ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communications
Cours d'Information, Calcul et Communication, sections MA et PH
Chappelier J.-C.

Information, Calcul et Communication (SMA/SPH)
Correction de l'Examen

12 janvier 2021

INSTRUCTIONS (à lire attentivement)

IMPORTANT ! Veuillez suivre les instructions suivantes à la lettre sous peine de voir votre examen annulé dans le cas contraire.

1. Vous disposez de trois heures pour faire cet examen (16h15 – 19h15).
2. Vous devez **écrire à l'encre noire ou bleu foncée**, pas de crayon ni d'autre couleur.
   N'utilisez **pas non plus de stylo effaçable** (perte de l'information à la chaleur).
3. Vous avez droit à toute documentation papier.
   En revanche, vous ne pouvez pas utiliser d'ordinateur personnel, ni de téléphone portable, ni aucun autre matériel électronique.
4. Répondez aux questions directement sur la donnée.
   Ne joignez aucune feuilles supplémentaires ; **seul ce document sera corrigé**.
5. Lisez attentivement et *complètement* les questions de façon à ne faire que ce qui vous est demandé. Si l'énoncé ne vous paraît pas clair, ou si vous avez un doute, demandez des précisions à l'un des assistants.
6. L'examen comporte 8 exercices indépendants sur 12 pages, qui peuvent être traités dans n'importe quel ordre, mais qui ne rapportent pas la même chose (les points sont indiqués, le total est de 150 points).
   Tous les exercices comptent pour la note finale.

Question 1 – Un p'tit algo [10 points]

Un étudiant a écrit l'algorithme suivant :

Machin

entrée : a, *entier strictement positif*
sortie : *? ? ?*

$b \longleftarrow 0$
$c \longleftarrow 1$
$d \longleftarrow 0$
Tant que $c < a$
  $d \longleftarrow d + 1$
  Si d est un multiple de 7 ou de 11
    $b \longleftarrow b + d$
    $c \longleftarrow c + 1$
Sortir : b

① **[4 points]** Quelle valeur sort cet algorithme pour $a = 7$? Expliquez cette valeur en une phrase/formule.

② **[6 points]** Ecrire une fonction C++ qui implémente cet algorithme.

Réponse ① : somme des **6** (attention à l'inégalité stricte) premiers multiples communs :
$103 = 7 + 11 + 14 + 21 + 22 + 28$.

Réponse ② :

```
1   unsigned int f(unsigned int a)
2   {
3     unsigned int b(0);
4     unsigned int c(1);
5     unsigned int d(0);
6     while (c < a) {
7       ++d;
8       if ((d % 7 == 0) or (d % 11 == 0)) {
9         b += d;
10        ++c;
11      }
12    }
13    return b;
14  }
```

Question 2 – Ecriture d'algorithmes [41 points]

① **[15 points]** Ecrivez une **fonction C++** qui effectue la fusion triée de deux listes triées.

Par exemple, si en entrée on a les listes $L_1 = (-5, 1, 4, 18)$ et $L_2 = (-12, -6, 8, 33, 107)$, alors en sortie on aura la liste $(-12, -6, -5, 1, 4, 8, 18, 33, 107)$.

② **[5 points]** Quelle est la complexité de l'algorithme correspondant à votre fonction C++ ?

Réponses : ① Il y a plein de façons d'écrire un tel programme. En voici une itérative (avec `liste` un `vector<int>` ; ils peuvent utiliser les `deque` s'ils les connaissent) :

```
liste fusion_triee(liste const& L1, liste const& L2)
{
  liste retour;

  size_t i1(0), i2(0);
  while ((i1 < L1.size()) or (i2 < L2.size())) {
    // Note : on pourrait faire une fonction de cette boucle interne...
    while ((i1 < L1.size()) and ((i2 >= L2.size()) or (L1[i1] <= L2[i2]))) {
      retour.push_back(L1[i1]);
      ++i1;
    }
    while ((i2 < L2.size()) and ((i1 >= L1.size()) or (L2[i2] <= L1[i1]))) {
      retour.push_back(L2[i2]);
      ++i2;
    }
  }

  return retour;
}
```

et une récursive :

```
liste fusion_triee(liste& L1, liste& L2)
{
  if (L1.empty()) return L2;
  if (L2.empty()) return L1;

  liste retour;
  auto valeur(L1.back());

  if (L1.back() > L2.back()) {
    L1.pop_back();
  } else {
    valeur = L2.back();
    L2.pop_back();
  }
  retour = fusion_triee(L1, L2);
  retour.push_back(valeur);

  return retour;
}
```

Note : évidemment l'aspect non-`const` des paramètres de cette version n'est pas terrible. On pourra pour cela fournir un « *wrapper* » qui effectue une copie préalable. Voir pour cela le commentaire de la solution à ③.

② La taille n de l'entrée de cet algorithme est la somme des taille de L_1 et L_2. Parcourant chacune des listes une seule fois, la complexité temporelle pire cas de cet algorithme est en $\Theta(n)$.

③ **[21 points]** Ecrivez une **fonction C++** ***récursive*** qui, prennant en entrée deux listes quelconques (y compris vides), effectue leur fusion alternée, « poussée à droite » en cas d'éléments excédentaires dans une des deux listes.

Par exemple :

- si en entrée on a les deux listes $L_1 = (1, 2, 3)$ et $L_2 = (-11, -22, -33)$ (dans cet ordre), alors en sortie on aura la liste $(1, -11, 2, -22, 3, -33)$;
- si en entrée on a les deux listes $L_1 = (1, 2, 3, 4)$ et $L_2 = (-11, -22, -33)$ (dans cet ordre ; L_1 plus grande), alors en sortie on aura la liste $(1, 2, -11, 3, -22, 4, -33)$;
- par contre, si en entrée on a les deux listes dans l'autre ordre : $L_1 = (-11, -22, -33)$ et $L_2 = (1, 2, 3, 4)$, alors en sortie on aura la liste $(1, -11, 2, -22, 3, -33, 4)$;
- encore deux exemples si nécessaire : si en entrée on a les deux listes : $L_1 = (-11, -22)$ et $L_2 = (1, 2, 3, 4, 5, 6)$, alors en sortie on aura la liste $(1, 2, 3, 4, -11, 5, -22, 6)$;
- enfin, si en entrée on a les deux listes dans l'autre ordre : $L_1 = (1, 2, 3, 4, 5, 6)$ et $L_2 = (-11, -22)$, alors en sortie on aura la liste $(1, 2, 3, 4, 5, -11, 6, -22)$.

Réponse :

Voici une solution possible (avec `liste` un `vector<int>` ; ils peuvent utiliser les `deque` s'ils les connaissent) :

```
liste fusion_alt_rec(liste& L1, liste& L2)
{
  if (L1.empty()) return L2;
  if (L2.empty()) return L1;

  const auto first (L1.back());
  L1.pop_back();
  const auto second(L2.back());
  L2.pop_back();

  liste retour(fusion_alt_rec(L1, L2));
  retour.push_back(first);
  retour.push_back(second);

  return retour;
}
```

Note : évidemment l'aspect non-`const` des paramètres de cette version n'est pas terrible ; on pourrait pour cela fournir un « *wrapper* » qui effectue une copie préalable :

```
// content preserving version: working copies made by pass-by-value
liste fusion_alt(liste L1, liste L2)
{
  return fusion_alt_rec(L1, L2);
}

// another content preserving version
liste fusion_alt2(liste const& L1, liste const& L2)
{
  liste L1bis(L1);
  liste L2bis(L2);
  return fusion_alt_rec(L1bis, L2bis);
}
```

Question 3 – Compression de génome [20 points]

Une entreprise de séquençage de génome veut stocker efficacement ses résultats sur disque.
Les séquences à coder n'utilisent que 5 lettres : A, C, G, T et l'espace (pour dire « inconnu » ou « coupé », notée ici ␣).

Par exemple (20 lettres) :

T A G C G ␣ C G C C T A C G ␣ C A T G T

① **[7 points]** Quelle est l'entropie (en bit) de la séquence ci-dessus ?

Donnez votre réponse sous la forme $a + b\,\log_2(3) + c\,\log_2(5)$ en exprimant la valeur de a, de b et de c sous forme de fraction irréductible :

$$a = \frac{6}{5} \qquad b = -\frac{9}{20} \qquad c = \frac{3}{4}$$

et **justifiez** votre réponse :

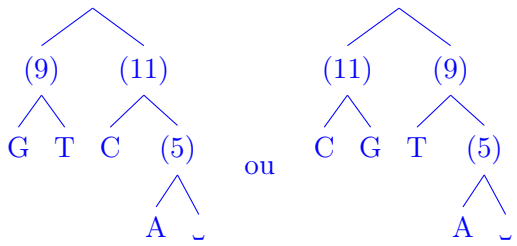
A	C	G	T	␣
3	6	5	4	2

(= | C | G | T | A | ␣ | (pour Huffman))
|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 2 |

$$\begin{aligned}
H(X) &= \sum_x p_x \log_2(\frac{1}{p_x}) = p_A \log_2(\frac{1}{p_A}) + p_C \log_2(\frac{1}{p_C}) + p_G \log_2(\frac{1}{p_G}) + p_T \log_2(\frac{1}{p_T}) + p_␣ \log_2(\frac{1}{p_␣}) \\
&= \frac{3}{20}\log_2\frac{20}{3} + \frac{6}{20}\log_2\frac{20}{6} + \frac{5}{20}\log_2\frac{20}{5} + \frac{4}{20}\log_2\frac{20}{4} + \frac{2}{20}\log_2\frac{20}{2} \\
&= \log_2 20 - \frac{3}{20}\log_2 3 - \frac{6}{20}\log_2 6 - \frac{5}{20}\log_2 5 - \frac{4}{20}\log_2 4 - \frac{2}{20}\log_2 2 \\
&= 2 + \log_2 5 - \frac{3}{20}\log_2 3 - \frac{3}{10} - \frac{3}{10}\log_2 3 - \frac{5}{20}\log_2 5 - \frac{1}{5}\times 2 - \frac{1}{10} \\
&= \frac{6}{5} - \frac{9}{20}\log_2 3 + \frac{3}{4}\log_2 5
\end{aligned}$$

② **a) [5 points]** Proposez un code de Huffman binaire pour cette séquence. Donnez ici votre arbre de codage :

Par exemples (tout arbre donnant la distribution de longueurs ci-contre est correct) :



Ce qui donne les longueurs suivantes :

- C : 2
- G : 2
- T : 2
- A : 3
- ␣ : 3

b) [1.5 points] Si vous codez cette séquence avec votre code, combien avez-vous de bits au total ?

$(6 + 5 + 4) \times 2 + (3 + 2) \times 3 = 30 + 15 = 45$ bits.

c) [2 points] Quel taux de compression obtenez vous par rapport à un codage binaire naïf ? Justifiez votre réponse.

5 lettres nécessitent 3 bits pour être différenciées en binaire ; un message de 20 lettres utilise alors 60 bits en tout.

On a donc un taux de compression de $1 - 45/60 = 25\%$

③ **[4.5 points]** La vraie entropie de leurs séquences est de 2.5 bit. Quelle taille de disque (en Gio) doivent ils prévoir (environ) pour les 10 prochaines années sachant qu'ils produisent en moyenne par année 10^5 séquences de longueur moyenne 10^9 caractères ?

Justifiez votre réponse.

Ils produisent donc environ 10^{14} caractères par an, soient 10^{15} caractères sur 10 ans.

Si L est la longueur moyenne du code des caractères par code de Huffman (on ne souhaite pas avoir de perte dans cette situation !), nous savons que, en général, L est comprise entre H et $H+1$, donc entre 2.5 et 3.5 bits. Mais ici on sait aussi que le code binaire naïf (le pire) est de 3 bits ; il faut donc prévoir entre 2.5 et 3 bits par caractères, soit au maximum $3 \cdot 10^{15}$ bits, soient $0.375 \cdot 10^{15}$ octets.

Sachant que 1 Gio c'est environ 10^9 octets, il faut donc prévoir environ $\simeq 4 \cdot 10^5$ Gio, soit environ 400 Tio.

Réponses acceptables : entre $3.50 \cdot 10^5$ (réponse exacte sans approximation) et $4 \cdot 10^5$.

Question 4 – Complexité [15 points]

Considérez le programme suivant :

```
unsigned int f(unsigned int i, unsigned int j)
{
  if (i <= j) return j;
  unsigned int a(2*i - j);
  while (i > j) {
    a += 3;
    --i;
  }
  return a + f(j, a) + 2;
}
```

① **[3 points]** Calculer `f(6, 4)`.

② **[12 points]** Quelle est la complexité de l'algorithme implémenté par `f` ?
Justifier votre réponse.

Réponses :

① Comme $6 > 4$, on passe le premier `if`. `a` est initialisé à `8`, puis vaut `14` (cf ci-dessous).
La valeur est donc `14 + f(4, 14) + 2`, soit $14 + 14 + 2 = \mathbf{30}$.

Le seul but de cette question est de vous « faire entrer » dans l'algorithme.

② L'algorithme s'arrête si $i <= j$ (donc $\Theta(1)$ dans ce cas, en considérant comme toujours pour ce cours les opérations arithmétiques comme élémentaires).

Si $i > j$, a vaut alors $(2i - j) + 3 \times (i - j) = 5i - 4j$, puisque la boucle se fait autant de fois que $i - j$. Le contenu de la boucle elle-même est en $\Theta(1)$ (en considérant comme toujours pour ce cours les opérations arithmétiques comme élémentaires).

On rappelle donc $f(j, 5i - 4j)$, mais $j <= 5i - 4j$ puisque $i > j$, donc cet appel retourne j directement.

Au final, la complexité est en $\Theta(i - j)$, « le » pire cas correspondant à $i > j$.

Note : on peut aussi considérer comme pire cas $i > 0$ et $j = 0$ et dire que la complexité est en $\Theta(i)$.

Question 5 – RSA [13 points]

Nous considérons utiliser RSA avec $p = 7$ et $q = 13$ (et donc $n = 91$).

① **[3.5 points]** Dans ce cadre, est-ce que les valeurs suivantes sont des valeurs possibles pour notre clé privée d ? Justifiez à chaque fois votre réponse.

$d = 15$: **non** : 3 est facteur commun avec $m = 6 \times 12 = 72 = 2^3 \times 3^2$;

$d = 35$: **ok** ;

$d = 85$: **non** : supérieur à m.

② **[2 points]** Nous choisissons finalement comme clé privée $d = 29$. Vérifiez alors, en explicitant ici vos calculs, que notre clé publique est $e = 5$.

Justification : On vérifie qu'ils sont inverse l'un de l'autre modulo m : $29 \times 5 = 145 = 1 + 144 = 1 + 2 \times 72$

③ **[3 points]** Quelqu'un veut nous envoyer le message correspondant à la valeur (décimale) 10.
Quel message (valeur décimale) nous envoit-il pour assurer la confidentialité de ce message ?
Justifiez pleinement votre réponse (en explicitant tous les calculs si nécessaire).

Résponse et justification : $10^5 \mod 91 = 10 \times 100 \times 100 = 10 \times 9 \times 9 = -9$ (là, il y a au moins 2 façons d'arriver à -9) $= 82$ (modulo 91).

④ **[3 points]** Nous décidons de changer de clé et utilisons maintenant la clé publique $(5'239, 1'863'227)$ avec la clé privée $11'719$.

Supposons que nous recevions le message crypté suivant, en binaire : `11011110`.
Quel message nous a-t-on envoyé ?
Donnez ici simplement la formule (en décimal), *sans* calculer explicitement la valeur.

Réponse : Dans RSA, nous travaillons avec des nombres entiers positifs (modulo n), donc le pattern binaire doit se décoder comme un entier positif : c'est 222.

Le décodage est alors : $222^{11719} \mod 1863227$ $(= 824645)$

⑤ **[1.5 points]** Toujours avec la paire de clés $(5'239, 1'863'227)$ et $11'719$, supposons que nous voulions signer un message correspondant à la valeur (décimale) $12'345$.
Quelle signature utilisons nous ? (uniquement la signature, pour responsabilisation, *sans* souci de confidentialité ici).
Donnez ici simplement la formule (en décimal), *sans* calculer explicitement la valeur.

Réponse : $12345^{11719} \mod 1863227$ $(= 1721268)$

Question 6 – La bonne droite [25 points]

On s'intéresse dans cet exercice à écrire un programme C++ pour faire passer une droite au mieux au milieu d'un nuage de points dans le plan.

① **[1 point]** Commencez par écrire ici un type C++, nommé `Point` permettant de modéliser un point dans le plan (représenté par deux nombres réels x et y).

```
struct Point {
  double x;
  double y;
};
```

② **[1 point]** Écrivez ensuite ici un type C++, nommé `Line` permettant de modéliser une droite d'équation $y = a\,x + b$ par ses deux coefficients : il suffit simplement de représenter a et b.

```
struct Line {
  double a;
  double b;
};
```

③ **[8 points]** On veut maintenant écrire une fonction C++ `fit()` qui reçoit en entrée une liste de n points et qui donne en sortie les paramètres a et b de la droite d'équation $y = a\,x + b$ qui passe « au mieux » au milieu de cet ensemble de points.

Si l'on note x_i et y_i les coordonnées du point P_i de la liste (i de 1 à n), les coefficients a et b de cette droite sont déterminés comme suit (b dépend de a) :

$$a = \frac{\left(\sum_{i=1}^{n} x_i\, y_i\right) - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\left(\sum_{i=1}^{n} {x_i}^2\right) - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2} \qquad b = \frac{1}{n}\left(\left(\sum_{i=1}^{n} y_i\right) - a\,\left(\sum_{i=1}^{n} x_i\right)\right)$$

Ecrivez ici le code C++ de la fonction `fit()`. **Celle-ci ne doit contenir qu'<u>*une*</u> *seule* boucle.**

```
typedef vector<Point> Cloud;

Line fit(Cloud const& data)
{
  if (data.empty()) return { 0.0, 0.0 };

  double Sxy(0.0);
  double Sx2(0.0);
  double Sx (0.0);
  double Sy (0.0);

  for (auto p : data) {
    Sxy += p.x * p.y;
    Sx2 += p.x * p.x;
    Sx  += p.x;
    Sy  += p.y;
  }

  double a(0.0);
  const double diviseur( Sx2 - (Sx * Sx) / data.size() );
  if (abs(diviseur) > 1e-12) {
    a = ( Sxy - (Sx * Sy) / data.size() ) / diviseur ;
  } else {
    return { 0.0, 0.0 };
  }
  return { a, (Sy - a * Sx) / data.size() };
}
```

④ **[1 point]** Ecrivez une fonction C++ `line_value()` qui prend en paramètres une `Line` et un nombre réél x et retourne la valeur $y = a\ x + b$ obtenue à partir de x et des coefficients de la droite passée en argument.

```
double line_value(Line const& line, double x)
{
  return line.a * x + line.b;
}
```

⑤ **[5 points]** Ecrivez une fonction C++ `mse()` qui prend en paramètres une `Line` (a, b) et un ensemble de points et qui calcule l'écart entre la droite et l'ensemble de points suivant la formule :

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - (a\ x_i + b))^2$$

(où x_i et y_i sont les coordonnées du point P_i de la liste ; avec i de 1 à n).

```
double mse(Line const& line, Cloud const& data)
{
  if (data.empty()) return 0.0;

  double s(0.0);
  for (auto p : data) {
    const double val(p.y - line_value(line, p.x));
    s += val * val;
  }
  return s / data.size();
}
```

⑥ **[9 points]** Pour finir, on souhaite vérifier empiriquement avec du code C++ que la droite calculée en ③ est celle qui minimise l'écart (tel que calculé en ⑤).

Ecrire pour cela une fonction C++ `check_best_fit()` qui prend en paramètres une liste de points, un nombre réél ε et un entier m, et qui :

1. calcule la droite obtenue par `fit()` ; appelons $\widehat{a}$ et $\widehat{b}$ ses paramètres ;
2. calcule et affiche l'écart (fonction `mse()`) entre l'ensemble des points et chacune des $(2m+1)^2$ droites définies comme suit (suite et exemple au dos) :

suite au dos ☞

a) le paramètre de pente a de ces droites variera entre $(\widehat{a} - m\ \varepsilon)$ et $(\widehat{a} + m\ \varepsilon)$ par pas de ε (exemple ci-dessous) ;

b) et le paramètre d'ordonnée à l'origine b de ces droites variera entre $(\widehat{b} - m\ \varepsilon)$ et $(\widehat{b} + m\ \varepsilon)$, également par pas de ε.

Par exemple, avec $\varepsilon = 0.1$ et $m = 2$, la fonction `check_best_fit()` testera les 25 droites suivantes : toutes les droites $y = a\ x + b$ avec a parmi $\{\widehat{a} - 0.2, \widehat{a} - 0.1, \widehat{a}, \widehat{a} + 0.1, \widehat{a} + 0.2\}$ et b parmi $\left\{\widehat{b} - 0.2, \widehat{b} - 0.1, \widehat{b}, \widehat{b} + 0.1, \widehat{b} + 0.2\right\}$. A noter donc que la droite calculée par `fit()` est testée au milieu de toutes ces droites.

Vous pouvez faire l'affichage comme vous voulez ; par exemple, afficher sur chaque ligne le a, le b puis l'écart (`mse()`) trouvé.

Code de la fonction `check_best_fit()` :

Voici une possibilité :

```
void check_best_fit(Cloud const& data, double step, int m)
{
  const Line best(fit(data));

  for (int i(-m); i <= m; ++i) {
    const double a(best.a + i*step);

    for (int j(-m); j <= m; ++j) {
      const double b(best.b + j*step);
      cout << a << ", " << b << " : "
           << mse({a, b}, data) << endl;
    }
  }
}
```

et en voici une autre :

```
void check_best_fit(Cloud const& data, double step, int m)
{
  const Line best(fit(data));

  // sort les invariants
  const double a_last (best.a + m*step);
  const double b_first(best.b - m*step);
  const double b_last (best.b + m*step);

  for (double a(best.a - m*step); a <= a_last; a += step)
    for (double b(b_first); b <= b_last; b += step)
      cout << a << ", " << b << " : " << mse({a, b}, data) << endl;
}
```

Question 7 – Mal aux sinus ? [16 points]

① **[9 points]** Soit f une fonction de $\mathbb{R}$ dans $\mathbb{R}$ définie par

$$f(t) = \sum_{m\in\mathbb{Z}} a_m \,\mathrm{sinc}(35\,t - m)$$

où $\mathrm{sinc}(x) = \dfrac{\sin(\pi\,x)}{\pi\,x}$ et $a_m = K_1 \sin(\frac{4\pi}{7}m) + K_2 \sin(\frac{11\pi}{6}m + \frac{\pi}{5}) + K_3 \sin(\frac{6\pi}{5}m + \frac{\pi}{3}) + K_4 \sin(\frac{\pi}{4}m + \frac{\pi}{3})$.
Proposez des valeurs K_1, K_2, K_3, K_4 pour pouvoir écrire $f(t)$ comme la somme de *deux* fonctions sinus, puis écrivez $f(t)$ comme la somme de deux fonctions sinus :

$K_1 = \ldots$ (n'importe quoi) $\qquad K_2 = 0 \qquad K_3 = 0 \qquad K_4 = \ldots$

$$f(t) = K_1 \sin(20\pi\,t) + K_4 \sin(\frac{35\pi}{4}\,t + \frac{\pi}{3})$$

Justifiez votre réponse :

Voyons f comme la reconstruction d'un signal échantillonné (à une fréquence d'échantillonnage de $f_e = 35$ Hz), les a_m étant les échantillons d'un signal $X(t)$: $a_m = X(m\,T_e) = X(\frac{m}{f_e})$, c.-à-d.

$$X(t) = K_1 \sin(2\pi\,\frac{2f_e}{7}\,t) + K_2 \sin(2\pi\,\frac{11f_e}{12}\,t + \frac{\pi}{5}) + K_3 \sin(2\pi\,\frac{3f_e}{5}\,t + \frac{\pi}{3}) + K_4 \sin(2\pi\,\frac{f_e}{8}\,t + \frac{\pi}{3})$$

Si tel est le cas, pour pouvoir ré-écrire f directement à partir de l'écriture de X, il faut que la bande passante de X soit strictement inférieure à $\frac{f_e}{2}$. Or les 4 fréquences présentes dans X sont (dans l'ordre d'écriture ci-dessus) : $\frac{2}{7}f_e$, $\frac{11}{12}f_e$, $\frac{3}{5}f_e$ et $\frac{1}{8}f_e$. Seules la première et la dernière sont strictement inférieures à $\frac{f_e}{2}$. Donc si K_2 et K_3 ne sont pas nuls, cela rendra notre démarche bien plus compliquée (voire impossible en l'état des connaissances de ce cours). Par contre, si on les pose égaux à 0, alors notre démarche est possible et f s'écrit bien la somme de deux fonctions sinus.

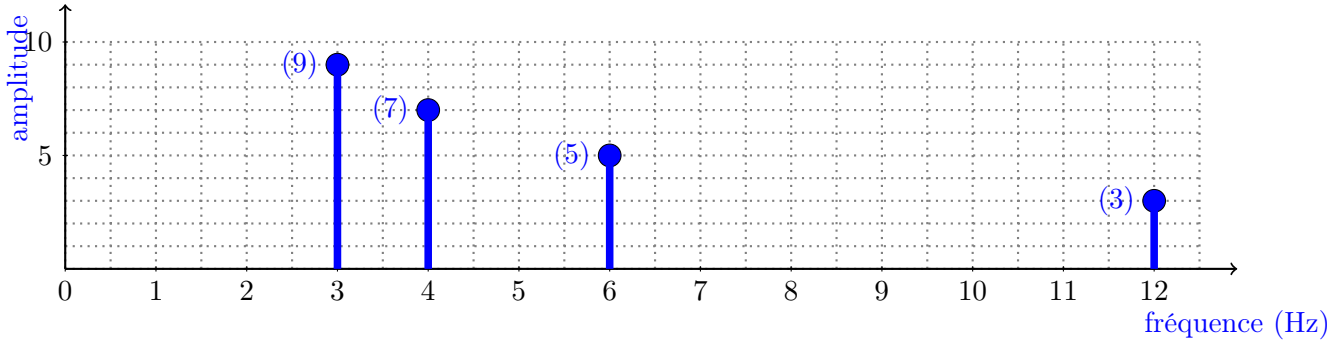
Note : considérer la fonction nulle (c.-à-d. tous les K_i nuls) comme la somme de *deux* (emphatique) fonctions sinus est vraiment tiré par les cheveux : c'est autant une somme d'aucune qu'une somme d'une infinité de fonctions sinus (que n'importe quoi d'autre entre deux)...

② On considère le signal

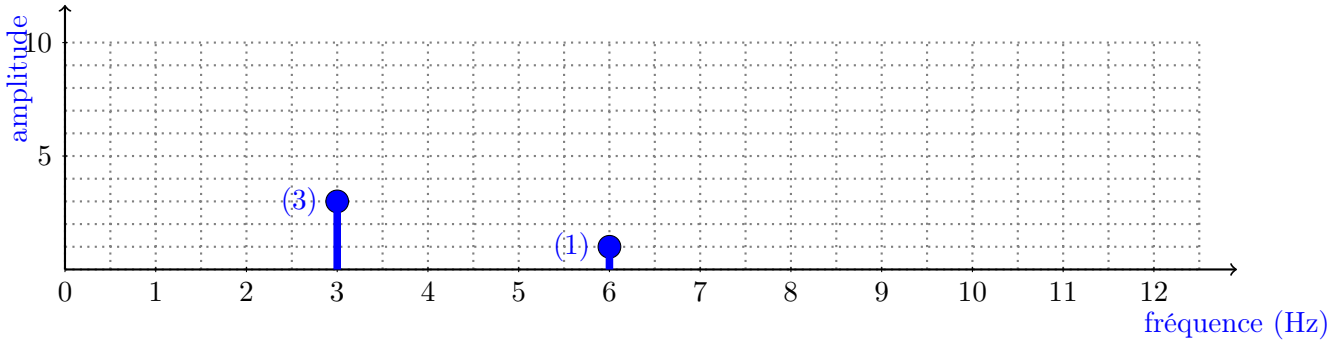
$$X(t) = 5\sin(12\pi\,t + \frac{2\pi}{3}) + 3\sin(24\pi\,t - \frac{\pi}{6}) + 7\sin(8\pi\,t + \frac{3\pi}{4}) + 9\sin(6\pi\,t - \frac{3\pi}{5})$$

que l'on passe dans un filtre à moyenne mobile de fréquence de coupure $f_c = 4$ Hz.

a) [3 points] Dessinez le spectre de X *avant* filtrage :



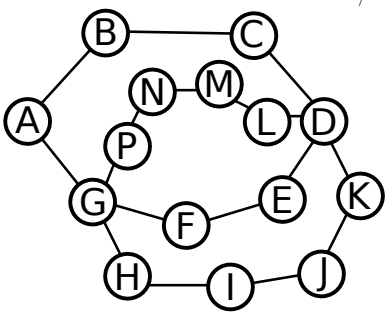
b) [4 points] Dessinez le spectre de $\widehat{X}(t)$, signal *après* filtrage, sachant que $\text{sinc}(\frac{3}{4}) \simeq 0.33$ et $\text{sinc}(\frac{3}{2}) \simeq -0.2$:



Les multiples de 4 Hz sont totalement annulés.

Question 8 – Des routes [10 points]

On considère le réseau TCP/IP suivant :



① **[2 points]** Donnez les lignes de la table de routage de A correspondant aux nœuds D, E, L et I :

destination	route	distance
D	B	3
E	G	3
L	B	4
I	G	3

② **[1 point]** Parmi les lignes précédentes (de la table de routage de A) lesquelles changent, et comment, si le nœud C « tombe » (c.-à-d. disparait) ?

Seules les lignes de D et L changent (il ne reste de toutes façons que G pour partir de A plus loin que B !) :

destination	route	distance
D	G	4
L	G	5

③ **[7 points]** On envoie un message de A vers K (avec C présent). Ce message est décomposé en 80 paquets. Sachant que

- chaque nœud met 8 ms à transmettre un paquet au nœud suivant ;
- on négligera tout autre temps ;
- en moyenne 1 paquet sur 5 est malheureusement perdu (au niveau 4) ;

quel est (en moyenne) le temps total de transmission de ce message ?

Justifiez votre réponse. **Note :** on pourra, si nécessaire, faire l'approximation $9 \simeq 10$.

À chaque fois que A envoie x paquets à K :

- 1/5 de ces paquets est perdu, ce dont A se rend compte par le fait que K n'a pas envoyé de paquet d'acquittement ;
- **mais** 1/5 des $(1-1/5)\cdot x$ paquets d'acquittement renvoyés par K dont **aussi** perdus, donc A considère les paquets de départ correspondants aussi comme perdus.

Donc au final pour x paquets, A doit renvoyer $x/5 + x(1-1/5)/5 = (2\times 5-1)x/5^2 = 9x/25$ paquets.

(**Non demandé :**) et cela tant que A doit renvoyer des paquets, donc un nombre $n+1$ de fois tel que $(9/25)^n \times 80 = 1$, soit $n = -\log 80/\log(9/25) + 1 = 5$ fois.

Du point de vue du temps, comme il y a 4 nœuds de transmission, chaque paquet met 4×8 ms à aller de A à K (et réciproquement, pour l'acquittement).

Si l'on néglige la latence des 4 derniers paquets [1], le temps total est donc $N \times 4 \times 8$ ms, où N est le nombre total de paquets à envoyer. Pour nous :

$$N = 80 \times \left(1 + \frac{9}{25} + \left(\frac{9}{25}\right)^2 + \left(\frac{9}{25}\right)^3 + ...\right)$$

Si on fait l'approximation suggérée [2] : $\frac{9}{25} \simeq \frac{10}{25} = 0.4$, on obtient :

$$N = 80 \times (1 + 0.4 + 0.16 + 0.064 + ...)$$

1. Pour tous les paquets avant ces 4 derniers, A peut tout de suite renvoyer le paquet manquant (s'il y en a). Par contre, si l'un de ces 4 derniers paquets devait manquer, il faut que A attende 1,2,3 ou 4 cycle de 8 ms avant de s'en rendre compte et renvoyer le paquet manquant. Il y a ici une latence de réaction ; que nous négligeons.
2. Et sinon on pourra reconnaître la somme des termes d'une suite géométrique...

dont l'approximation 1.56 nous suffira ici (pas de calculette à l'examen ; note : on tolère jusqu'à 1.67, somme de la suite géométrique de raison 0.4 ; on tolère aussi, bien sûr, ceux qui ont pu calculer $\left(1 + \frac{9}{25} + \left(\frac{9}{25}\right)^2\right) \simeq 1.49$.)

Le temps total est donc $80 \times \lambda \times 4 \times 8$ ms (avec λ toléré entre 1.49 et 1.67), soit entre 3815 ms et 4275 ms, 4 s étant donc un bon ordre de grandeur.