330	...
PR(C)	0.25	0.68125	0.64937	0.84580	...	1.57660	...
PR(D)	0.25	0.15	0.15	0.15	...	0.15	...

7.3 Exercice

Rosa vient de créer son blog personnel et souhaiterait que ses articles atteignent un large public. Pour que les internautes puissent tomber sur son blog, il lui faut être classée le mieux possible par Google. Pour ce faire elle aimerait maximiser son score, tel que donné par l'algorithme PageRank.

Il se trouve que Rosa a quatre amis qui ont déjà une page web. La figure 1 représente les liens entres les pages web des amis de Rosa. Par exemple, le site de Sofien contient un lien vers le site de George, mais il n'y a pas de lien du site de George vers le site de Sofien.

Rosa décide de demander à un de ses amis d'insérer dans sa page web un lien pointant vers le blog de Rosa. Quel ami Rosa devra-t-elle choisir pour maximiser le score donné par PageRank à son blog ?

Indications : commencez par calculer une *approximation en 3 étapes* du score PageRank de chacune des pages des amis de Rosa. Écrivez ensuite la forme (équation) du score PageRank de la page de Rosa si un seul ami l'ajoute sur sa page. Conclure.

8 EdgeRank

8.1 Algorithme

Il n'y a pas que PageRank® (cf. exercice 7) qui ait participé à faire la fortune de ses inventeurs. Un autre algorithme très célèbre de nos jours est l'algorithme EdgeRank utilisé par Facebook pour classer les « posts » de vos amis.

EdgeRank calcule un score pour chacun des « posts » de vos amis et Facebook n'affiche sur votre fils d'actualité que les « posts » ayant les meilleurs scores.

Ce calcul utilise trois grandeurs pour évaluer un « post » P en fonction des activités liées à P qui sont issues d'un ami X de l'utilisateur U [4] :

4. Cf. la vidéo à l'adresse `https://www.youtube.com/watch?v=1ParbwnSJpM`.

— l'affinité A (de U pour X), qui mesure combien l'utilisateur U « suit » l'ami X (qui a une activité liée à P) ;
— le (poids du) type T d'activité liée à P : publication (avec un poids différent si P est une vidéo, un texte court, ...), commentaire, partage, etc. ;
— la fraîcheur F de l'activité liée à P ; c'est simplement l'inverse de l'ancienneté (sa « durée ») : plus une activité est ancienne, moins elle est fraîche ;

Ces grandeurs sont combinées comme suit pour calculer le score EdgeRank :

— on dit qu'un ami a un lien (Edge) avec un « post » s'il y intervient (création ou commentaire) ;
— le score EdgeRank ER d'un « post » P est simplement la somme pour tous les amis ayant un lien avec P du produit de A par T par F :

$$\mathrm{ER}(P) = A_1 \times T_1 \times F_1 + A_2 \times T_2 \times F_2 + ... + A_n \times T_n \times F_n$$

Par exemple, si un ami X_1 de U pour lequel U a une affinité faible ($A_1 = 1$) a créé il y a 2 heures ($F_1 = 1/2$) un « post » vidéo (de type, disons, 5) et qu'un ami X_2 de U pour lequel U a une affinité forte ($A_2 = 2$) a ajouté un commentaire (de type, disons, 3) il y a une heure ($F_2 = 1/1 = 1$), alors le score EdgeRank de ce « post » pour le fil d'actualité de U à ce moment là sera :

$$\mathrm{ER}(P) = 1 \times 5 \times \frac{1}{2} + 2 \times 3 \times 1 = 8.5$$

8.2 Exercice

George est sur Facebook et il aimerait faire circuler une vidéo au delà de son cercle d'amis.

Le réseau social des amis de George est décrit par la figure 2.

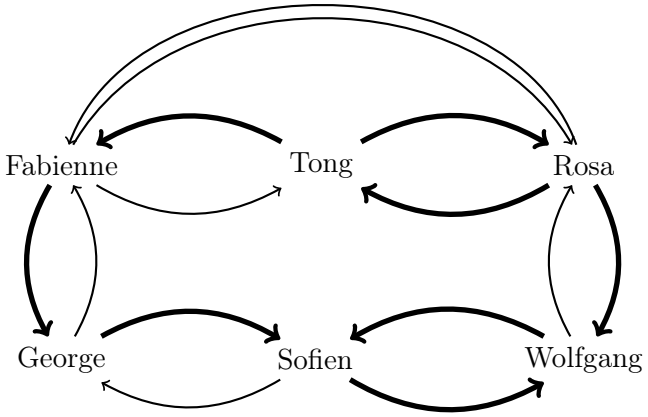


FIGURE 2 – EdgeRank : réseau social des amis de George.

Dans cette figure, si deux personnes sont amies alors elles sont reliées par deux flèches de sens contraire. Une flèche fine indique une affinité de 1 (faible) ; une flèche épaisse indique une affinité de 2 (forte). Par exemple, Tong et Fabienne sont amis. Tong a une affinité forte pour Fabienne alors que Fabienne a une affinité faible pour Tong. Tong et Sofien ne sont pas amis.

Question 1. Supposons que George poste sa vidéo sur Facebook. Si rien d'autre ne se passe, son « post » atteindra-t-il Rosa ?

Question 2. Pour simplifier, supposons que Facebook affiche uniquement les « posts » ayant un score EdgeRank supérieur à 0.5. On suppose aussi que le type « vidéo » a un poids de 5 et que la fraîcheur vaut l'inverse du temps écoulé en heures. Le « post » de George apparaitra-t-il sur le fil d'actualités de ses amis et pour combien de temps ?

Question 3. Pour faire circuler sa nouvelle au delà de ses amis, George décide de demander à un des ses amis de « liker son post ». Quel ami doit il choisir pour maximiser le nombre de personnes qui verront son « post » ? On suppose que le type « like » a un poids de 1.

Question 4. Au bout de combien d'heures après le « like » de l'ami choisi à la question 3 George n'a-t-il plus aucune chance que son « post » soit vu par une personnes avec qui il n'est pas ami ?

Question 5. Supposons que Fabienne commente le « post » de George et que Tong « like » le commentaire de Fabienne une heure après ? On suppose que le type « commentaire » a un poids de 3. Au bout de combien de temps Rosa ne pourra-t-elle plus accéder à la vidéo de George (score inférieur à 0.5) ?

9 La plus courte séquence

Soit `M` une liste de zéros (0) et de un (1), par exemple `(1, 0, 0, 1, 1, 1, 0)`.

On souhaite identifier la longueur de la k^{e} plus courte séquence de 1 consécutifs dans `M`.

Par exemple, si l'algorithme prend en entrée la liste :

```
M = (0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1)
```

et le nombre `k = 2`, il doit afficher `3`. Il y a en effet dans `M` quatre séquences de 1 consécutifs : une séquence avec un seul 1, une séquence avec cinq 1, une séquence avec trois 1 et une séquence avec quatre 1. La deuxième plus petite séquence de 1 consécutifs est donc celle de longueur 3.

1. Écrivez un algorithme réalisant ce traitement. On suppose que l'on dispose d'un algorithme « `trier` » permettant de trier une liste d'entiers.
2. Quel est le nombre d'instructions élémentaires exécutées par votre algorithme ? Vous supposerez que `trier` permet de trier une liste de n entiers en utilisant de l'ordre de $n \log(n)$ instructions élémentaires.

Pour le fun... (optionnel)

Un algorithme de tri inhabituel

Dans un pays loin d'ici, vivaient sous le joug d'un magicien 99 nains (c'est ce qu'il en restait) sourds et muets. Comme tous les nains, ils avaient chacun un bonnet.

Tous les soirs, lorsqu'ils rentraient dans leur caverne très sombre (sans aucune visibilité), le magicien changeait la couleur de leur bonnet : rouge ou bleu. Ils n'avaient donc aucun moyen de savoir la couleur de leur bonnet, ni des autres (il faisait noir), et bien sûr aucun moyen de communiquer.

Tous les matins, par contre ils devaient sortir (au jour) un par un et se présenter au magicien en ligne (de face, côte à côte) de sorte à ce que tous les nains à bonnet bleu (B) soient d'un coté et tous les nains à bonnet rouge (R) de l'autre ; par exemple :

....RRRRRRRRBBBBBBBBB.....

Les nains étaient très motivés car le magicien détruisait tout nain qui n'était pas à sa place (ainsi que quelques voisins : boule de feu !).

Comment ont fait ces 99 nains pour réussir à se présenter bien ordonnés tous les matins et ainsi survivre ?

Une variante

Imaginez maintenant un autre magicien et des autres nains, ni sourds ni muets, mais toujours au nombre de 99, et toujours avec un bonnet de couleur rouge ou bleue sur la tête. Ils sont maintenant alignés en file indienne dans la caverne dans le noir. Au matin, ils sortent un à un de la caverne, en commençant par celui qui se trouve en tête de file ; à chaque fois, le magicien révèle à l'oreille du nain sortant le nombre de bonnets rouges restants derrière lui. Le nain doit ensuite annoncer la couleur de son *propre* bonnet. Comment les nains vont-ils faire pour s'en sortir ?

Remarques :

- on suppose que les nains ont toute la nuit pour élaborer une stratégie commune ;
- quoi qu'il arrive, le premier nain qui sort a 50% de chances d'y passer, malheureusement ; mais tous les suivants peuvent être sauvés à coup sûr ;
- une bonne façon d'aborder le problème est de faire l'identification : rouge=1, bleu=0.

Cours ICC : liens théorie $\longleftrightarrow$ Programmation

- Tous les exercices des séries 3 et 4 de programmation sont de bons entraînements pour l'écriture d'algorithmes assez simples.
- L'exercice 6 de la série 6 de programmation reviendra sur la recherche par dichotomie.

Retrouvez tous les exercices de programmation liés à la partie théorie du cours sous le lien « Exercices de C++ spécifiques à ICC (lien programmation - théorie) » en bas de la page Moodle du cours, dans la section « Ressources complémentaires / Références ».