Polytechnique de Montréal

MTH2302A
Probabilités et Statistique

Devoir 1

Gabriel-Andrew Pollo-Guilbert (1837776)

Remis à
Simon Demontigny

6 juin 2017

1 Deux par deux

Dans ce problème, on alloue des sièges à B voyageurs solitaires en premier et ensuite à A paires. On s'intérèsse aux cas où les D paires ne peuvent pas être ensemble.

L'avion est constitué de C rangées contenant chacune 2 sections de 3 sièges. La figure 1 représente la répartition des sièges dans l'avion.

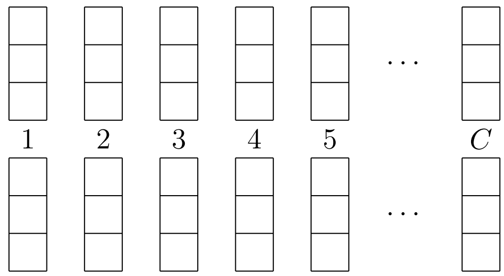

FIGURE 1 – répartition des sièges dans l'avion

On cherche à placer les voyageurs seuls de manière à ce qu'il reste exactement $A - D$ sections pouvant contenir des paires. Pour chaque section, il existe 5 manières d'empêcher une paire d'être ensemble. La figure 2 montre ces 5 cas.

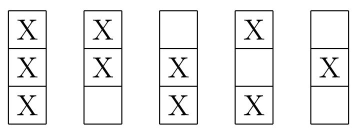

FIGURE 2 – 5 cas pouvant empêcher une paire d'être ensemble

Si l'on commence à la première section, on peut choisir une des 5 manières de bloquer la formation d'une paire. À la deuxième section, on choisie à nouveau une des 5 manières. À la troisième aussi et ainsi de suite. Par contre, il faut s'assurer qu'il reste des voyageurs à placer et aussi de laisser $A - D$ sections pour les paires.

Pour visualiser le problème, on peut dessiner un arbre pour chaque étape possible. Pour chaque feuille, on dénote l'état (c, n, s) où c est le nombre de sections manquantes à bloquer, n est le nombre restant de voyageurs seuls et s le nombre de sections à laisser. À chaque feuille, il y a une branche possible pour chaque cas pouvant bloquer une paire. La figure 3 représente une telle feuille.

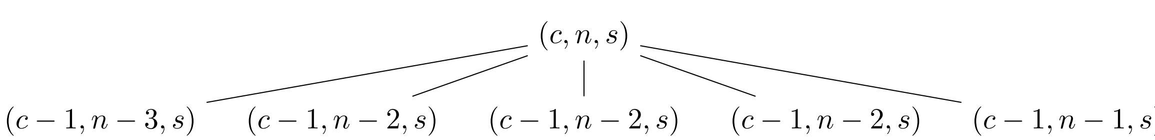

FIGURE 3 – feuille d'un arbre (c, n, s)

En effet, le nombre de sections restantes à bloquer diminue à chauque étape. De plus, le nombre restant de voyageurs diminue en fonction du cas pris. On remarque que trois branches se répète, c'est-à-dire celles où deux voyageurs sont utilisés pour bloquer une paire. Pour chaque feuille, on peut définir une fonction de récurrence comme suit

$$f(c,n,s) = f(c-1,n-1,s) + 3f(c-1,n-2,s) + f(c-1,n-3,s).$$

Pour définir la fin de la récurrence, il faut regarder un exemple. Soit 5 voyageurs qui doivent être placer sur 3 sections afin de permettre une paire libre. La figure 4 montre quelques cas possibles lorsque le paramètre $c = 0$ dans la récurrence, c'est-à-dire qu'il ne reste plus de sections à bloquer. Les X dénotent deux voyageurs seuls tandis que les P la paire qui doit être ensemble.

	X	P
X	X	P

X	X	P
X	X	P
X	X	

X	X	P
X	X	P
X		

X		P
X	X	P
X		

FIGURE 4 – exemples de cas finals

Lorsque $c = 0$, il faut finir d'allouer des sièges aux voyageurs restants. On ne peut pas leur donner les sièges restants des cas bloquants, car ce sont des possibilités calculées à priori par une autre branche. Par conséquent, on peut seulement allouer les sièges à coté des paires ensembles.

Dans certains cas, la branche est impossible car il n'y a pas assez de sièges disponibles pour les voyageurs ou il y plus de sièges alloués que de voyageurs seuls. Ces conditions se remarque si $n < 0$ ou $n > s$, car il y a un siège restant pour chaque paire ensemble.

Dans certains cas, il ne reste plus de voyageurs sans siège, alors la branche est valide. Lorsqu'il y reste des voyageurs et qu'il y a assez de sièges libres à coté des paires ensembles, il faut regarder un autre exemple.

Soit 3 voyageurs qui doivent être placés sur 4 sections afin de permettre trois paires d'être ensemble. La figure 5 montre des exemples respectant ces contraintes, avec R dénotant les voyageurs restants après avoir remplie les sections bloquées.

	P	P	P
X	P	P	P
	R	**R**	

	P	P	P
X	P	P	P
	R		**R**

	P	**R**	P
X	P	P	P
	R	P	

	R	**R**	P
X	P	P	P
	P	P	

FIGURE 5 – exemples de cas finals

Dans ces cas, il y a 2 voyageurs restants. Ils peuvent être à coté de n'importe quelle paire. Par conséquent, il faut choisir 2 paires parmis les 3. De plus, chaque voyageur restant peut être à droite ou à gauche de la paire. Par conséquent, le nombre de possibilités de d'allouer des sièges aux voyageurs restants est donné par $2^2\mathcal{C}_3^2$.

En général, on peut définir le système d'équations à récurrence suivant,

$$\begin{cases} f(0,0,s) = 1 \\ f(0,n,s) = \begin{cases} 2^n\mathcal{C}_s^n & \text{si } 0 \leq n \leq s \\ 0 & \text{sinon} \end{cases} \\ f(c,n,s) = f(c-1,n-1,s) + 3f(c-1,n-2,s) + f(c-1,n-3,s) \end{cases}$$

pour calculer le nombre de possibilités à partir de n'importe quelle feuille.

Dans notre cas, il faut placer 12 voyageurs solos afin de laisser 28 paires ensembles et 2 paires séparées dans les 18 rangées (36 sections). Il y a donc $36 - 28 = 8$ sections bloquées. Le nombre de possibilités est donné par $f(8, 12, 28)$. Ceci dit, le calcul prend juste en compte les 8 premières rangées comme étant bloquées. Hors, il faut choisir 8 rangées bloquées parmis les 36 disponibles. On ajuste le calcul de sorte que le nombre de possibilités est donné par $\mathcal{C}_{36}^{8} f(8, 12, 28)$.

Si on ne prend pas compte de la méthode de répartition des sièges, le nombre total de possibilités est donné en choisisant 12 sièges parmis les 108 disponibles. Par conséquent, la probabilité de A, c'est-à-dire d'avoir exactement 2 paires séparées, est donnée par

$$\mathbb{P}(A) = \frac{\mathcal{C}_{36}^{8} f(8, 12, 28)}{\mathcal{C}_{12}^{108}} = \frac{8\,199\,405\,016}{520\,752\,510\,551} \approx 1{,}574\,\%.$$

À la page suivant ce trouve une implémentation de $f(c, n, s)$ en Haskell. Le temps d'exécution est instantané pour les paramètres du problème. De plus, un simulateur écrit en C est fournit afin de comparer la réponse du calcul. Il est probablement possible d'optimiser le calcul en éliminant les branches impossibles dès qu'elles arrivent, mais le calcul fut assez rapide qu'il en n'était pas nécessaire.

```haskell
module Main where

fact :: Int -> Int
fact n = product [1..n]

comb :: Int -> Int -> Int
comb k n = (product [n-k+1..n]) `div` (fact k)

f :: Int -> Int -> Int -> Int
f 0 0 s = 1
f 0 n s = if n < 0 || n > s then 0 else (2^n) * (comb n s)
f c n s = 3*(f (c-1) (n-2) s) + (f (c-1) (n-1) s) + (f (c-1) (n-3) s)

main :: IO ()
main = putStrLn $ show $ f 8 12 28
```

```c
#include <stdio.h>
#include <stdint.h>
#include <sys/random.h>

#define A 30
#define B 12
#define C 18
#define D  2

#define P (A - D)
#define R (2 * C)
#define S (R - P)

#define SIZE 1024
#define K 1000000

int fill(uint8_t* siege) {
    uint32_t buf[SIZE];

    /* on recommence à nouveau */
    for(int k = 0; k < 3*R; k++)
        siege[k] = 0;

    /* on génère des bons nombres aléatoires */
    getrandom((void*) buf, sizeof(uint32_t)*SIZE, 0);

    /* on assigne des places */
    int i = B; int j = 0;
    do {
        /*
         * possibilité d'aller chercher des
         * nombres extras aléatoires avec
         * un buffer overflow
         */
        double s = (double) buf[j++];
        int r = (int) (3*R*s/0xFFFFFFFF);

        /* on alloue un siège si possible */
        if(siege[r] == 0)
            siege[r] = 1;
        else
            continue;

        i--;
    } while(i > 0);

    /* on compte les paires libres */
    int sum = 0;
    for(int k = 0; k < R; k++)
        if((siege[3*k+0] == 0 && siege[3*k+1] == 0)
        || (siege[3*k+1] == 0 && siege[3*k+2] == 0))
            sum++;

    return sum;
}


int main() {
    uint8_t siege[3*R];

    /* on exécute la simulation plusieurs fois */
    int cas = 0;
    for(int k = 0; k < K; k++)
        if (fill(siege) == P)
            cas++;

    /* on calcule la probabilité */
    printf("%f\n", ((double) cas)/((double) K));

    return 0;
}
```

2 Deux têtes valent mieux qu'une

Soit une équipe dont un membre à le numéro le plus élevé m. Cela implique qu'il y a au moins 1 membre pour chaque numéro positif inférieur à m. La prochaine personne à ce joindre au groupe peut être inviter par n'importe quelle personne ayant un numéro dans $\{0, 1, \ldots, m\}$. Par conséquent, le nouveau membre se vera attribuer un numéro dans $\{1, 2, \ldots, m+1\}$. La figure 6 montre les formations possibles d'une équipe à 5 personnes.

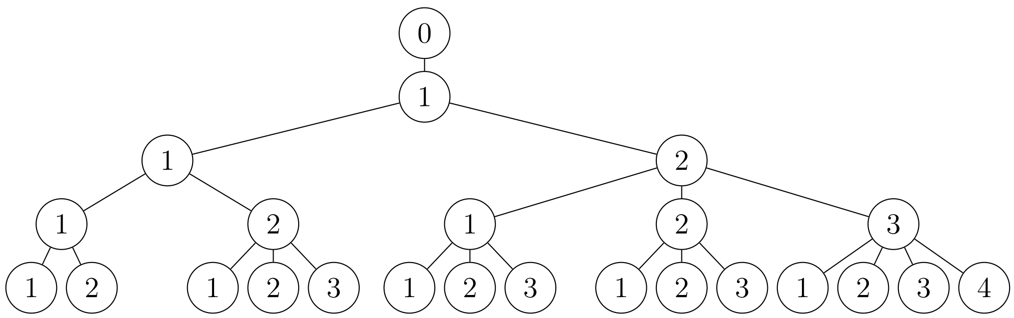

FIGURE 6 – formations possibles pour un groupe à 5 personnes

On remarque que le numéro de la dernière personne est sans importance dans le processus. Seulement le plus grand numéro dans l'équipe détermine le nombre de choix à une feuille de l'arbre. On peut représenter chaque feuille par un couple (n, m) où n est le nombre restant de membre à ajouter et m le numéro maximum actuel dans le groupe. La figure 7 représente un arbre en utilisant cette notation pour les feuilles. Les branches sont numérotés en fonction du membre inviter à une étape.

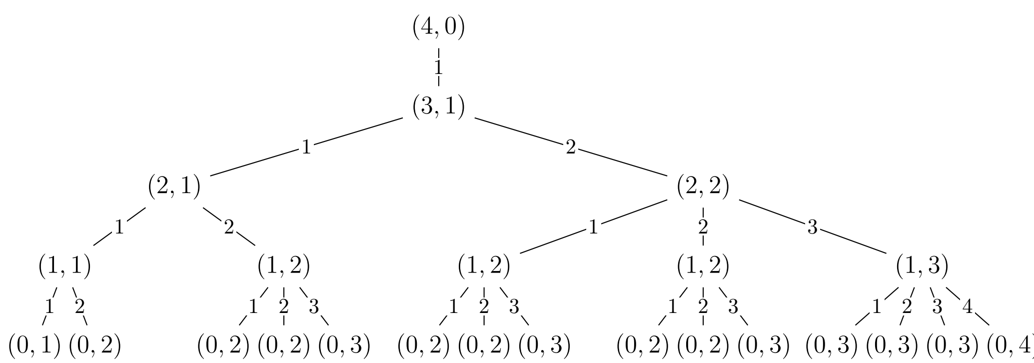

FIGURE 7 – représentation (n, m) de l'abre de la figure 6

2.1 Formations possibles

En examinant l'arbre de la figure 7, on remarque que chaque feuille contient m sous-feuilles identiques auxquelles le prochain membre obtient un numéro déjà existant dans le groupe. De plus, une feuille a aussi toujours une sous-feuille où le prochain membre obtient un numéro supérieur à ceux dans tout le groupe. La figure 8 résume n'importe quelle feuille (n, m).

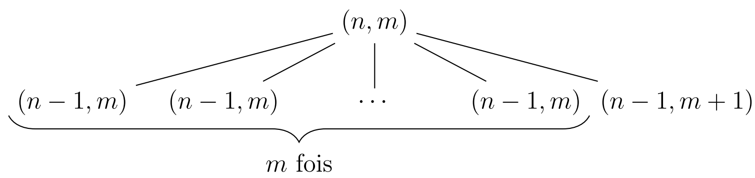

FIGURE 8 – possibilités du prochain numéro à une feuille (n, m)

Pour chaque feuille, on définie une fonction récursive f_1 nous donnant le nombre de formations possibles sous celle-ci tel que

$$f_1(n, m) = f_1(n-1, m+1) + m f_1(n-1, m).$$

La récursion termine lorsqu'il ne reste plus personne à ajouter, soit

$$f_1(0, m) = 1,$$

et elle commence par le fondateur ayant le numéro 0, c'est-à-dire

$$f_1(n, 0) = f_1(n-1, 1),$$

car il ne peut qu'inviter une personne en lui attribuant le numéro 1. Par conséquent, le système d'équations suivant calcule le nombre de formations possibles, soit

$$\begin{cases} f_1(n, m) = f_1(n-1, m+1) + m f_1(n-1, m) \\ f_1(n, 0) \ \ = f_1(n-1, 1) \\ f_1(0, m) = 1 \end{cases} \tag{1}$$

2.2 Formations possibles ayant exactement F numéro 1

On s'intéresse au nombre de formations ayant exactement F personnes invitées par le fondateur. La figure 9 montre les formations d'une équipe contenant 5 membres et exactement 3 personnes invitées par le fondateur. Les feuilles en rouge dénote une formation ne respectant pas cette condition.

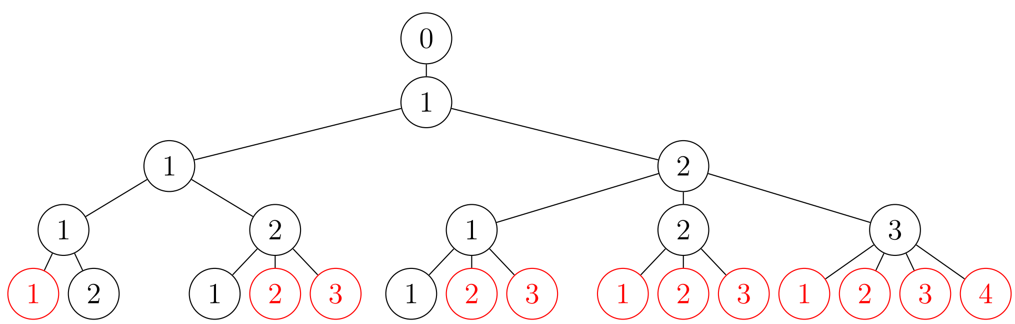

FIGURE 9 – formations de 5 membres contenant exactement 3 numéro 1

Pour calculer le nombre de formations possibles, on utilise une récursion similaire à celle présentée en (1). Par contre, il faut ajouter un nouveau paramètre à chaque feuille de l'arbre. On dénote chaque feuille (n, m, c) où c est le compte actuel de personnes invitées par le fondateur. La figure 10 représente un arbre utilisant cette notation.

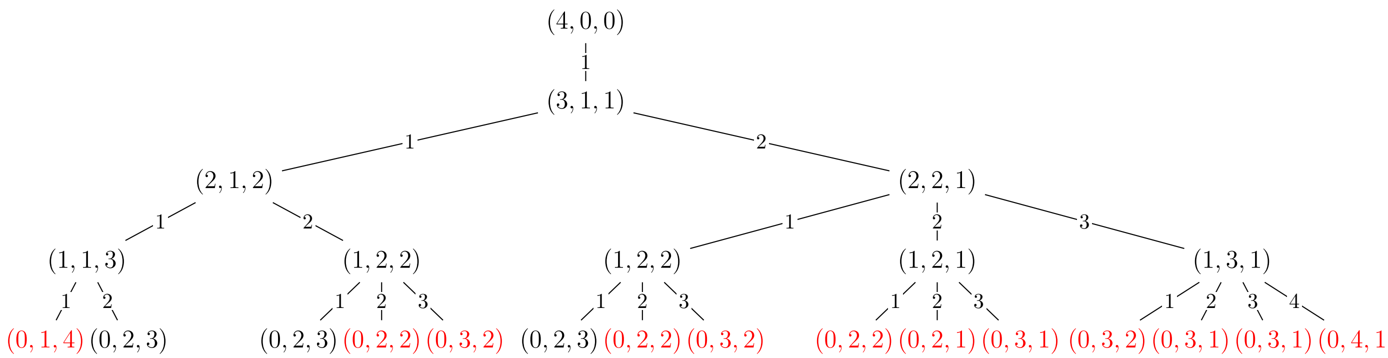

FIGURE 10 – représentation (n, m, c) de l'abre de la figure 9

En examinant l'arbre de la figure 10, on remarque qu'il est très similaire à celui de la figure 7. La seule différence est que c est incrémenter une fois par feuille, c'est-à-dire lorsque le fondateur invite une personne. La figure 11 représente n'importe quelle feuille (n, m, c).

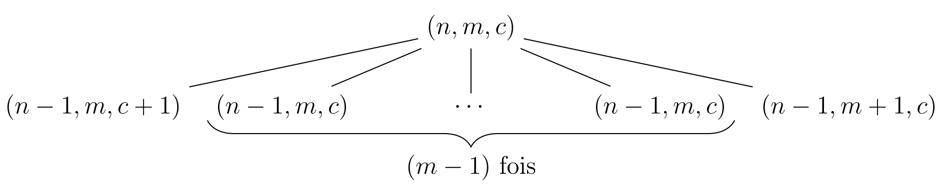

FIGURE 11 – possibilités du prochain numéro à une feuille (n, m, c)

Pour chaque feuille, on définie une fonction récursive f_2 nous donnant le nombre de formations possibles sous celle-ci tel que

$$f_2(n, m, c) = f_2(n-1, m, c+1) + f_2(n-1, m+1, c) + (m-1)f_2(n-1, m, c)$$

La récursion termine lorsqu'il ne reste plus personne à ajouter et elle est est valide si $c = F$, soit

$$f_2(0, m, F) = 1,$$

sinon

$$f_2(0, m, c) = 0,$$

et elle commence par le fondateur ayant le numéro 0, c'est-à-dire

$$f_2(n, 0, c) = f_2(n-1, 1, 1),$$

car il ne peut qu'inviter une personne en lui attribuant le numéro 1. Par conséquent, le système d'équations suivant calcule le nombre de formations possibles, soit

$$\begin{cases} f_2(n,m,c) &= f_2(n-1,m,c+1) + f_2(n-1,m+1,c) + (m-1)f_2(n-1,m,c) \\ f_2(n,0,c) &= f_2(n-1,1,1) \\ f_2(0,m,c) &= 0 \\ f_2(0,m,F) &= 1 \end{cases} \tag{2}$$

2.3 Formations possibles d'aucun numéro supérieur à G

Pour trouver les formations ne contenant pas de numéro supérieur à G, il suffit de les filtrer à la fin du calcul. Étant que l'on suit déjà le plus grand numéro dans la représentation (n,m), il suffit de redéfinir la fin de la récursion par

$$f_3(0,m) = \begin{cases} 1 & \text{si} \quad m \leq G \\ 0 & \text{si} \quad m > G \end{cases},$$

de sorte à obtenir la fonction f_3 définie par le système d'équations

$$\begin{cases} f_3(n,m) = f_3(n-1,m+1) + m f_3(n-1,m) \\ f_3(n,0) \ = f_3(n-1,1) \\ f_3(0,m) \ = \begin{cases} 1 \text{ si } m \leq G \\ 0 \text{ si } m > G \end{cases}, \end{cases} \tag{3}$$

La figure 12 montre le cas d'une équipe de 5 personnes ne pouvant pas contenir des numéros supérieurs à 2.

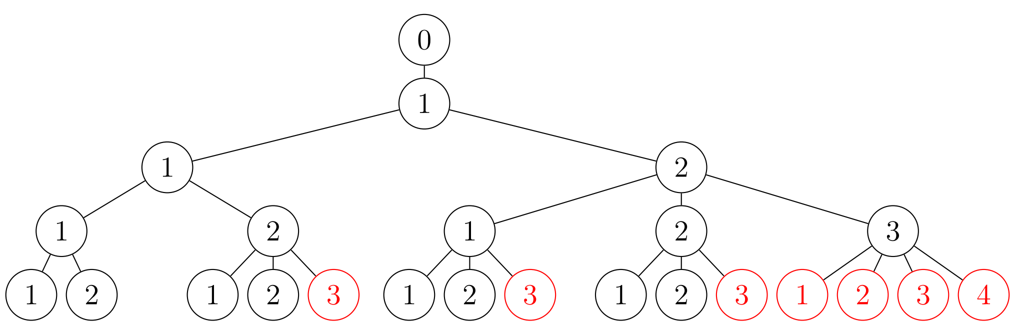

Figure 12 – formations ne contenant pas des numéros supérieurs à 2

2.4 Formations avec les deux contraintes

En utilisant la même logique qu'au paravant, il suffit d'ajouter la condition établie en (3) dans (4) afin d'obtenir la fonction f_5 définie par le système d'équations

$$\begin{cases} f_4(n,m,c) &= f_4(n-1,m,c+1) + f_4(n-1,m+1,c) + (m-1)f_4(n-1,m,c) \\ f_4(n,0,c) &= f_4(n-1,1,1) \\ f_4(0,m,c) &= 0 \\ f_4(0,m,F) &= \begin{cases} 1 \text{ si } m \leq G \\ 0 \text{ si } m > G \end{cases}, \end{cases} \tag{4}$$

2.5 Solution

Soit des groupes de $E = 17$ personnes et les événements

A : le fondateur a invité exactement $F = 3$ personnes
B : il n'y a aucun numéro supérieur à $G = 7$

Il y a un total de $f_1(16, 0)$ formations possibles. Parmis celles-ci, il y a $f_2(16, 0, 0)$ formations dont le fondateur a invité 3 personnes. Finalement, il y a $f_3(16, 0)$ formations dans lequel il n'y a aucun numéro supérieur à 7. [1]

La probabilité qu'une équipe formée contient 3 membres invités par le fondateur est

$$\mathbb{P}(A) = \frac{f_2(16, 0, 0)}{f_1(16, 0)} = \frac{2\,902\,665\,885}{10\,480\,142\,147}$$

et celle qu'il n'y ait aucune numéro supérieur à 7 est

$$\mathbb{P}(B) = \frac{f_3(16, 0)}{f_1(16, 0)} = \frac{7\,291\,973\,067}{10\,480\,142\,147}.$$

La probabilité qu'on ne retrouve aucun étudiant avec un numéro supérieure à 7 sachant que le fondateur a invité exactement 3 personnes est donnée par

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)},$$

où la probabilité d'avoir les deux contraintes est donné par

$$\mathbb{P}(A \cap B) = \frac{f_4(16, 0, 0)}{f_1(16, 0)} = \frac{2\,060\,946\,720}{10\,480\,142\,147},$$

de sorte à obtenir

$$\mathbb{P}(B|A) = \frac{2\,060\,946\,720}{2\,902\,665\,885} \approx 0{,}71\,\%.$$

La prochaine page contient le listage du code ayant effectué les calculs de f_1, f_2, f_3 et f_4 en Haskell. Le temps d'exécution fut pratiquement instantanné pour les paramètres de ce problème.

1. On utilise $n = 16$ au lieu de $n = E = 17$, car le fondateur est compté parmis les membres de l'équipe dès le départ

```haskell
module Main where

f1 :: Int -> Int -> Int
f1 0 _ = 1
f1 n 0 = (f1 (n-1) 1)
f1 n m = (f1 (n-1) (m+1)) + m * (f1 (n-1) m)

f2 :: Int -> Int -> Int -> Int
f2 0 _ 3 = 1
f2 0 _ _ = 0
f2 n 0 _ = (f2 (n-1) 1 1)
f2 n m c = (f2 (n-1) (m+1) c) + (f2 (n-1) m (c+1)) + (m-1) * (f2 (n-1) m c)

f3 :: Int -> Int -> Int
f3 0 m = if m <= 7 then 1 else 0
f3 n 0 = (f3 (n-1) 1)
f3 n m = (f3 (n-1) (m+1)) + m * (f3 (n-1) m)

f4 :: Int -> Int -> Int -> Int
f4 0 m 3 = if m <= 7 then 1 else 0
f4 0 _ _ = 0
f4 n 0 _ = (f4 (n-1) 1 1)
f4 n m c = (f4 (n-1) (m+1) c) + (f4 (n-1) m (c+1)) + (m-1) * (f4 (n-1) m c)

main :: IO ()
main = putStrLn $ show $ map show [f1 16 0, f2 16 0 0, f3 16 0, f4 16 0 0]
```

3 Au confluent de deux rivières

Soit les événements

A : la zone 1 n'est pas inondée
B : la zone 2 n'est pas inondée
C : la zone 3 n'est pas inondée
D : au moins une zone est inondée

On peut définir la probabilité d'avoir au moins une zone inondée avec son complément, c'est-à-dire qu'il n'y ait aucune zone d'inondée, soit

$$\mathbb{P}(D) = 1 - \mathbb{P}(D^c) = 1 - \mathbb{P}(A \cap B \cap C).$$

Hors, la probabilité que la zone 3 ne soit pas inondée est équivalente à la probabilité que son niveau d'eau h_T soit inférieure à H, soit

$$\mathbb{P}(C) = \mathbb{P}(h_T \leq H) = F_T(H) = \int_{-\infty}^{H} f_T(t)\,\mathrm{d}t, \tag{5}$$

où F_T est la fonction de répartition de $h_T(h)$ et $f_T(h)$ sa fonction de densité de probabilité. De plus, le niveau d'eau à la zone 3 dépend du niveau d'eau h_1 à la zone 1, h_2 à la zone 2 et d'une valeur h_3 selon

$$h_T = Mh_1 + Nh_2 + h_3.$$

Les valeurs h_1, h_2 et h_3 sont des variables aléatoires uniformes telles que $h_1 \in \mathbb{D}_1 = [I, J]$, $h_2 \in \mathbb{D}_2 = [K, L]$ et $h_3 \in \mathbb{D}_3 = [O, P]$. On peut définir

$$h_T = h_4 + h_5 + h_3,$$

où h_4 et h_5 sont aussi des variables aléatoires uniformes telles que $h_4 \in \mathbb{D}_4 = [M \cdot I, M \cdot J]$ et $h_5 \in \mathbb{D}_5 = [N \cdot K, N \cdot L]$.

On cherche la fonction de densité de probabilité $f_T(h)$. On suppose que $h_T = h$, $h_4 = x$ et $h_5 = y$. L'équation (3) est valide si est seulement si $h_3 = h_T - x - y$. La probabilité d'avoir $h_T = h$ est équivalente à la probabilité d'avoir ces trois conditions, c'est-à-dire

$$\mathbb{P}(h_T = h) = \mathbb{P}(\{h_4 = x\} \cap \{h_5 = y\} \cap \{h_3 = h - x - y\}).$$

Puisque les variables h_3, h_4 et h_5 prennent des valeurs indépendamments l'un de l'autre, il en résulte qu'on peut écrire

$$\begin{aligned} \mathbb{P}(h_T = h) &= \mathbb{P}(h_4 = x)\mathbb{P}(h_5 = y)\mathbb{P}(h_3 = h - x - y) \\ &= f_4(x) f_5(y) f_3(h - x - y), \end{aligned} \tag{6}$$

où $f_3(h)$, $f_4(h)$ et $f_5(h)$ sont respectivement les fonctions de densité de probabiltié de h_3, h_4 et h_5 définies par

$$f_3(h) = \begin{cases} 1/(P-O) & \text{si } h \in \mathbb{D}_3 \\ 0 & \text{sinon} \end{cases},$$

$$f_4(h) = \begin{cases} {}^1/_{M(J-I)} & \text{si } h \in \mathbb{D}_4 \\ 0 & \text{sinon} \end{cases}$$

et

$$f_5(h) = \begin{cases} {}^1/_{N(L-K)} & \text{si } h \in \mathbb{D}_5 \\ 0 & \text{sinon} \end{cases}.$$

Hors, l'équation (6) est valide pour tout $x, y \in \mathbb{R}$ de sorte que la probabilité d'obtenir $h_T = h$ est la somme

$$\mathbb{P}(h_T = h) = \sum_{\forall x} \sum_{\forall y} f_4(x) f_5(y) f_3(h - x - y),$$

ou plus mathématiquement correcte sous la forme d'intégrale,

$$\mathbb{P}(h_T = h) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_4(x) f_5(y) f_3(h - x - y)\, \mathrm{d}x\mathrm{d}y,$$

car x et y sont continues.

On sait que $f_4(x) = 0$ si $x \in\notin \mathbb{D}_4$ et $f_5(y) = 0$ si $y \in\notin \mathbb{D}_5$ de sorte qu'on peut écrire

$$\begin{aligned} \mathbb{P}(h_T = h) &= \int_{\mathbb{D}_5} \int_{\mathbb{D}_4} f_4(x) f_5(y) f_3(h - x - y)\, \mathrm{d}x\mathrm{d}y \\ &= \int_{\mathbb{D}_5} \int_{\mathbb{D}_4} \left(\frac{1}{M(J-I)}\right) \left(\frac{1}{N(L-K)}\right) f_3(h - x - y)\, \mathrm{d}x\mathrm{d}y \\ &= \left(\frac{1}{M(J-I)}\right) \left(\frac{1}{N(L-K)}\right) \int_{\mathbb{D}_5} \int_{\mathbb{D}_4} f_3(h - x - y)\, \mathrm{d}x\mathrm{d}y\,. \end{aligned}$$

Même avec les paramètres biens définis, il est difficile d'intégrer cette fonction en raison de la définition par partie de f_3. Une intégration numérique est idéale. La figure 13 montre le graphique de f_T calculée numériquement.

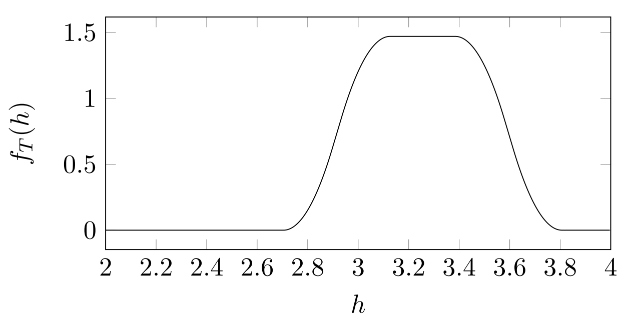

Figure 13 – graphique de $f_T(h)$ avec les paramètres du problème

L'aire sous sa courbe fut aussi approximée jusqu'à une précision de 1,000 000, confirmant validité de la démarche. De plus, on remarque $f_T(h) = 0$ lorsque h_3, h_4 et h_5 sont inférieurs à leur valeur minimum ou lorsqu'ils sont supérieurs à leur maximum.

Par définition, $f_T(h) = \mathbb{P}(h_t = h)$ de sorte qu'il est maintenant possible d'évaluer $F_T(h)$ avec (5), c'est-à-dire la probabilité que la zone 3 ne soit pas inondée. Hors, on cherche la probabilité que les trois zones ne sont pas inondées.

Par conséquent, la hauteur de la rivière à chaque zone ne doit pas dépasser H, c'est-à-dire $h_1 \in \mathbb{H}_1 = [I, H]$, $h_2 \in \mathbb{H}_1 = [K, H]$ et $h_T \leq H$. On obtient ainsi que $h_4 \in \mathbb{H}_1 = [M \cdot I, M \cdot H]$ et $h_4 \in \mathbb{H}_1 = [N \cdot K, N \cdot H]$ de sorte que la probabilité que $h_T = h$ et que les deux autres zones ne sont pas inondées peut se calculer avec

$$\mathbb{P}(\{h_T = h\} \cap \{h_1 \leq H\} \cap \{h_2 \leq H\}) = \left(\frac{1}{M(J-I)}\right)\left(\frac{1}{N(L-K)}\right)\int_{\mathbb{H}_5}\int_{\mathbb{H}_4} f_3(h-x-y)\,\mathrm{d}x\mathrm{d}y\,.$$

En d'autres termes, la dernière équation calcule la probabilité que $h_T = h$ tout en ignorant les cas où les deux autres rivières sont inondées. On s'intérèsse lorsque $h_T \leq H$. En intégrant, on obtient

$$\mathbb{P}(A \cap B \cap C) = \left(\frac{1}{M(J-I)}\right)\left(\frac{1}{N(L-K)}\right)\int_{-\infty}^{H}\int_{\mathbb{H}_5}\int_{\mathbb{H}_4} f_3(h-x-y)\,\mathrm{d}x\mathrm{d}y\mathrm{d}h\,.$$

On peut facilement calculer les valeurs possibles de h_T en utilisant les cas minimums et maximus de h_4, h_5 et h_3 de sorte que

$$\mathbb{P}(A \cap B \cap C) = \left(\frac{1}{M(J-I)}\right)\left(\frac{1}{N(L-K)}\right)\int_{h_{\min}}^{H}\int_{\mathbb{H}_5}\int_{\mathbb{H}_4} f_3(h-x-y)\,\mathrm{d}x\mathrm{d}y\mathrm{d}h,$$

où $h_{\min} = M \cdot I + N \cdot K + O$.

Par intégration numérique, on obtient que $\mathbb{P}(A \cap B \cap C) \approx 0{,}432\,835$ de sorte que la probabilité qu'au moins une zone soit inondée est $1 - 0.432835 \approx 57\,\%$. La prochaine page contient le code effectuant le calcul ainsi qu'une simulation numérique du problème. Les deux résultats semblent être en accord ensemble.

```c
/**
 * gcc main.c -D_CALCUL_AIRE && ./a.out
 * gcc main.c && ./a.out
 */

#include <stdlib.h>
#include <stdint.h>
#include <stdio.h>
#include <sys/random.h>

#define H 3.35
#define I 2.55
#define J 3.59
#define K 2.80
#define L 3.49
#define M 0.22
#define N 0.28
#define O 1.36
#define P 2.04

#define M1  (M * I)
#define M2  (M * J)
#define N1  (N * K)
#define N2  (N * L)
#define MIN (M * I + N * K + O)
#define MAX (M * J + N * L + P)

double f3(double h) {
    if(O <= h && h <= P)
        return 1/(P-O);
    else
        return 0;
}

double ft(double h) {
    int n = 900;
    int m = 900;

#ifdef _CALCUL_AIRE
    /* calcul si les domaines sont D4 et D5 */
    double dx = (M2-M1)/m;
    double dy = (N2-N1)/n;
#else
    /* calcul si les domaines sont H4 et H5 */
    double dx = (M*H-M1)/m;
    double dy = (N*H-N1)/n;
#endif
    double sum = 0;
    for(int j = 0; j < n; j++) {
        for(int i = 0; i < m; i++) {
            double x = M1 + i * dx;
            double y = N1 + j * dy;

            sum += f3(h-x-y)*dx*dy;
        }
    }

    return (1/(M2-M1))*(1/(N2-N1))*sum;
}

double Ft(double h) {
    int n = 2000;

    double dh = (h-MIN)/n;
    double sum = 0;
    for(int i = 0; i < n; i++) {
        double t = MIN + i * dh;

        sum += ft(t)*dh;
    }

    return sum;
}

double simulate(uint64_t n) {
    uint32_t* buf1 = malloc(sizeof(uint32_t)*n);
    uint32_t* buf2 = malloc(sizeof(uint32_t)*n);
    uint32_t* buf3 = malloc(sizeof(uint32_t)*n);

    getrandom((void*) buf1, sizeof(uint32_t)*n, 0);
    getrandom((void*) buf2, sizeof(uint32_t)*n, 0);
    getrandom((void*) buf3, sizeof(uint32_t)*n, 0);

    uint64_t count = 0;
    for(int i = 0; i < n; i++) {
        double t1 = buf1[i];
        double t2 = buf2[i];
        double t3 = buf3[i];

        double h1 = t1*((J-I)/0xFFFFFFFF)+I;
        double h2 = t2*((L-K)/0xFFFFFFFF)+K;
        double h3 = t3*((P-O)/0xFFFFFFFF)+O;

        double ht = M*h1+N*h2+h3;
        if(ht <= H && h1 <= H && h2 <= H)
            count++;
    }

    free(buf1);
    free(buf2);
    free(buf3);

    return ((double) count)/((double) n);
}



int main() {
#ifdef _CALCUL_AIRE
    printf("%f\n", Ft(MAX));
#else
    printf("%f\n", simulate(8000000));
    printf("%f\n", Ft(H));
#endif

    return 0;
}
```

4 Chasse à l'agent double

Pour chaque personne dans l'équipe, on définie des événements pour chaque classe possible, soit

$$\begin{aligned} A &: \text{la personne est une recrue} \\ B &: \text{la personne est un agent régulier} \\ C &: \text{la personne est un agent senior} \end{aligned}$$

avec $\mathbb{P}(A) = {}^{20}\!/_{67}$, $\mathbb{P}(B) = {}^{22}\!/_{67}$ et $\mathbb{P}(C) = {}^{25}\!/_{67}$, les probabilités qu'une personne fasse partie d'un groupe.

Dans les 67 agents, il y a un agent double. Soit l'événement

$$D : \text{la personne est l'agent double}$$

de sorte que $\mathbb{P}(D) = {}^{1}\!/_{67}$. Le directeur estime que $\mathbb{P}(A|D) = 0.51$, $\mathbb{P}(B|D) = 0.32$ et $\mathbb{P}(C|D) = 0.17$.

Pour l'interrogation, on définie les événements

$$\begin{aligned} I &: \text{la personne est interrogée} \\ T &: \text{l'interrogation révèle un agent double} \\ L &: \text{l'interrogation révèle un agent loyal} \end{aligned}$$

avec $\mathbb{P}(T^c) \neq \mathbb{P}(L)$, car on peut voir ces événements comme deux tests à part à l'intérieur de l'interrogation. Les probabilités d'identifier correctement les agents sont données par $\mathbb{P}(D|T \cap I \cap A) = 0.96$, $\mathbb{P}(D|T \cap I \cap B) = 0.48$ et $\mathbb{P}(D|T \cap I \cap C) = 0.24$, tandis que celles d'identifier correctement un agent lotal sont $\mathbb{P}(D^c|L \cap I \cap A) = 0.84$, $\mathbb{P}(D^c|L \cap I \cap B) = 0.92$ et $\mathbb{P}(D^c|L \cap I \cap C) = 0.96$.

On sait que l'interrogation a révélée un seul agent double dans parmi les 20 recrues. La probabilité que l'interrogation en révèle qu'un seul agent parmi ceux-ci est donnée par

$$\mathbb{P}\left[(L_1 \cap I_1 \cap A_1) \cap (L_2 \cap I_2 \cap A_2) \cap \cdots \cap (L_{17} \cap I_{17} \cap A_{17}) \cap (T_{18} \cap I_{18} \cap A_{18})\right],$$

où les indices dénotes un agent interrogé. Puisqu'il sont tous interrogés indépendaments, on a

$$\mathbb{P}(L_1 \cap I_1 \cap A_1)\mathbb{P}(L_2 \cap I_2 \cap A_2) \cdots \mathbb{P}(L_{17} \cap I_{17} \cap A_{17})\mathbb{P}(T_{18} \cap I_{18} \cap A_{18}),$$

ou encore

$$\mathbb{P}(Z_A) = \mathbb{P}(L_i \cap I_i \cap A_i)^{17}\mathbb{P}(T_i \cap I_i \cap A_i),$$

où Z_A est l'événement tel que l'interrogatoire a révélé un agent double parmi les recrues. Par le même raisonement, on a aussi

$$\mathbb{P}(Z_B) = \mathbb{P}(L_i \cap I_i \cap B_i)^{19}\mathbb{P}(T_i \cap I_i \cap B_i)$$

et

$$\mathbb{P}(Z_B) = \mathbb{P}(L_i \cap I_i \cap C_i)^{21}\mathbb{P}(T_i \cap I_i \cap C_i),$$

pour les agents réguliers et seniors.

On cherche les probabilités que soit la recrue, soir l'agent régulier ou l'agent senior soit l'agent double. Par conséquent, on cherche $\mathbb{P}(D|Z_A)$, $\mathbb{P}(D|Z_B)$ et $\mathbb{P}(D|Z_C)$. Dans le cas où aucun de ces agents n'est le vrai agent double, il faut calculer le complément des trois cas, soit $1 - \mathbb{P}(D|Z_A) - \mathbb{P}(D|Z_B) - \mathbb{P}(D|Z_C)$.

Annexe

Table 1 – table des valeurs

lettre	nombre	lettre	nombre
A	30,00	O	1,36
B	12,00	P	2,04
C	18,00	Q	67,00
D	2,00	R	20,00
E	17,00	S	22,00
F	3,00	T	0,51
G	7,00	U	0,32
H	3,35	V	18,00
I	2,55	W	0,96
J	3,59	X	14,00
K	2,80	Y	71,00
L	3,49	Ψ	24 000,00
M	0,22	Ω	67,00
N	0,28		