EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communications
Cours ICC aux sections MA et PH
Chappelier J.-C.

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

INFORMATIQUE, CALCUL & COMMUNICATIONS

Sections MA & PH

Correction Examen intermédiaire I

23 octobre 2015

SUJET 1

Instructions :

— Vous disposez d'une heure quinze minutes pour faire cet examen (15h15 - 16h30).

— L'examen est composé de 2 parties : un questionnaire à choix multiples, à 12 points, prévu sur 45 minutes, et une partie à questions ouvertes, à 8 points, prévue sur 30 minutes.
Mais vous êtes libres de gérer votre temps comme bon vous semble.

— **AUCUN DOCUMENT N'EST AUTORISÉ, NI AUCUN MATÉRIEL ÉLECTRONIQUE**.

— Pour la première partie (questions à choix multiples), chaque question n'a qu'une seule réponse correcte parmi les quatre propositions.
Indiquez vos réponses en bas de **cette** page en écrivant *clairement* pour chaque question une lettre majuscule parmi A, B, C et D.
Aucune autre réponse ne sera considérée, et en cas de rature, ou de toute ambiguïté de réponse, nous compterons la réponse comme fausse.
(Vous êtes autorisés à dégrafer cette page)

— Pour la seconde partie, répondez directement sur la donnée, à la place libre prévue à cet effet.

— Toutes les questions comptent pour la note finale.

NOTE : dans cet examen, le premier élément d'une liste L est noté $L[1]$.

Réponses aux quiz :

Reportez ici *en majuscule* la lettre de la réponse choisie pour chaque question, sans aucune rature.

1	2	3	4	5	6	7	8	9	10	11	12
A	B	A	C	B	C	B	C	A	D	B	A

PARTIE QUIZ

1 – Des problèmes... [2 points]

Question 1) Parmi les affirmations suivantes, laquelle est vraie ?

✔**A]** $P \subset NP$

B] $NP \subset P$

C] NP est l'ensemble des problèmes insolubles par l'informatique

D] NP n'est pas dans P

Question 2) On considère une collection d'objets plats posés sur une table, ne se touchant que par des cotés (c.-à-d. en contact par des segments ; pas de contact ponctuel, « en pointe »).

Le problème consistant à déterminer s'il est possible de colorier une telle collection d'objets sans jamais attribuer la même couleur à deux objets qui se touchent est :

A] dans P pour 2 et 3 couleurs et dans NP pour 4 couleurs

✔**B]** dans P pour 2 et 4 couleurs et dans NP pour 3 couleurs

C] dans P pour 2 couleurs et dans NP mais pas dans P pour 3 et 4 couleurs

D] dans P pour 2 et 4 couleurs et dans NP mais pas dans P pour 3 couleurs

---

2 – Des caractères... [3 points]

L'algorithme ci-dessous prend en entrée une « chaînes de caractères » s. On notera s_i le i^e caractère de s (à partir de 1). « $s_i \cdots s_j$ » est donc la sous-chaîne de s allant de son i^e à son j^e caractère. Si $i > j$, « $s_i \cdots s_j$ » représentera par convention la chaîne vide.

Par exemple, si s est « `examen` », s_3 est le caractère '`a`', « $s_3 \cdots s_6$ » est la chaîne « `amen` » et « $s_4 \cdots s_3$ » est la chaîne vide.

sékoi
entrée : *chaîne de caractères s* sortie : *? ?*

$t \longleftarrow$ **taille**(s)
Si $t < 2$
    sortir : oui
Si $s_1 = s_t$
    sortir : sékoi$(s_2 \cdots s_{t-1})$
Sinon
    sortir : non

Question 3) **sékoi**(`kayak`) :

✔**A]** renvoie « oui » **B]** renvoie « non » **C]** ne renvoie rien **D]** ne se termine pas

suite au dos ☞

Question 4) Si t est la taille du message s et si la complexité de la fonction **taille** est en $\mathcal{O}(1)$, la complexité de **sékoi** est :

A] $\mathcal{O}(\log t)$, mais pas $\mathcal{O}(1)$

B] $\mathcal{O}(t \log t)$, mais pas $\mathcal{O}(t)$

✔**C]** $\mathcal{O}(t)$, mais pas $\mathcal{O}(\log t)$

D] $\mathcal{O}(t^2)$, mais pas $\mathcal{O}(t)$

Question 5) Lequel des algorithmes suivants est correct et résout le même problème que **sékoi** ?

Note : La notation $\lfloor x \rfloor$ représente la « partie entière par défaut » de x, c.-à-d. le plus grand entier inférieur ou égal à x, et $\lceil x \rceil$ la « partie entière par excès », c.-à-d. le plus petit entier supérieur ou égal à x. Par exemple $\lfloor 1.5 \rfloor = 1$, $\lfloor 1 \rfloor = 1$, $\lceil 1.5 \rceil = 2$ et $\lceil 1 \rceil = 1$.

A]

algo1
entrée : *message s* sortie : *? ?*

$t \longleftarrow$ **taille**(s)
Pour i de 1 à $\lceil \frac{t}{2} \rceil$
  Si s_i et s_{t+1-i} sont égaux
    sortir : oui
sortir : non

C]

algo2
entrée : *message s* sortie : *? ?*

$t \longleftarrow$ **taille**(s)
Pour i de 1 à t
  Si $s_{\lfloor i/2 \rfloor}$ et $s_{\lfloor (t-i)/2 \rfloor}$ sont différents
    sortir : non
sortir : oui

✔

algo3
entrée : *message s* sortie : *? ?*

$t \longleftarrow$ **taille**(s)
Pour i de 1 à $\lfloor \frac{t}{2} \rfloor$
  Si s_i et s_{t+1-i} sont différents
    sortir : non
sortir : oui

D]

algo4
entrée : *message s* sortie : *? ?*

$t \longleftarrow$ **taille**(s)
Si s_1 et s_t sont différents
  sortir : algo4$(s_2 \cdots s_{t-1})$
Sinon
  Si $t \geq 2$
    sortir : non
  Sinon
    sortir : oui

---

3 – Des nombres... [3 points]

Question 6) En prenant une représentation *signée* des nombres entiers en binaire sur 4 bits, `1000` représente :

A] -0 **B]** 8 ✔**C]** -8 **D]** -7

Question 7) On s'intéresse ici à la représentation en virgule flottante vue en cours mais utilisant ici 3 bits pour l'exposant (strictement positif) et 3 bits pour la mantisse. Avec une telle représentation, écrite dans l'ordre signe, exposant, mantisse, que vaut l'addition de `0010011` et `1001001` ?

A] `0010110` ✔**B]** `0001101` **C]** `1011010` **D]** `1011100`

Question 8) Combien de bits sont différents entre les représentations binaires de 144 et 125 (représentations de nombres entiers) ?

A] 8 **B]** 2 ✔**C]** 6 **D]** 4

4 – Des algorithmes... [4 points]

Question 9) Les différents états intermédiaires de L dans l'algorithme suivant :

késako
entrée : *L liste de 0 et de 1* sortie : *? ? ?*

$l \longleftarrow$ **taille**(L)
Tant que $l \geq 1$ **<u>et</u>** $L[l] = 1$
   $L[l] \longleftarrow 0$
   $l \longleftarrow l - 1$
Si $l = 0$
   ajouter 1 au début de L
Sinon
   $L[l] \longleftarrow 1$
sortir : L

appliqué à l'entrée $(1, 0, 1, 1)$ sont :

✔**A]** $(1, 0, 1, 1)$, $(1, 0, 1, 0)$, $(1, 0, 0, 0)$, $(1, 1, 0, 0)$

B] $(1, 0, 1, 1)$, $(1, 0, 1, 0)$, $(1, 0, 0, 0)$, $(1, 1, 0, 0)$, $(0, 1, 0, 0)$

C] $(1, 0, 1, 1)$, $(1, 0, 1, 0)$, $(1, 0, 0, 0)$, $(0, 0, 0, 0)$, $(1, 0, 0, 0, 0)$

D] $(1, 0, 1, 1)$, $(0, 0, 1, 1)$, $(0, 1, 1, 1)$

Question 10) **Note :** Pour un ensemble fini E, la notation $|E|$ représente son cardinal, i.e. son nombre d'éléments.

Que calcule/renvoie l'algorithme suivant :

safékoi
entrée : *Deux ensemble finis E_1 et E_2* sortie : *? ? ?*

$l_1 \longleftarrow \|E_1\|$
$l_2 \longleftarrow \|E_2\|$
$w \longleftarrow$ vrai
Pour u de 1 à l_1
   $x \longleftarrow$ faux
   Pour v de 1 à l_2
      Si $E_1[u] = E_2[v]$
         $x \longleftarrow$ vrai
   Si x est faux
      sortir : x
sortir : w

A] $E_1 \cup E_2$ **B]** $E_1 \cap E_2$ **C]** $|E_1| < |E_2|$ ✔**D]** $E_1 \subset E_2$

suite au dos ☞

Question 11) Considérez un algorithme qui affiche la liste de tous les entiers entre 1 et $n!$ (factorielle n). En fonction de la taille de cette liste, la complexité temporelle pire cas de cet algorithme est :

A] au plus constante ✔**B]** au moins linéaire **C]** au moins quadratique **D]** au moins exponentielle

Question 12) Soit $L_b(n)$ le nombre de symboles/chiffres de l'écriture d'un nombre entier n en base b, avec $b \geq 2$. Quelle proposition est vraie ?

✔**A]** $L_b(n)$ est en $\mathcal{O}(\log(n))$ mais pas en $\mathcal{O}(1)$
B] $L_b(n)$ est en $\mathcal{O}(b \times n)$ mais pas en $\mathcal{O}(\log(n))$
C] $L_b(n)$ est en $\mathcal{O}\left(n^b\right)$ mais pas en $\mathcal{O}\left(n^{b-1}\right)$
D] $L_b(n)$ est en $\mathcal{O}(b^n)$ mais pas en $\mathcal{O}\left(b^{n-1}\right)$

PARTIE EXERCICES

5 – Deux algorithmes [4 points]

Considérons le deux algorithmes suivants :

algo1
entrée : L, *une liste de nombres* sortie : *??*

$t \longleftarrow$ **taille**(L)
Pour i de 1 à $t-1$
  $k \longleftarrow i$
  Pour j de $i+1$ à t
    Si $L[j] < L[k]$
      $k \longleftarrow j$
  $b \leftarrow L[i]$
  $L[i] \leftarrow L[k]$
  $L[k] \leftarrow b$
sortir : L

algo2
entrée : L, *une liste de nombres* sortie : *??*

$t \longleftarrow$ **taille**(L)
Pour i de 2 à t
  $a \longleftarrow L[i]$
  $j \longleftarrow i$
  Tant que $j > 1$ **<u>et</u>** $L[j-1] > a$
    $L[j] \leftarrow L[j-1]$
    $j \leftarrow j-1$
  $L[j] \leftarrow a$
sortir : L

Question 13) Quels sont les différents états intermédiaires de la liste $L = (2, 4, 3, 7, 1)$ lors du déroulement de l'algorithme « algo1 » ?

Réponse : $(2, 4, 3, 7, 1), (1, 4, 3, 7, 2), (1, 2, 3, 7, 4), (1, 2, 3, 4, 7)$

BARÈME : 1 point. Etat final : 0.5, le reste sur 0.5 si parfait, 0.25 si semble compris mais petite faute, 0 si n'a rien compris.

Question 14) En supposant que le calcul de la taille est au plus linéaire, donner la complexité de chacun des deux algorithmes (en notation de Landau, la plus petite possible). Précisez ce que représente la variable utilisée dans cette notation.

« algo1 » : $\mathcal{O}(n^2)$, où n est la taille de la liste en entrée.

« algo2 » : aussi $\mathcal{O}(n^2)$

BARÈME : 1 point. 0.4 chacun, 0.2 pour préciser la variable.

Question 15) Supposons que l'accès à un élément quelconque de la liste soit une opération en temps constant par rapport à la taille de la liste. L'un de ces algorithmes est-il plus efficace que l'autre en nombre d'opérations dans le pire des cas ? Si oui, lequel ? *Justifiez pleinement* votre réponse.

Réponse : Non, les deux ont exactement la même complexité car rechercher le minimum est aussi couteux que déplacer une valeur de proche en proche.

On peut calculer le nombre d'opération de manière exacte et trouver $\frac{7n^2}{2} + \mathcal{O}(n)$ pour « algo1 » et « algo2 » (ou en tout cas le même coefficient de n^2 ; nous avons ici compté 1 pour chaque lecture/écriture de valeur) :

« algo1 » :

$$\mathcal{O}(n) + \sum_{i=1}^{n-1} \left(\sum_{j=i+1}^{n} 4 \right) + 8 = \mathcal{O}(n) + \sum_{i=1}^{n-1} 4(n-i) + 8 = (n-1)(4n+8) - \sum_{i=1}^{n-1} i = 4n^2 + \mathcal{O}(n) - \frac{n(n-1)}{2}$$
$$= \frac{7n^2}{2} + \mathcal{O}(n)$$

« algo2 » :

$$\mathcal{O}(n) + \sum_{i=2}^{n} 7i + 6 = \mathcal{O}(n) + 7\sum_{i=2}^{n} i = \mathcal{O}(n) + 7 \cdot (\frac{n(n+1)}{2} - 1) = \frac{7n^2}{2} + \mathcal{O}(n)$$

BARÈME : 2 points. 1 point pour chaque justification.

6 – Sous-séquences [4 points]

Question 16) Ecrire un algorithme qui, dans une liste de nombres (positifs et négatifs), retourne la [1] sous-liste (éléments consécutifs) dont la somme est maximale.

Par exemple, pour la liste $(-2, 1, -3, 4, -1, 2, 1, -5, 4)$, la sous-liste de somme maximale est $(4, -1, 2, 1)$ (et sa somme est 6).

Réponse : Voici l'algorithme peut être le plus simple à imaginer, en $\mathcal{O}(n^3)$:

Tranche de somme maximale

entrée : L, *une liste de nombres*
sortie : *sous-liste de* L *de somme maximale*

```
n ⟵ taille(L)
Si n ≤ 1
   sortir : L
extreme ⟵ L[1]
start ⟵ 1
end ⟵ 1
Pour i de 1 à n
   Pour j de i à n
      current ⟵ 0
      Pour k de i à j
         current ⟵ current + L[k]
      Si current > extreme
         extreme ⟵ current
         start ⟵ i
         end ⟵ j
sortir : (L[start], ..., L[end])
```

Voici une version simplifiée en $\mathcal{O}(n^2)$ qui exploite le fait que l'on travaille sur des sous-listes (pas la peine de recalculer la partie commune) :

1. Une quelconque s'il y en a plusieures.

Tranche de somme maximale
entrée : L, *une liste de nombres*
sortie : *sous-liste de* L *de somme maximale*

```
n ⟵ taille(L)
Si n ≤ 1
   sortir : L
extreme ⟵ L[1]
start ⟵ 1
end ⟵ 1
Pour i de 1 à n
   current ⟵ 0
   Pour j de i à n
      current ⟵ current + L[j]
      Si current > extreme
         extreme ⟵ current
         start ⟵ i
         end ⟵ j
sortir : (L[start], ..., L[end])
```

Et voici une solution en $\mathcal{O}(n)$:

Tranche de somme maximale
entrée : L, *une liste de nombres*
sortie : *sous-liste de* L *de somme maximale*

```
n ⟵ taille(L)
Si n ≤ 1
   sortir : L
current ⟵ L[1]
extreme ⟵ current
pos_start ⟵ 1
start ⟵ pos_start
end ⟵ 1
Si 0 > current
   current ⟵ 0
   pos_start ⟵ 2
Pour i de 2 à n
   current ⟵ current + L(i)
   Si current > extreme
      extreme ⟵ current
      start ⟵ pos_start
      end ⟵ i
   Si 0 > current
      current ⟵ 0
      pos_start ⟵ i + 1
sortir : (L[start], ..., L[end])
```

BARÈME : 3 points : 1 pt pour une somme de sous-liste, 1 pt pour l'extremum, 1 pt pour la combinaison du tout.

Question 17) Quelle est la complexité de votre algorithme ?

BARÈME : 1 point.