NSI – Sujets 0

Stéphane Pasquet (mathweb.fr)

15 juillet 2019

Thème « Types et valeurs de base »

1. Combien d'entiers positifs ou nuls peut-on représenter en machine sur 32 bits ?

   (a) $2^{32}-1$ (b) 2^{32} (c) 2×32 (d) 32^2

   Réponse b. En effet, un bit peut représenter 2 états : 0 ou 1. Donc sur 2 bits, on peut représenter $2\times 2 = 2^2$ nombres, etc. Sur une machine de 32 bits, il y a donc 2^{32} possibilités de nombres.

2. Les entiers positifs ou nuls dont l'écriture en base 16 (hexadécimal) est constituée par un 1 suivi de 0 (par exemple 1, 10, 100, 1000, etc.) sont :

   (a) les puissances de 2
   (b) les puissances de 8
   (c) les puissances de 10
   (d) les puissances de 16

   Réponse d. En effet, en hexadécimal, un nombre s'écrivant sous la forme $\overline{x_n x_{n-1}\cdots x_1 x_0}$ est égal, en base 10, à $\sum_{k=0}^{n} x_k \times 16^k$.
   Ainsi, si $x_{n-1} = x_{n-2} = \cdots = x_0 = 0$ et $x_n = 1$, ce nombre est égal à 16^n. C'est donc une puissance de 16.

3. Dans le programme ci-dessous, qui prend en entrée un entier naturel non nul et renvoie son écriture binaire, remplacer les pointillés par l'opérateur qui convient.

```
def cascade(n):
        chiffres = ''
        while n != 0:
                chiffres = str(n ... 2) + chiffres
                n = n // 2
        return chiffres
```

   (a) `//` (b) `+` (c) `*` (d) `%`

   Réponse d. En effet, l'écriture binaire d'un entier s'obtient à partir de divisions euclidiennes successives par 2 en prenant leurs *restes*, obtenus à l'aide de l'opérateur %. Rappelons que « a% b » représente le reste dans la division euclidienne de a par b.

Thème « Types construits »

1. On définit un tableau `t` rempli de 0 en langage Python. Ce tableau est une liste de listes, toutes les sous-listes ayant le même nombre d'éléments.

```
t = [ [0, 0, ... , 0],
      [0, 0, ... , 0],
      ...
      [0, 0, ... , 0] ]
```

   On appelle `h` le nombre de listes contenus dans `t` et `n` le nombre d'éléments appartenant à ces listes. Parmi les propositions suivantes, laquelle permet de calculer `h` et `n` ?

   (a) `h, n = len(t[0]), len(t)`

   (b) `h, e = len(t[0]), len(t[1])`

   (c) `h, e = len(t), len(t[0])`

   (d) `h, e = len(t[1]), len(t[0])`

   Réponse c. En effet, `len(t)` représente le nombre d'éléments du tableau `t`, c'est-à-dire ici le nombre de listes dans `t`. De plus, `len(t[0]` représente le nombre d'éléments de `t[0]`.

2. On considère le code suivant :

```
def f(t):
       for i in range(len(t)//2):
              t[i] , t[-i-1] = t[-i-1] , t[i]
```

   Après les lignes suivantes :

```
t = [2,3,4,5,7,8]
f(l)
```

   quelle est la valeur de `t` ?

   (a) `[2,3,4,5,7,8]`

   (b) `[5,7,8,2,3,4]`

   (c) `[8,7,5,4,3,2]`

   (d) `[4,3,2,8,7,5]`

   Réponse c. En effet, l'instruction `for i in range(len(t)//2)` stipule que l'on parcourt la moitié de la liste `t`. Pour chaque valeur de `i`, on intervertit les valeurs de `t[i]` et `t[-i-1]`. Par exemple, pour `i=0`, `t[0]` et `t[-1]` sont intervertis, c'est-à-dire le premier et dernier élément de la liste `t`. Ici, `len(t)//2=3` donc `i` prend les valeurs 0, 1 et 2. On aura donc successivement :
   - Avant l'entrée dans la boucle : `t = [2,3,4,5,7,8]`
   - Pour $i = 0$: `t = [8,3,4,5,7,2]`
   - Pour $i = 1$: `t = [8,7,4,5,3,2]`
   - Pour $i = 2$: `t = [8,7,5,4,3,2]`

3. On dispose d'un tableau d'entiers, ordonné en ordre croissant. On désire connaître le nombre de valeurs distinctes contenues dans ce tableau. Quelle est la fonction qui **ne convient pas** ?

(a)
```
def compte(t):
        cpt = 1
        for i in range(1,len(t)):
                if t[i] != t[i-1]:
                        cpt = cpt + 1
        return cpt
```

(b)
```
def compte(t):
        cpt = 1
        for i in range(0,len(t)-1):
                cpt = cpt + int(t[i] != t[i+1])
        return cpt
```

(c)
```
def compte(t):
        cpt = 0
        for i in range(0,len(t)-1):
                cpt = cpt + int(t[i] != t[i+1])
        return cpt
```

(d)
```
def compte(t):
        cpt = 0
        for i in range(0,len(t)-1):
                if t[i] != t[i+1]:
                        cpt = cpt + 1
        return cpt+1
```

Réponse c. En effet, ce programme ressemble à celui proposé en (b), à ceci près que l'initialisation de la variable `cpt` n'est pas la même. Le programme ne convenant pas est donc (b) ou (c).
L'instruction `int(t[i] != t[i+1])` renvoie « 1 » si la différence est vérifiée, et « 0 » si les deux valeurs sont égales. Si on prend l'exemple de `t = [1,2,2,3]`, on a (avec le programme (c) :

- Avant l'entrée dans la boucle : `cpt = 0`
- Pour $i = 0$: `cpt = cpt + 1 = 0 + 1 = 1`
- Pour $i = 1$: `cpt = cpt + 0 = 1 + 0 = 1`
- Pour $i = 2$: `cpt = cpt + 1 = 1 + 1 = 2`

Ainsi, au lieu de retourner « 3 » (car dans cet exemple, il y a 3 nombres distincts), le programme retourne 2.

Thème « Traitement de données en tables »

1. On dispose d'une liste de triplets :

```
t = [  (1,12,250),
       (1,12,251),
       (2,12,250),
       (2,13,250),
       (2,11,250),
       (1,12,249) ]
```

On trie cette liste par ordre croissant des valeurs du second éléments des triplets. En cas d'égalité, on trie par ordre croissant du troisième champ. Si les champs 2 et 3 sont égaux, on trie par ordre croissant du premier champ. Après ce tri, quel est le contenu de la liste `t` ?

(a)
```
[ (1,12,249),
  (1,12,250),
  (1,12,251),
  (2,11,250),
  (2,12,250),
  (2,13,250) ]
```

(b)
```
[ (2,11,250),
  (1,12,249),
  (1,12,250),
  (2,12,250),
  (1,12,251),
  (2,13,250) ]
```

(c)
```
[ (2,11,250),
  (1,12,249),
  (1,12,250),
  (1,12,251),
  (2,12,250),
  (2,13,250) ]
```

(d)
```
[ (1,12,249),
  (2,11,250),
  (1,12,250),
  (2,12,250),
  (2,13,250),
  (1,12,251) ]
```

Réponse b. Il suffit ici de respecter les consignes de tris. On commence par prendre le triplet dont le 2e nombre est le plus petit – donc `(2,11,250)` – puis on passe au suivant : il y a 4 possibilités car 4 triplets comporte « 12 » en 2e position ; on trie alors selon le 3e nombre. Il y a 2 triplets ayant « 250 » comme 3e nombre donc on tris en fonction du 1er pour ces deux triplets.

2. À partir de fichiers remplis à l'aide de formulaires en ligne, on dispose d'un tableau `t1` de numéros de client, nom, âge, ville et d'un tableau `t2` de numéros de commande, numéros de client, prix :

```
t1 = [ [3,"Pierre",25,"Ambérieu"],
       [5,"Gaspe",45,"Lagnieu"],
       [98,"Abraham",75,"Meximieux"],
       [24,"Zoé",34,"Lyon"] ]

t2 = [ [1287,5,1025], [13245,98,234],
       [23,5,42],[10001,24,53] ]
```

Que devient `t1` après l'exécution du code suivant :

```
for i in range(len(t1)):
       for c in [ com for com in t2 if com[1]==t1[i][0] ]:
              t1[i].append( (c[0], c[2]) )
```

(a) `[ [5, 'Gaspe', 45, 'Lagnieu', (1287, 1025), (23, 42)]`
`[98, 'Abraham', 75, 'Meximieux', (13245, 234)]`
`[24, 'Zoé', 34, 'Lyon', (10001, 53)] ]`

(b) `[ [3, 'Pierre', 25, 'Ambérieu']`
`[5, 'Gaspe', 45, 'Lagnieu', (1287, 1025), (23, 42)]`
`[98, 'Abraham', 75, 'Meximieux', (13245, 234)]`
`[24, 'Zoé', 34, 'Lyon', (10001, 53)] ]`

(c) `[ [3, 'Pierre', 25, 'Ambérieu', (1287, 1025)]`
`[5, 'Gaspe', 45, 'Lagnieu', (13245, 234)]`
`[98, 'Abraham', 75, 'Meximieux', (23, 42)]`
`[24, 'Zoé', 34, 'Lyon', (10001, 53)] ]`

(d) `[ [3, 'Pierre', 25, 'Ambérieu']`
`[5, 'Gaspe', 45, 'Lagnieu']`
`[98, 'Abraham', 75, 'Meximieux']`
`[24, 'Zoé', 34, 'Lyon'] ]`

Réponse b. Il faut avant tout comprendre l'instruction :

```
for c in [ com for com in t2 if com[1]==t1[i][0] ]:
```

Ici, `com` représente une entrée de `t2` sous la condition que `com[1]` soit égal à `t1[i][0]`. Il faut donc que le 2e nombre de l'élément de `t2` coïncide avec le numéro client dans l'entrée de `t1`. Sous cette unique condition, on ajoute à l'entrée de `t1` les 1er et derniers nombres de l'entrée de `t2`. Concrètement,

- pour $i = 0$, on parcourt `t2` jusqu'à ce que le 2e nombre de l'élément de `t2` soit 3. Comme il n'y en a pas, on n'ajoute rien.
- Pour $i = 1$, on parcourt `t2` jusqu'à rencontrer « 5 » en 2e nombre : il y a 2 triplets qui correspondent donc en ajoute deux fois un couple : (1287,1025) et (23,42).
- etc.

3. On considère le programme suivant :

```
def maxi(tab):
        """
        tab est une liste de couples (nom, note)
        où nom est de type str
        et où note est un entier entre 0 et 20.
        """
        m = tab[0]
        for x in tab:
                if x[1] >= m[1]:
                        m = x
        return m

L = [('Adrien', 17), ('Barnabé', 17), ('Casimir', 17),
     ('Dorian', 17), ('Emilien', 16), ('Fabien', 16)]
```

Quelle est la valeur de maxi(L) ?

(a) `('Adrien',17)`

(b) `('Dorian',17)`

(c) `('Fabien',16)`

(d) `('Emilien',16)`

Réponse b. Exécutons pas à pas le programme :

x	x[1] >= m[1]	Valeur de m
		('Adrien',17)
('Adrien',17)	Vrai	('Adrien',17)
('Barnabé',17)	Vrai	('Barnabé',17)
('Casimir',17)	Vrai	('Casimir',17)
('Dorian',17)	Vrai	('Dorian',17)
('Emilien',16)	Faux	('Dorian',17)
('Fabien',16)	Faux	('Dorian',17)

Thème « Interaction Homme-machine sur le web »

1. Après avoir saisi dans son navigateur l'URL de son forum favori, Clotilde reçoit comme réponse du serveur une erreur « 404 ». Une seule des réponses suivantes ne correspond pas à cette situation, laquelle ?

(a) Une panne de sa connexion internet

(b) Une mise à jour du serveur qu'elle consulte

(c) Une erreur de saisie de sa part

(d) Un changement de titre du forum qu'elle consulte

Réponse a. En effet, l'erreur « 404 »signifie que la page n'a pas été trouvée... et cela suppose que la connexion internet est bonne.

2. Saisir l'URL *http ://monsite.com/monprogramme.py ?id=25* dans la barre d'adresse du navigateur **ne permet à coup sûr pas** :

   (a) de télécharger un programme Python

   (b) d'exécuter un programme Python sur le serveur

   (c) d'exécuter un programme Python sur le client

   (d) d'afficher une page html

   Réponse c. En effet, si un exécuteur Python n'est pas présent sur le serveur, taper cette URL aura pour effet de télécharger le programme (réponse a) ou de l'afficher dans le navigateur (réponse d). En revanche, si le serveur est doté d'un exécuteur Python, le fait de saisir l'URL aura pour effet de le lancer... mais côté serveur (réponse b), pas côté client.

3. Un élément `form` d'une page HTML contient un élément *button* de type *submit*. Un clic sur ce bouton :

   (a) envoie les données du formulaire vers la page définie par l'attribut action de l'élément `form`

   (b) efface les données entrées par l'utilisateur dans le formulaire

   (c) envoie les données du formulaire vers la page définie par l'attribut *method* de l'élément `form`

   (d) ne fait rien si un script javascript n'est pas associé au bouton

   Réponse a. C'est le principe des boutons de type *submit*.

Thème « Architectures matérielles et systèmes d'exploitation »

1. Parmi les affirmations suivantes, laquelle est vraie ?

   (a) La mémoire RAM est une mémoire accessible en lecture seulement

   (b) La mémoire RAM est une mémoire accessible en écriture seulement

   (c) La mémoire RAM est une mémoire accessible en lecture et en écriture

   (d) La mémoire RAM permet de stocker des données après extinction de la machine

   Réponse c. La mémoire RAM (Random Access Machine) sert à stocker le programme à exécuter ainsi que ses données. Elle est donc accessible en lecteur ET écriture, mais elle se vide à chaque extinction du système. Elle ne doit pas être confondue avec la *mémoire vive*, qui se présente souvent sous forme de barrette dans un ordinateur.

2. À partir du dossier `~/Doc/QCM`, quelle commande permet de rejoindre le dossier `~/.Hack/Reponses` ?

   (a) `cd ../../.Hack/Reponses`

   (b) `cd .Hack/Reponses`

   (c) `cd /~/.Hack/Reponses`

   (d) `cd /.Hack/Reponses`

   Réponse a. En effet, `cd ../` permet de remonter l'arborescence d'un niveau (donc après l'exécution de cette commande, on se trouve dans le répertoire `~/Doc/`). Donc avec `cd ../../`, on remonte de 2 niveaux et on se trouve dans `~/`. Il suffit ensuite d'insérer le chemin du répertoire dans lequel on veut se déplacer.

3. Que permet la commande *ping* lancé sur ce poste ?

   (a) de vérifier que l'on a accès à internet

   (b) de connaître l'adresse Ethernet d'une autre machine

   (c) de tester l'accessibilité d'une autre machine à travers un réseau IP

   (d) de saturer le réseau avec des requêtes ICMP

   Réponse c. Par exemple, `ping www.monsite.com` permet d'obtenir des résultats du type :

```
Envoi d'une requête 'ping' sur monsite.com
[2001:41d0:1:1b00:213:186:33:19] avec 32 octets de données:
Réponse de 2001:41d0:1:1b00:213:186:33:19 : temps=45 ms
Réponse de 2001:41d0:1:1b00:213:186:33:19 : temps=101 ms
Réponse de 2001:41d0:1:1b00:213:186:33:19 : temps=165 ms
Réponse de 2001:41d0:1:1b00:213:186:33:19 : temps=231 ms
Statistiques Ping pour 2001:41d0:1:1b00:213:186:33:19:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 45ms, Maximum = 231ms, Moyenne = 135ms
```

Thème « Langages et programmation »

1. a et m étant des entiers strictement positifs, la fonction suivante calcule a^m :

```
def puissance(a,m):
        p = 1
        n = 0
        while n < m:
                p = p * a
                #
                n = n + 1
        return p
```

À la ligne marquée d'un #, on a :

(a) $p = a^n$ (b) $p = a^{n-1}$ (c) $p = a^{n+1}$ (d) $p = a^m$

Réponse c. Il suffit de voir ce qui se passe pas à pas :

n < m	p	n
	1	0
Vrai	a	1
Vrai	a*a	2
Etc.		

La colonne orange indique la valeur de p au niveau de la la ligne marquée d'un #. On voit que pour $n = 0$, $p = a$. Pour $n = 1$, $p = a^2$. L'exposant de a est égal à $n+1$.

2. Voici une fonction Python de recherche d'un maximum :

```
def maxi(t):
        m = -1
        for k in range(len(t)):
                if t[k] > m:
                        m = t[k]
        return m
```

Avec quelle précondition sur la liste `t` la postcondition « `m` est un élément maximum de la liste `t` » n'est-elle pas assurée ?

(a) Tout élément de `t` est un entier positif ou nul

(b) Tout élément de `t` est un entier supérieur ou égal à -2

(c) Tout élément de `t` est un entier supérieur ou égal à -1

(d) Tout élément de `t` est un entier strictement supérieur à -2

Réponse b. En effet, on initialise la variable `m` à -1 et par suite, on effectue des tests de supériorité par rapport à `m`. Si `t` est une liste uniquement composée de -2, par exemple `t = [-2,-2,-2]`, alors ce programme ne fonctionne pas.

3. La fonction `indice_maxi` ci-dessous doit rechercher l'indice de la valeur maximale présente dans une liste de nombres et le renvoyer.
   Dans le cas où cette valeur maximale est présente plusieurs fois, c'est le plus petit de ces indices qui doit être renvoyé.

```
def indice_maxi(liste):
        valeur_max = liste[0]
        indice = 0
        for i in range(len(liste)):
                if liste[i] > valeur_max:
                        indice = i
        return indice
```

   Cette fonction a mal été programmée. On souhaite mettre en évidence son incorrection par un test bien choisi. Quelle valeur de test mettra l'erreur en évidence ?

(a) `[ 1, 2, 3, 4 ]`
(b) `[ 4, 3, 2, 1 ]`
(c) `[ 1, 3, 3, 2 ]`
(d) `[ 1, 1, 1, 1 ]`

Réponse c. Les listes des propositions (a) et (b) sont soit croissante, soit décroissante, ce qui ne permet pas de mettre en relief une anomalie. Quant à la liste de la proposition (d), elle est constante donc là encore, on ne détecte pas l'anomalie. Si on exécute le programme de la proposition (c), on a :

i	liste[i] > valeur_max	indice
0	1 > 1 : faux	0
1	3 > 1 : vrai	1
2	3 > 1 : vrai	2
3	2 > 1 : vrai	3

Le programme retourne donc « 3 »... ce qui est faux.

Thème « Algorithmique »

1. `A` désignant un entier, lequel des codes suivants ne termine pas ?

(a)
```
i = A + 1
while i < A:
        i = i-1
```

(c)
```
i = A - 1
while i < A:
        i = i - 1
```

(b)
```
i = A + 1
while i < A:
        i = i + 1
```

(d)
```
i = A - 1
while i < A:
        i = i + 1
```

Réponse c. En effet, dans ce code, on initialise `i` à `A-1` donc $i < A$. On entre donc dans la boucle et `i` diminue de 1. Donc $i < A$ encore une fois... On reste donc dans la boucle tout le temps car `i` sera toujours plus petit que `A`.

2. On considère la fonction suivante :

```
def f(t,i):
        im = i
        m = t[i]
        for k in range(i+1, len(t)):
                if t[k] < m:
                        im, m = k, t[k]
        return im
```

Que vaut l'expression `f([ 7, 3, 1, 8, 19, 9, 3, 5 ], 0)` ?

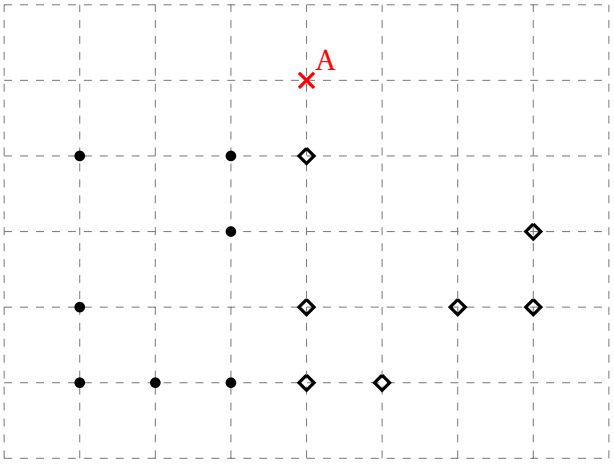
(a) 1 (b) 2 (c) 3 (d) 4

Réponse b. Exécutons pas à pas ce code :

k	t[k] < m	im	m
		0	7
1	3 < 7 : vrai	1	3
2	1 < 3 : vrai	2	3
3	8 < 3 : faux	2	3
4	19 < 3 : faux	2	3
etc.			

On recherche donc ici l'indice du premier minimum de la liste.

3. Dans le quadrillage ci-dessous, 14 points sont dessinés, dont 7 de la classe C1, avec des ronds noirs , et 7 de la classe C2, avec des losanges.



On introduit un nouveau point A, dont on cherche la classe à l'aide d'un algorithme des k plus proches voisins pour la distance géométrique habituelle, en faisant varier la valeur de k parmi 1, 3 et 5. Quelle est la bonne réponse (sous la forme d'un triplet de classes pour le triplet (1,3,5) des valeurs de k) ?

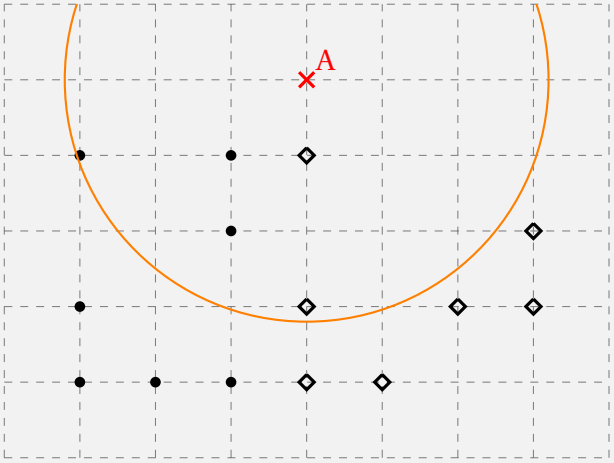
(a) C1, C2, C3
(b) C2, C1, C2
(c) C2, C2, C2
(d) C2, C1, C1

Réponse d. En effet,

— pour $k = 1$, le point le plus proche de A est le losange qui est juste en-dessous ; donc la classe est C2.

— pour $k = 3$, on cherche les 3 points les plus proches de A : il y a un triangle et deux cercles ; la classe prédominante est donc C1.

— pour $k = 5$, il y a 3 cercles et 2 triangles :



La classe prédominante est donc C1.

Ce sujet est fourni à titre d'exemple par l'Éducation Nationale.
L'épreuve finale de 1re NSI (pour les élèves ne souhaitant pas garder cette spécialité en Terminale) ne ressemblera pas nécessairement trait pour trait à ce sujet, mais sera tout de même composée de QCM de ce type.
Le corrigé est de l'auteur de ce document.