Sujet X-ENS 2014 : correction

Question 1

```
let minimum a = let N(g,x,d)=a in x;;
```

Question 2

```
let rec est_un_arbre_croissant a = match a with
   | E -> true
   | N(E,x,d) -> (x<minimum d) && est_un_arbre_croissant d
   | N(g,x,E) -> (x<minimum g) && est_un_arbre_croissant g
   | N(g,x,d) -> (x< min (minimum g) (minimum d)) && (est_un_arbre_croissant g) && (est_un_arbre_croissant d) ;;
```

Question 3

Notons C_n le nombre d'arbres croissants étiquetés par $[\![1,n]\!]$. Notons aussi $\mathcal{C}(F)$ l'ensemble des arbres croissants étiquetés par F, un ensemble totalement ordonné.

Comme $[\![1,0]\!]=\emptyset$, le seul arbre étiqueté par cet ensemble est l'arbre vide `E` qui est bien croissant. On a donc $C_0=1$ et donc $C_0=0!$.

Il y a un seul arbre étiqueté par 1 (et cet arbre est bien croissant), c'est l'arbre `N(E,1,E)`. On a donc $C_1=1=1!$.

Soit $n\geq 1$. Supposons que pour tout $1\leq k\leq n$, $C_k=k!$. La racine d'un arbre croissant étant son minimum, la racine d'un arbre croissant étiqueté par $[\![1,n+1]\!]$ est nécessairement 1. Ses sous-arbres gauche et droit sont croissants (et éventuellement vides). Le sous-arbre gauche sera étiqueté par un sous-ensemble $F\subset[\![2,n+1]\!]$ (éventuellement vide) à k éléments, $0\leq k\leq n$; le sous-arbre droit sera alors croissant et étiqueté par $[\![2,n+1]\!]\backslash F$ qui possède $n+1-k$ éléments. Réciproquement, en se donnant $F\subset[\![2,n+1]\!]$, un arbre `g` croissant étiqueté par F et un arbre `d` croissant étiqueté par $[\![2,n+1]\!]\backslash F$, l'arbre `N(g,1,d)` est un arbre croissant étiqueté par $[\![1,n+1]\!]$.

Ainsi $\mathcal{C}([\![1,n+1]\!])$ est en bijection avec $\bigcup\limits_{F\subset[\![2,n+1]\!]}\mathcal{C}(F)\times\mathcal{C}([\![2,n+1]\!]\backslash F)$.

F étant fixé et ayant k éléments, en choisissant une bijection entre F et $[\![1,k]\!]$, on obtient une bijection entre $\mathcal{C}(F)$ est $\mathcal{C}([\![1,k]\!])$. Ainsi $\mathrm{Card}(\mathcal{C}(F))=C_k$. De même, $\mathrm{Card}(\mathcal{C}([\![2,n+1]\!]\backslash F)=C_{n-k}$.

On a donc

$$C_{n+1}=\mathrm{Card}(\bigcup_{F\subset[\![2,n+1]\!]}\mathcal{C}(F)\times\mathcal{C}([\![2,n+1]\!]\backslash F))=\sum_{F\subset[\![2,n+1]\!]}\mathrm{Card}(\mathcal{C}(F))\,\mathrm{Card}([\![2,n+1]\!]\backslash F)=\sum_{k=0}^{n}\sum_{\mathrm{Card}(F)=k}C_kC_{n-k}.$$

Comme il y a $\binom{n}{k}$ ensembles F à k éléments inclus dans $[\![2,n+1]\!]$, on a $C_{n+1}=\sum\limits_{k=0}^{n}\binom{k}{n}C_kC_{n-k}$. D'après l'hypothèse de récurrence, $C_{n+1}=\sum\limits_{k=0}^{n}\binom{n}{k}k!(n-k)!=\sum\limits_{k=0}^{n}n!=(n+1)n!=(n+1)!$.

Ceci démontre, par récurrence forte, que pour tout $n\in\mathbb{N}$, $C_n=n!$.

Question 4

Soient $g_1=N(E,2,E)$, $d_1=N(E,4,E)$, $g_2=N(E,5,E)$, $d_2=N(E,6,E)$.

Comme $1\leq 2$, le résultat de la fusion sera par définition `f1=N(fusion(d1,N(g2,3,d2)),1,g1)`.

Comme $4>3$, la fusion de d_1 et de $N(g_2,3,d_2)$ sera `f2=N(fusion(d2,N(E,4,E)),3,g2)`.

En outre, comme $6>4$, la fusion de d_2 et de $N(E,4,E)=d_1$ est `f3=N(fusion(E,N(E,6,E)),4,E)`.

Enfin, la fusion de `E` et `N(E,6,E)=d2` est `d2`.

Ainsi, `f3=N(d2,4,E)`, `f2=N(N(d2,4,E),3,g2)` et `f1=N(N(N(d2,4,E),3,g2),1,g1)`.

L'arbre obtenu est donc

`N(N(N(N(E,6,E),4,E),3,N(E,5,E)),1,N(E,2,E))`.

Question 5

De façon générale, si $t=N(g,y,d)$, $occ(x,t)=occ(x,g)+occ(x,d)+\mathbb{1}_x(y)$ où $\mathbb{1}_x(y)$ vaut 1 si $x=y$, 0 sinon.

Supposons que $t_2=E$ (où E est l'arbre vide). Alors $occ(x,t_2)=0$. On a donc $occ(t,x)=occ(x,t_1)=occ(x,t_1)+occ(x,t_2)$. La propriété à démontrer est donc vraie dans ce cas. Par symétrie des rôles de t_1 et t_2, ce résultat vaut si t_1 est vide. Ainsi, la propriété à démontrer est vraie si t_1 est vide ou t_2 est vide.

Supposons que t_1 et t_2 sont non vides. Notons $t_1=N(g_1,x_1,d_1)$ et $t_2=N(g_2,x_2,d_2)$. Supposons que la propriété à démontrer est vraie pour tout fusion f entre un arbre quelconque et l'un des arbre g_1, d_1, g_2 ou d_2.

Supposons que $x_1\leq x_2$. Notons f la fusion de d_1 Alors

$$occ(t,x)=occ(f,x)+occ(g_1,x)+\mathbb{1}_x(x_1).$$

Comme f est la fusion de d_1 et $N(g_2,x_2,d_2)$,

$$occ(x,f)=occ(x,d_1)+occ(x,N(g_2,x_2,d_2))=occ(x,d_1)+occ(x,t_2).$$

On a donc

$$occ(x,t)=occ(x,d_1)+occ(x,t_2)+occ(x,g_1)+\mathbb{1}_x(x_1)=occ(x,d_1)+occ(x,g_1)+\mathbb{1}_x(x_1)+occ(x,t_2)=occ(x,t_1)+occ(x,t_2).$$

Lorsque $x_1 > x_2$, en appliquant le raisonnement précédent en échangeant t_1 et t_2, on obtient à nouveau $occ(x,t) = occ(x,t_1) + occ(x,t_2)$.
Par conséquent, par principe d'induction sur les arbres binaires, la propriété à démontrer est vraie pour tout couple d'arbres t_1, t_2.
Montrons désormais que t, l'arbre obtenu par fusion de deux arbres croissantes t_1 et t_2, est un arbre croissante par induction.
Si t_1 ou t_2 est vide, alors $t = t_2$ ou $t = t_1$ donc t est croissant.
Supposons que t est la fusion de $t_1 = N(g_1, x_1, d_1)$ et de $t_2 = N(g_2, x_2, d_2)$, que le résultat de la fusion d'arbres croissants de taille strictement inférieure à celle de t est un arbre croissant.
Si $x_1 \leq x_2$, alors, par hypothèse d'induction, la fusion de d_1 et de $N(g_2, x_2, d_2)$ est un arbre croissant. g_1 est aussi croissant car t_1 en est un. Enfin, x_1 est inférieur aux étiquettes de g_1 ; les étiquettes de la fusion de d_1 et de $N(g_2, x_2, d_2)$ sont celles de d_1, qui sont, par croissance de t_1, inférieurs à x_1 et celles de $N(g_2, x_2, d_2)$ sont toutes au minimum x_2 (par croissance de t_2) donc inférieures à x_1 (en vertu de l'hypothèse de $x_1 \leq x_2$). L'arbre obtenu par la fusion de t_1 et t_2 est donc bien croissant.
Si $x_1 > x_2$, de même, l'arbre obtenu par fusion est croissant. Par induction, tout arbre obtenu par fusion d'arbres croissants est croissant.

Question 6

```
let rec ajoute n a = match a with
   | E -> N(E,n,E)
   | N(g,x,d) when x<=n -> N(ajoute n d,x,g)
   | _ -> N(a,n,E);;
```

Question 7

```
let rec supprime_minimum a = match a with
   | N(E,x,E) -> E
   | N(g,_,d) when g=E -> d
   | N(g,_,d) when d=E -> g
   | N(N(g1,x1,d1),x,N(g2,x2,d2)) when x1<=x2 ->
   N(supprime_minimum N(d1,x,N(g2,x2,d2)),x1,g1)
   | N(N(g1,x1,d1),x,N(g2,x2,d2) ->
   N(supprime_minimum N(d2,x,N(g1,x1,d1)),x2,g2)
```

Question 8

```
let ajouts_successifs t =
   let rec aux i a = match i with
   | Array.length t -> a
   | i -> aux (i+1) (ajoute t.(i) a)
   in aux 0 E;;
```

Question 9
Notons $P_{i,j}$ le peigne gauche croissant allant de i à j.
Considérons le tableau `[|n,n-1,...,1|]`. Alors, le premier arbre est $N(E,n,E)$. C'est donc un peigne croissant allant de n à n.
Pour insérer $n-1$, on va fusionner $N(E,n,E)$ et $N(E,n-1,E)$ ce qui donne $N(N(E,n,E),n-1,E)$ qui est le peigne gauche croissant de $n-1$ à n.
Supposons qu'après avoir insérer $n, n-1, \cdots, n-i$, l'arbre obtenu soit le peigne gauche croissant allant de $n-i$ à n. En insérant l'élément suivant, $n-i-1$, on fusionnera le peigne gauche $P_{n-i,n} = N(P_{n-i+1,n}, n-i, E)$ et $N(E, n-i-1, E)$. On obtient $N(P_{n-i,n}, n-i-1, E) = P_{n-i-1,n}$.
Ainsi, par récurrence, l'arbre obtenu sera $P_{1,n}$. La hauteur de cet arbre est n.

Question 10
Etant donné un arbre t, on notera $2t$ (resp. $2t+1$) l'arbre obtenu en multipliant par 2 toutes les étiquettes (resp. en appliquant l'opération $e \mapsto 2e+1$ à toutes les étiquettes).
Montrons par récurrence que, si n est pair, $t_n = N(2t_{n/2}, 1, 2t_{(n-2)/2} + 1)$ et, si n est impair, $t_n = N(2t_{(n-1)/2} + 1, 1, 2t_{(n-1)/2})$.
L'arbre t_1 est $N(E,1,E) = N(t_0, 1, t_0)$. L'arbre t_2 est $N(N(E,2,E),1,E)$. Or $N(E,2,E)$ est $2t_1$ et E est t_0. La propriété est donc vraie pour $n = 1$ et $n = 2$.
Supposons que n est pair, $n = 2p$ et que le résultat vaut pour n. On va fusionner $t_n = N(2t_{n/2}, 1, 2t_{(n-2)/2}+1)$ et $N(E, 2p+1, E)$. On va obtenir $N(fusion(2t_{(n-2)/2}+1, N(E,2p+1,E)), 1, 2t_{n/2})$. Or $fusion(2t_{(n-2)/2}+1, N(E,2p+1,E))$ est $2(fusiont_{(n-2)/2}, N(E,p,E))+1$ qui est aussi $2(fusion(t_{p-1}, N(E,p,E))+1$ ou encore $2t_p + 1 = 2t_{(n+1-1)/2} + 1$. Ainsi $t_{n+1} = N(2t_{(n+1-1)/2} + 1, 1, 2t_{(n+1-1)/2})$. Ceci est le résultat pour $n+1$.
Supposons que n est impair, $n = 2p+1$ et que le résultat vaut pour n. Alors, par la fusion de $t_n = N(2t_{(n-1)/2} + 1, 1, 2t_{(n-1)/2})$ et de $N(E, n+1, E)$, on obtiendra $t_{n+1} = N(fusion(2t_{(n-1)/2}, N(E, 2p+2, E)), 1, 2t_{(n-1)/2} + 1)$. Or $fusion(2t_{(n-1)/2}, N(E, 2p+2, E)) = 2(fusion(t_p, N(E, p+1, E)) = 2t_{p+1} = 2t_{(n+1)/2}$. Ainsi, $t_{n+1} = N(2t_{(n+1)/2}, 1, 2t_{(n+1-2)/2} + 1)$. Ceci est le résultat à établir pour $n+1$.
Par récurrence, la propriété à établir est vraie pour tout $n \geq 1$. On montre alors par récurrence que h_n la hauteur de t_n est $\lfloor \log_2(n) \rfloor + 1$.
t_1 est de hauteur $1 = \log_2(1) + 1$; t_2 est de hauteur $2 = \log_2(2) + 1$.
Soit $n \geq 2$. Supposons que le résultat est vrai pour tout $1 \leq k \leq n$.
Si $n+1$ est pair, alors $t_{n+1} = N(2t_{(n+1)/2}, 1, 2t_{(n-1)/2}+1)$. Or la hauteur de $2t_{(n+1)/2}$ est celle de $t_{(n+1)/2}$ qui est, d'après l'hypothèse de récurrence $\lfloor \log_2((n+1)/2) \rfloor + 1 = \lfloor \log_2(n+1) - 1 \rfloor + 1 = \lfloor \log_2(n+1) \rfloor$. De même, la hauteur de $2t_{(n-1)/2+1}$ est celle de $t_{(n-1)/2}$ qui est, d'après l'hypothèse de récurrence $\lfloor \log_2((n-1)/2) \rfloor + 1 = \lfloor \log_2(n-1) \rfloor$. La hauteur de t_{n+1} est donc $1 + \max(\lfloor \log_2(n+1) \rfloor, \lfloor \log_2(n-1) \rfloor) = 1 + \lfloor \log_2(n+1) \rfloor$.
Si $n+1$ est impair, alors $t_{n+1} = N(2t_{n/2}+1, 1, 2t_{n/2})$. Or la hauteur de $2t_{n/2}+1$ et de $2t_{n/2}$ est celle de $t_{n/2}$ qui est, d'après l'hypothèse de récurrence, $\lfloor \log_2(n/2) \rfloor + 1 = \lfloor \log_2(n) \rfloor$. Ainsi, la hauteur de t_{n+1} est $1 + \lfloor \log_2(n) \rfloor$. Ceci établit le résultat pour $n+1$.
Par récurrence, la hauteur de t_n est bien $\lfloor \log_2(n) \rfloor + 1$ pour tout $n \geq 1$. En outre, la taille de cet arbre est $2n+1$.

Question 11
On va créer une fonction auxiliaire récursive qui calcule la taille et le potentiel d'un arbre.

```
let potentiel a =
```

```
let rec taille_pot a = match a with
   | E -> 0,0
   | N(g,_,d) -> let tg,pg = taille_pot g and td,pd = taille_pot d in
   if tg<td then tg+td+1,pg+pd+1
   else tg+td+1,pg+pd
in snd(taille_pot a);;
```

Notons C_a la complexité de la fonction auxiliaire sur un arbre a. Pour tout x, g, d, on a la relation$C(N(g,x,d)) = C(g) + C(d) + O(1)$. Par conséquent, $C(t) = O(|t|)$.

Question 12

Commençons par vérifier la propriété pour $t_1 = E$ ou $t_2 = E$. Si $t_1 = E$, alors $t = t_2$, $\Phi(t_1) = 0$, $C(t_1, t_2) = 0$ et $\Phi(t_1) + \Phi(t_2) - \Phi(t) + 2(\log_2|t_1| + \log_2|t_2|) = \Phi(t_2) - \Phi(t_2) + 2\log_2|t_2| = 2\log_2|t_2| \geq 0 = C(t_1, t_2)$.
De même, pour $t_2 = E$, alors $t = t_1$, $\Phi(t_2) = 0$, $\Phi(t_1) + \Phi(t_2) - \Phi(t) + 2(\log_2|t_1| + \log_2|t_2|) = \Phi(t_1) - \Phi(t_1) + 2\log_2|t_1| = 2\log_2|t_1| \geq 0 = C(t_1, t_2)$.
Supposons que le résultat à établir soit vrai pour g_1, d_1, g_2, d_2. Soient $t_1 = N(g_1, x_1, d_1)$, $t_2 = N(g_2, x_2, d_2)$, t la fusion de ces deux arbres.
De par la définition de la fonction "fusion", $C(t_1, t_2) \leq 1 + \max(C(d_1, t_2), C(d_2, t_1))$.
Notons t' la fusion de d_1 et t_2. De par l'hypothèse faite sur d_1, $C(d_1, t_2) \leq \Phi(d_1) + \Phi(t_2) - \Phi(t') + 2(\log_2|d_1| + \log_2|t_2|)$.
Supposons pour commencer que x_1 soit lourd pour t_1. Alors $\Phi(t_1) = \Phi(d_1) + \Phi(g_1) + 1$. On a donc, en remarquant également que $|d_1| \leq |t_1|$,

$$C(d_1, t_2) \leq \Phi(t_1) - \Phi(g_1) - 1 + \Phi(t_2) - \Phi(t') + 2(\log_2|d_1| + \log_2|t_2|) \leq -1 + \Phi(t_1) + \Phi(t_2) - (\Phi(g_1 + \Phi(t')) + 2(\log_2|t_1| + \log_2|t_2|).$$

En outre, $t = N(t', x_1, g_1)$; or $|g_1| < |d_1|$ et $|t'| = |d_1 + |t_2| + 1$ donc $|g_1| \leq |t'|$ donc x_1 est léger pour $t = N(t', x_1, g_1)$ donc $\Phi(t) \leq \Phi(t') + \Phi(g_1)$. On a donc $C(d_1, t_2) \leq -1 + \Phi(t_1) + \Phi(t_2) - \Phi(t) + 2(\log_2|t_1| + \log_2|t_2|)$ ou encore

$$1 + C(d_1, t_2) \leq \Phi(t_1) + \Phi(t_2) - \Phi(t) + 2(\log_2|t_1| + \log_2|t_2|).$$

Supposons maintenant que x_2 soit léger pour t_2. Alors $\Phi(t) = \Phi(g_1) + \Phi(d_1)$. De plus, $|d_1| \leq |g_1|$. On a donc $2|d_1| + 1 \leq |t_1|$ donc $|d_1| \leq \frac{|t_1|}{2}$ donc $\log_2|d_1| \leq \log_2|t_1| - 1$. Ainsi

$$C(d_1, t_2) \leq \Phi(t_1) - \Phi(g_1) + \Phi(t_2) - \Phi(t') + 2(\log_2|d_1| + \log_2|t_2|) \leq \Phi(t_1) + \Phi(t_2) - (\Phi(t') + \Phi(g_1)) + 2(\log_2|t_1| - 1 + \log_2|t_2|)$$
$$= -1 + \Phi(t_1) + \Phi(t_2) - (\Phi(t') + \Phi(g_1) + 1) + 2(\log_2|t_1| + \log_2|t_2|).$$

En outre, $t = N(t', x_1, g_1)$ donc $\Phi(t) \leq \Phi(t') + \Phi(g_1) + 1$. Pour conclure,

$$1 + C(d_1, t_2) \leq \Phi(t_1) + \Phi(t_2) - \Phi(t) + 2(\log_2|t_1| + \log_2|t_2|).$$

De même, en échangeant les rôles de t_1 et t_2, $1 + C(d_2, t_1) \leq \Phi(t_1) + \Phi(t_2) - \Phi(t) + 2(\log_2|t_1| + \log_2|t_2|)$. On a donc

$$C(t_1, t_2) \leq 1 + \max(C(d_1, t_2), C(d_2, t_1)) = \max(1 + C(d_1, t_2), 1 + C(d_2, t_1)) \leq \Phi(t_1) + \Phi(t_2) - \Phi(t) + 2(\log_2|t_1| + \log_2|t_2|).$$

De par le principe d'induction sur les arbres binaires, ceci prouve la propriété à établir.

Question 13

La construction de t_{i+1} à partir de t_i demande une complexité $C(t_i, N(E, x_{i+1}, E)) \leq \Phi(t_i) + \Phi(N(E, x_{i+1}, E)) - \Phi(t_{i+1}) + 2(\log_2|t_i| + \log_2|N(E, x_{i+1}, E)|) = \Phi(t_i) - \Phi(t_{i+1}) + 2(\log_2(2i+1) + 1)$.
En additionnant ces inégalités valables pour tout $0 \leq i \leq n-1$, on obtient $\sum_{i=0}^{n-1} C(t_i, t_{i+1}) \leq \Phi(t_0) - \Phi(t_n) + 2\sum_{i=0}^{n-1} \log_2(2i+1) + 2n$.
Le coût total de la construction est $\sum_{i=0}^{n-1} C(t_i, t_{i+1})$; en outre, $\Phi(t_n) \geq 0$, $\Phi(t_0) = 0$ donc le coût total de construction est majoré par $2n + 2\sum_{i=0}^{n-1} \log_2(2i+1) \leq 2n + 2\sum_{i=0}^{n-1} \log_2(2n+1) = 2n\log_2(2n+1) + 2n = O(n\log_2(n))$.

Question 14

Supposons que n impair. Soit $p = \frac{n-1}{2}$ et $x_0 = p$, $x_1 = p$, $x_2 = p-1$, $x_3 = p-1$, $x_4 = p-2$,..., $x_{n-2} = 1$. On note $a_{p,p} = N(N(E,p,E),p,E)$ et, pour tout $k \geq 1$, $a_{k,p}$ est $N(N(E,k,E),k,a_{k+1,p})$. L'arbre obtenu pour les valeurs $x_0, \cdots, x_{n-1}$ précédentes est $a_{1,p}$. Il a hauteur $\frac{n-1}{2}$.
Soit $b_{p,p} = N(N(E,p+1,E),p,N(E,p,E))$ et $b_{k,p} = N(b_{k+1,p},k,N(E,k,E))$ pour tout $k < p$. Alors l'arbre obtenu en insérant $x_{n-1} = p+1$ est $b_{1,p}$. Cette insertion appelle $\frac{n-1}{2} + 1 > \frac{n}{2}$ fois la fonction fusion.
La complexité amortie est la complexité moyenne de chaque opérations : le coût total est $O(n\log_2(n))$, le nombre d'insertions est n, le coût amortie de chaque insertion est $\frac{1}{n}O(n\log_2(n)) = O(\log_2(n))$; c'est pourquoi on dit que la complexité amortie est $O(\log_2(n))$. En revanche, le coût de chaque insertion n'est pas logarithmique ; nous avons vu grâce à l'exemple précédent qu'il peut être linéaire, c'est-à-dire en $\Theta(n)$.

Question 15

Le coût de construction de t_{i+1} est $C(g_i, d_i) \leq \Phi(g_i) + \Phi(d_i) - \Phi(t_{i+1}) + 2(\log_2|g_i| + \log_2|d_i|) \leq \Phi(t_i) - \Phi(t_{i+1}) + 2(\log_2|t_i| + \log_2|t_i|) \leq \Phi(t_i) - \Phi(t_{i+1}) + 4\log_2(2n+1)$. En additionnant ces inégalités, on obtient que le coût total est $\sum_{i=1}^{n-1} C(g_i, d_i) \leq \Phi(t_0) - \Phi(t_n) + 4n\log_2(2n+1)$. Comme $\Phi(t_0) \leq n$ et $\Phi(t_n) = 0$, ce coût est $O(n\log_2(n))$.

Question 16

```
let tri t =
   let a = ref ajouts_successifs t in
   for i=0 to n-1 do
      t.(i) <- minimum !a
      !a:=supprime_minimum !a
   done;;
```

La fonction `ajouts_successifs` est en $O(n\log_2(n))$ d'après la question 13. Le fait de trouver le minimum est en $O(1)$. D'après la question 15, l'ensemble des opérations de suppressions de minimum est en $O(n\log_2(n))$. Par conséquent, la complexité totale est en $O(n\log_2(n))$.

Question 17

Par définition de cette construction, son coût total est

$$C = \sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-1-i}-1} \Phi(t_i^{2j}, t_i^{2j+1}).$$

En utilisant la question 12, on obtient

$$C \leq \sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-i-1}-1} (\Phi(t_i^{2j}) + \Phi(t_i^{2j+1}) - \Phi(t_{i+1}^{j}) + 2(\log_2|t_i^{2j}| + \log_2|t_i^{2j+1}|))$$

Par une récurrence simple, on montre que $|t_i^j| = 2^{i+1}+1$ (montrer puis utiliser pour cela que la taille de la fusion de a_1 et a_2 est $|a_1|+|a_2|-1$). Ainsi

$$C \leq \sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-i-1}-1} \Phi(t_i^{2j}) + \sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-i-1}-1} \Phi(t_i^{2j+1}) - \sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-i-1}-1} \Phi(t_{i+1}^{j}) + 4\sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-i-1}-1} \log_2(2^{i+1}+1).$$

De plus, $\{2j \,/\, 0 \leq j \leq 2^{k-i-1}-1\} \cup \{2j+1 \,/\, 0 \leq j \leq 2^{k-i-1}-1\} = \{j \,/\, 0 \leq j \leq 2^{k-i}-1\}$. On a donc

$$C \leq \sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-i}-1} \Phi(t_i^{j}) - \sum_{i=1}^{k}\sum_{j=0}^{2^{k-i}-1} \Phi(t_i^{j}) + \sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-i-1}-1} \log(2^{i+1}+1) \leq \sum_{j=0}^{2^k-1} \Phi(t_0^j) - \sum_{j=0}^{0} \Phi(t_k^j) + \sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-i-1}-1} \log_2(2^{i+1}+1).$$

En outre, $\Phi(t_0^j) = 0$ pour tout $0 \leq j \leq k-1$; $\Phi(t_k^0) \geq 0$. Ainsi

$$C \leq \sum_{i=0}^{k-1}\sum_{j=0}^{2^{k-i-1}-1} \log_2(2i+1) = \sum_{i=0}^{k-1} 2^{k-i-1}\log_2(2^{i+1}+1).$$

De plus, pour tout i, $\log_2(2^{i+1}+1) \leq i+2$ et

$$\sum_{i=0}^{k-1} 2^{k-i-1}i = \sum_{i=1}^{k-1}\sum_{j=1}^{i} 2^{k-1-i} = \sum_{j=1}^{k-1}\sum_{i=j}^{k-1} 2^{k-1-i} = \sum_{j=1}^{k-1} \frac{2^{k-1-j}-2^{-1}}{1-2^{-1}} = \sum_{j=1}^{k-1}(2^{k-j}-1)$$

$$= \frac{2^{k-1}-1}{1-2^{-1}} - (k-1) = 2^k - 2 - (k-1) = 2^k - k - 1 = O(2^k).$$

De surcroît, $\sum_{i=0}^{k-1} 2^{k-1-i} = \frac{2^{k-1}-2^{-1}}{1-2^{-1}} = 2^k - 1 = O(2^k)$.

On en déduit que $C \leq \sum_{i=0}^{k-1} 2^{k-i-1}\log_2(2^{i+1}+1) \leq \sum_{i=0}^{k-1}(i+2)2^{k-i-1} = O(2^k)$. En conclusion, le coût total est $C = O(2^k)$.

Question 18

On implémente `fusion` puis `construire`

```
let rec fusion t1 t2 = match t1,t2 with
   | E,_ -> t2
   | _,t1 -> t1
   | N(g1,x1,d1),N(_,x2,_) when x1<=x2 -> N((fusion d1 t2),x1,g1)
   | _,N(g2,x2,d2) -> N((fusion d2 t1),x2,g2);;

let construire t =
   let rec aux i j = match i,j with
      | 0,j -> N(E,t.(j),E)
      | i,j -> fusion (aux (i-1) (2*j)) (aux (i-1) (2*j+1))
   in aux (Array.lenght t-1) 0;;
```

Question 19

Soit $k = \lfloor \log_2(n) \rfloor$. On reprend la construction précédente de la suite d'arbres avec $t_0^j = E$ lorsque $j \geq 2^k$. Cela donne

```
let construire2 t =
   let k=int_of_float(log(Array.length t)) in
   let p=int_of_float(2.**k) in
   let rec aux i j = match i,j with
      | 0,j when j<p -> N(E,t.(j),E)
      | 0,j -> E
      | i,j -> fusion (aux (i-1) (2*j)) (aux (i-1) (2*j+1))
   in aux (Array.lenght t-1) 0;;
```