TP 3 - Option informatique - MP* - Codes correcteurs - Correction

Question 1 –

```
let canal (p: float) (x: string): string =
  let f i =
    if Random.float 1. < p then
      if x.[i] = '0' then '1' else '0'
    else
      if x.[i] = '0' then '0' else '1'
  in
  let n = String.length x in
  String.init n f;;
```

Question 2 – Étant donné que chaque bit est flippé de manière indépendante, la probabilité qu'aucun bit n'ait été flippé est $(1-p)^n$.

Question 3.a –

```
let phi (n: int) (m: string): string =
  if m <> "0" && m <> "1" then failwith "phi: entrée invalide";
  let f i = m.[0] in
  String.init n f;;
```

Question 3.b –

```
(* Suppose n >= 1 *)
let est_valide (n: int) (x: string): bool =
  if String.length x <> n || (x.[0] <> '0' && x.[0] <> '1') then
    false
  else begin
      let i = ref 1 in
      while !i < n && (x.[!i] = x.[0]) do
        incr i
      done;
      !i = n
    end;;
```

```
let phi_inv (n: int) (x: string): string =
  if not (est_valide n x) then failwith "phi_inv: entrée invalide";
  let f i = x.[0] in
  String.init 1 f;;
```

Question 4 –

```
let vote_maj (y: string): string =
  let n = String.length y in
  let nb_uns = ref 0 in
  for i = 0 to n-1 do
    if y.[i] = '1' then incr nb_uns
  done;
  let c = if !nb_uns <= n/2 then '0' else '1' in
  let f i = c in
  String.init n f;;
```

Question 5.a – Pour que la communication soit un succès, il faut et il suffit que le nombre de bits flippés par le canal soit inférieur ou égal à r. On numérote les bits du message de 1 à n. Soit X la variable aléatoire à valeurs dans $\mathcal{P}(\llbracket 1;n \rrbracket)$ qui donne l'ensemble des numéros des bits flippés par le canal. Alors :

$$\begin{aligned} S(r,p) &= \mathbb{P}(\mathrm{card}(X) \leqslant r) \\ &= \sum_{k=0}^{r} \mathbb{P}(\mathrm{card}(X) = k) \\ &= \sum_{k=0}^{r} \sum_{\substack{E \subset \llbracket 1;n \rrbracket \\ \mathrm{card}(E)=k}} \mathbb{P}(X = E) \\ &= \sum_{k=0}^{r} \binom{n}{k} p^k (1-p)^{n-k} \\ &= \sum_{k=0}^{r} \binom{2r+1}{k} p^k (1-p)^{2r+1-k} \end{aligned}$$

Question 5.b – On se place dans le cas d'un code de répétition à $2r+3$ bits. Soit Y la variable aléatoire à valeurs dans $\llbracket 0; 2r+1 \rrbracket$ qui donne le nombre de bits flippés par le canal (Y ne tient pas compte des 2 derniers bits). On définit les évènements :

A : "La valeur de Y est inférieure ou égale à r."
B : "Y a pour valeur r et les 2 derniers bits sont flippés par le canal."
C : "Y a pour valeur $r+1$ et les 2 derniers bits ne sont pas flippés par le canal."
D : "La communication est un succès."

On a alors :

$$D = (A \setminus B) \cup C.$$

Donc :

$$S(r+1,p) = \mathbb{P}(D) = \mathbb{P}(A) - \mathbb{P}(B) + \mathbb{P}(C)$$

On remarque que $\mathbb{P}(A)$ est égal à la probabilité que la communication avec un code de répétition à $2r+1$ bits soit un succès. Ainsi :

$$\begin{aligned} S(r+1,p) &= S(r,p) - \mathbb{P}(B) + \mathbb{P}(C) \\ &= S(r,p) - \binom{2r+1}{r} p^r (1-p)^{2r+1-r} p^2 + \binom{2r+1}{r+1} p^{r+1} (1-p)^{2r+1-r-1} (1-p)^2 \\ &= S(r,p) - \binom{2r+1}{r} p^{r+2} (1-p)^{r+1} + \binom{2r+1}{r} p^{r+1} (1-p)^{r+2} \\ &= S(r,p) + \binom{2r+1}{r} p^{r+1} (1-p)^{r+1} (1-2p) \end{aligned}$$

En conclusion :

- ⋆ Si $0 < p < 1/2$ alors $S(r+1,p) > S(r,p)$.
- ⋆ Si $p \in \{0, 1/2, 1\}$ alors $S(r+1,p) = S(r,p)$.
- ⋆ Si $1/2 < p < 1$ alors $S(r+1,p) < S(r,p)$.

Ainsi :

- ⋆ Si $0 < p < 1/2$ alors augmenter le nombre de répétitions rend la communication plus fiable.
- ⋆ Si $p \in \{0, 1/2, 1\}$ alors augmenter le nombre de répétitions ne change rien à la fiabilité de la communication.
- ⋆ Si $1/2 < p < 1$ alors augmenter le nombre de répétitions rend la communication moins fiable.

Question 6 –

```
(* Effectue une simulation du protocole *)
let une_simul (p: float) (n: int): bool =
  let m = if Random.float 1. < 0.5 then "0" else "1" in
  let x = phi n m in
  let y = canal p x in
  let x_hat = vote_maj y in
  let m_hat = phi_inv n x_hat in
  m = m_hat;;
```

```
let simul (p: float) (n: int) (m: int) =
  let nb_s = ref 0 in
  for i = 1 to m do
    if une_simul p n then
      incr nb_s;
  done;
  (float_of_int !nb_s) /. (float_of_int m);;
```

Question 7 – Il s'agit d'un code de répétition à 5 bits :

$$\mathcal{M} = \Big\{(0,0,0,0,0),(1,1,1,1,1)\Big\}.$$

En effet, soit $x = (x_0, x_1, x_2, x_3, x_4) \in \mathbb{F}_2^5$ un élément de $\mathcal{M}$ et $i \in [\![0;3]\!]$. Alors, pour que c_i soit satisfait par x, il faut nécessairement que $x_{i+1} = x_i$. On en déduit que $x_0 = x_1 = x_2 = x_3 = x_4$, d'où la valeur de $\mathcal{M}$.

Question 8 – Soient x_i, y_i, σ_j, σ'_j et σ''_j les composantes de x, y, $\sigma(x)$, $\sigma(y)$ et $\sigma(x \oplus y)$:

$$\begin{aligned} x &= (x_0, x_1, \dots, x_{n-1}), & \sigma(x) &= (\sigma_0, \sigma_1, \dots, \sigma_{\ell-1}),\\ y &= (y_0, y_1, \dots, y_{n-1}), & \sigma(y) &= (\sigma'_0, \sigma'_1, \dots, \sigma'_{\ell-1}).\\ & & \sigma(x \oplus y) &= (\sigma''_0, \sigma''_1, \dots, \sigma''_{\ell-1}). \end{aligned}$$

Soit c_j un noeud de parité de G. On pose :

$$\begin{aligned} I_1 &= \Big\{i \in [\![0; n-1]\!] : v_i \in \Gamma(c_j) \text{ et } x_i = 1\Big\},\\ I_2 &= \Big\{i \in [\![0; n-1]\!] : v_i \in \Gamma(c_j) \text{ et } y_i = 1\Big\},\\ I_3 &= \Big\{i \in [\![0; n-1]\!] : v_i \in \Gamma(c_j) \text{ et } x_i \oplus y_i = 1\Big\}. \end{aligned}$$

Pour que $a \oplus b = 1$, il faut que $(a, b) = (1, 0)$ ou $(a, b) = (0, 1)$. Ainsi :

$$I_3 = \Big(I_1 \setminus (I_1 \cap I_2)\Big) \cup \Big(I_2 \setminus (I_1 \cap I_2)\Big)$$

Il s'agit d'une union disjointe, donc :

$$\mathrm{card}(I_3) = \mathrm{card}(I_1) + \mathrm{card}(I_2) - 2\mathrm{card}(I_1 \cap I_2).$$

On réduit modulo 2 :

$$\sigma''_j \equiv \mathrm{card}(I_3) \equiv \mathrm{card}(I_1) + \mathrm{card}(I_2) \equiv \sigma_j + \sigma'_j \mod 2$$

Comme σ_j, σ'_j et σ''_j appartiennent à $\{0, 1\}$, on a bien $\sigma''_j = \sigma_j + \sigma'_j$.

Question 9.a –

```
(* Cette fonction prend en entrée un tableau d'entiers ainsi qu'une liste
   d'entiers.
   Pour chaque entier i de la liste, elle remplace tab.(i) par (1-tab.(i)) *)
let rec swap (tab: int array) (li: int list) = match li with
  | [] -> ()
  | i :: q -> tab.(i) <- 1-tab.(i);
              swap tab q;;
```

```
let syndrome (g: tanner) (x: string): int array =
  let n = Array.length g.adj in
  let synd = Array.make g.m 0 in
  for i = 0 to n-1 do
    if x.[i] = '1' then swap synd g.adj.(i)
  done;
  synd;;
```

Question 9.b –

```
(* Cette fonction indique si un mot binaire est un mot de code *)
let est_mot_code (g: tanner) (x: string): bool =
  let synd = syndrome g x in
  let m = Array.length synd in
  let i = ref 0 in
  while !i < m && synd.(!i) = 0 do
    incr i;
  done;
  !i = m;;
```

```
(* L'idée est de générer tous les tableaux contenant des '0' et '1' et pour
   chacun de ces tableaux de tester si la chaîne de caractères correspondante
   est un mot de code. *)
let liste_mots_code (g: tanner): string list =
  let n = Array.length g.adj in
  let tab = Array.make n '0' in
  let f i = tab.(i) in
  let rec aux res i = match i with
    | i when i < n ->
       tab.(i) <- '1';
       let res2 = aux res (i+1) in
       tab.(i) <- '0';
       aux res2 (i+1)
    | i -> (* i = n *)
       let x = String.init n f in
       if est_mot_code g x then x :: res else res
  in
  aux [] 0;;
```

Question 10 –

```
(* Renvoie le mot binaire x où le bit numéro i a été flippé *)
let flip (x: string) (i: int): string =
  let n = String.length x in
  let f j =
    if i = j then
      if x.[i] = '0' then '1' else '0'
    else
      x.[j]
  in
  String.init n f;;
```

```
let poids (tab: int array) =
  let n = Array.length tab in
  let s = ref 0 in
  for i = 0 to n-1 do
    s := !s + tab.(i)
  done;
  !s;;
```

```
(* Renvoie n si aucun flip ne fait décroite le poids du syndrome *)
let trouver_flip (g: tanner) (x: string) =
  let n = Array.length g.adj in
  let i = ref 0 in
  while !i < n && poids(syndrome g x) <= poids(syndrome g (flip x !i)) do
    incr i
  done;
  !i;;
```

```
let rec bit_flip (g: tanner) (y: string) =
  (* Printf.printf "%s \n" y; *)
  let n = Array.length g.adj in
  let i = trouver_flip g y in
  if i = n then
    y
  else
    bit_flip g (flip y i);;
```

Question 11 – Pour commencer, remarquons que :

$$|\Gamma_0(E)| = |\Gamma_u(E)| + |\Gamma_m(E)|.$$

Comme tous les sommets de $E \subset V$ ont pour degré d_V, on a :

$$\sum_{e \in E} |\Gamma(e)| = d_V |E|.$$

Dans la somme $\sum_{e\in E} |\Gamma(e)|$, chaque élément de $\Gamma_u(E)$ est compté une fois et chaque élément de $\Gamma_m(E)$ est compté au moins deux fois. Donc :

$$\sum_{e \in E} |\Gamma(e)| \geqslant |\Gamma_u(E)| + 2|\Gamma_m(E)|$$

On obtient :

$$d_V |E| \geqslant |\Gamma_u(E)| + 2|\Gamma_m(E)| = |\Gamma_0(E)| + |\Gamma_m(E)|$$

En utilisant la définition de graphe expanseur :

$$|\Gamma_{\mathrm{m}}(E)| \leqslant d_V|E| - |\Gamma_0(E)| \leqslant \delta d_V|E|$$

De plus :

$$|\Gamma_{\mathrm{u}}(E)| = |\Gamma_0(E)| - |\Gamma_{\mathrm{m}}(E)| \geqslant (1-\delta)d_V|E| - \delta d_V|E| = (1-2\delta)d_V|E|.$$

Question 12.a –
⋆ À chaque tour de boucle, le poids du syndrome diminue. Ainsi, $f \leqslant \|\sigma(y)\|$. En utilisant le fait que x est un mot de code et avec la question 8 :

$$f \leqslant \|\sigma(y)\| = \|\sigma(x) \oplus \sigma(y)\| = \|\sigma(x \oplus y)\|.$$

⋆ Comme tous les sommets de V ont pour degré d_V, on a :

$$\|\sigma(x \oplus y)\| \leqslant d_V\|x \oplus y\|$$

Ainsi :

$$\begin{aligned} |U| &\leqslant |supp(x \oplus y)| + f \\ &= \|x \oplus y\| + f \\ &\leqslant \|x \oplus y\| + d_V\|x \oplus y\| \\ &= (1 + d_V)\|x \oplus y\|. \end{aligned}$$

Question 12.b – Soit $F = supp(z)$. Puisque $|F| = \|z\| \leqslant \gamma|V|$, on peut utiliser la question 11 :

$$|\Gamma_{\mathrm{u}}(F)| \geqslant (1-2\delta)d_V|F|$$

On a :

$$\frac{1}{|F|}\sum_{e\in F}\left|\Gamma_{\mathrm{u}}(F) \cap \Gamma(e)\right| = \frac{\left|\Gamma_{\mathrm{u}}(F)\right|}{|F|} \geqslant d_V(1-2\delta).$$

Ainsi, il existe au moins un élément $e_0 \in F$ tel que $\left|\Gamma_{\mathrm{u}}(F) \cap \Gamma(e_0)\right| \geqslant d_V(1-2\delta)$. Soit $e \in \mathbb{F}_2^n$ le mot binaire tel que $e_0 = supp(\{e\})$. Alors $\|e\| = 1$ et :

$$\begin{aligned} \|\sigma(z \oplus e)\| &= \|\sigma(z) \oplus \sigma(e)\| \\ &= |supp(\sigma(z) \oplus \sigma(e))| \\ &= |supp(\sigma(z))\Delta supp(\sigma(e))| \end{aligned}$$

où Δ est la différence symétrique. Ainsi :

$$\begin{aligned} \|\sigma(z \oplus e)\| &= |supp(\sigma(z))| + |supp(\sigma(e))| - 2|supp(\sigma(z)) \cap supp(\sigma(e))| \\ &= \|\sigma(z)\| + \left|\Gamma(e_0)\right| - 2\left|supp(\sigma(z)) \cap \Gamma(e_0)\right| \end{aligned}$$

Comme $\Gamma_{\mathrm{u}}(F) \subseteq supp(\sigma(z))$:

$$\begin{aligned} \|\sigma(z \oplus e)\| &\leqslant \|\sigma(z)\| + \left|\Gamma(e_0)\right| - 2\left|\Gamma_{\mathrm{u}}(F) \cap \Gamma(e_0)\right| \\ &\leqslant \|\sigma(z)\| + d_V - 2d_V(1-2\delta) \\ &= \|\sigma(z)\| - d_V(1-4\delta). \end{aligned}$$

Question 12.c – Soit $\hat{x}$ le mot binaire renvoyé par l'algorithme bit-flip et $z = x \oplus \hat{x}$. Notre but est de montrer que $z = (0, 0, \ldots, 0)$. Par l'absurde, supposons que $\|z\| > 0$ et montrons qu'on peut alors appliquer le résultat de la question 12.b à z. Comme $supp(z) \subset U$, alors d"après la question 12.a :

$$\|z\| \leqslant |U| \leqslant \|x \oplus y\|(1 + d_V)$$

Avec l'hypothèse de l'énoncé, on obtient $\|z\| \leqslant \gamma|V|$. On peut donc appliquer le résultat de la question 12.b, il existe $e \in \mathbb{F}_2^n$ tel que $\|e\| = 1$ et :

$$\|\sigma(z \oplus e)\| \leqslant \|\sigma(z)\| - d_V(1 - 4\delta) < \|\sigma(z)\|.$$

Ainsi :

$$\|\sigma(\hat{x} \oplus e)\| = \|\sigma(x \oplus \hat{x} \oplus e)\| = \|\sigma(z \oplus e)\| < \|\sigma(z)\| = \|\sigma(x \oplus \hat{x})\| = \|\sigma(\hat{x})\|.$$

La condition de la boucle while est donc vérifier pour $\hat{x}$ ce qui contredit le fait que l'algorithme s'arrête pour ce mot binaire. On en déduit que $\|z\| = 0$, c'est à dire $\hat{x} = x$.