Mini-corrigé du partiel
Louis Granboulan

Exercice 2 : Exponentiation d'exposant aléatoire.

1) La boucle est exécutée k fois, elle fait un carré à chaque fois et une multiplication par bit à 1 dans l'écriture binaire de r. Cela fait donc exactement $\sigma = k$. La valeur exacte de la moyenne $\bar{\mu}$ dépend de la valeur de M, et vaut environ $k/2$, puisqu'un entier entre 0 et M a environ la moitié de ses bits à 1.

   L'algorithme des carrés itérés est donc de performances $[k/2, k, 0]$, à comparer avec la multiplication itérative, de performances $[M/2, 0, 0]$. Si on considère qu'une élévation au carré est au moins aussi rapide qu'une multiplication (l'énoncé n'imposant pas des opérandes distincts pour la multiplication), l'algorithme des carrés itérés est plus rapide dès que $k > 3$.

   On peut aussi obtenir un encadrement de $\bar{\mu}$, par exemple avec le calcul ci-dessous.[1]

   > Définissons la valeur h telle que $2^{h-1} < M \leq 2^h$. L'ensemble des M pour h fixé est l'ensemble de ceux pour lesquels exactement h bits sont nécessaires pour représenter toutes les valeurs possibles de r. Autrement dit $k = \lfloor \log_2 M \rfloor + 1$ et $h = \lceil \log_2 M \rceil$. Remarquons que $h = k$ sauf si M est une puissance de 2, auquel cas $h + 1 = k$.
   >
   > Appelons N le nombre de bits à 1 dans les écritures binaires de toutes les valeurs $r \in [0, M-1]$. On a ainsi $\bar{\mu} = N/M$. Montrons que $\frac{h}{2} - 2 < \bar{\mu} \leq \frac{h}{2}$.
   >
   > Posons $x = \frac{h}{2} - \bar{\mu}$. Supposons $M > 1$ (c'est-à-dire $h \geq 1$) et posons $M' = \lceil M/2 \rceil$, avec les valeurs correspondantes h', N' et x'. Remarquons que $h' = h - 1$.
   >
   > Si $M = 2M'$ est pair, alors on fait correspondre à chaque q dans $[0, M'-1]$ les deux possibilités $r = 2q$ et $r = 2q+1$. On en déduit donc que $N = 2N' + M'$, d'où $x = x'$.
   >
   > Si $M = 2M' - 1$ est impair, et si $\alpha \in [1, h']$ est le nombre de bits à 1 dans l'écriture binaire de $M - 1$, alors on a $N = 2N' + M' - \alpha$, d'où $x = x' + \alpha/M$. Or $\alpha \leq h'$ et $M > 2^{h'}$, donc $0 \leq x - x' < h' 2^{-h'}$.
   >
   > La valeur $M = 1$ donne $x = 0$, donc on en déduit que pour tout M on a $0 \leq x < \sum_{i=0}^{h-1} i 2^{-i} < 2$.

2)
```
/* T[i]=g^{2^i} est précalculé */
Exp2(r){
    a=1;
    for(i=0;i<k;i++)
        if(r&(1<<i)) /* i-ième bit */
            a=(a*T[i]);
    return a;
}
```
   De performances $[k/2, 0, k]$.

3.a) En fonction de h défini ci-dessus (et valant presque toujours k) le nombre de chiffres de la représentation de r en base b est $t = \lceil h/l \rceil = \lfloor (h-1)/l \rfloor + 1$

3.b)
```
/* U[i]=g^{b^i} est précalculé ; b=2^l */
Exp3(r){
    a=1;
    for(i=0;i<t;i++){
        for(j=0;j<r%b;j++){
            a=(a*U[i]);
        r=r/b;
    }
    return a;
}
```
   Le nombre de multiplications effectuées est $\sum_i u_i$. Comme chaque u_i vaut en moyenne approximativement $(b-1)/2$, l'algorithme est de performance $[t(b-1)/2, 0, t]$, ce qui est toujours plus lent que `Exp2`, mais utilise moins de mémoire.

[1] Certaines copies ont prétendu montrer que $\frac{k-1}{2} < \bar{\mu}$, mais c'est faux, car par exemple si $M = 9$, alors $k = 4$ et $\bar{\mu} = 13/9 < 3/2$.

Chaque $g^{u_i b^i} = U[i]^{u_i}$ est calculé par multiplication itérative. On peut donc vouloir utiliser l'algorithme des carrés itérés, ce qui donne l'algorithme ci-dessous.

```
/* U[i]=g^{b^i} est précalculé ; b=2^l */
Exp3'(r){
    a=1;
    for(i=0;i<t;i++){
        a=(a*Exp(U[i],r%b)));
        r=r/b;
    }
    return a;
}
```

On calcule chaque $g^{u_i b^i} = U[i]^{u_i}$ en appelant la fonction Exp de l'énoncé. Chaque appel de Exp fait ainsi en moyenne $l/2$ multiplications et l carrés. Cet algorithme est donc de performance moyenne $[k/2+t, k, t]$, ce qui est toujours plus lent que Exp, et utilise plus de mémoire.

4.a) On montre par récurrence que $z_{j+1} = g^{\sum_{u_i>j}(u_i-j)b^i}$. C'est vrai pour $z_b = 1$ puisque $u_i \leq b-1$. Et $z_j = z_{j+1} * y_j = g^{\sum_{u_i>j}(u_i-j)b^i} g^{\sum_{u_i\geq j} b^i} = g^{\sum_{u_i\geq j}(u_i-j+1)b^i}$.

Donc $z_1 = g^{\sum_{u_i>0} u_i b^i} = g^r$

4.b)
```
/* U[i]=g^{b^i} est précalculé ; b=2^l */
Exp4(r){
    /* Calcul des valeurs u_i */
    for(i=0;i<t;i++){
        u[i]=r%b; r=r/b;
    }
    /* Calcul des valeurs y_i, et de g^r */
    y=1; a=1;
    for(j=b-1;j>0;j--){
        for(i=0;i<t;i++)
            if (u[i]==j)
                y=(y*U[i]);
        a=(a*y);
    }
    return a;
}
```

Pour le calcul des y_i, chaque U[i] intervient au plus une fois, et uniquement si $u_i > 0$. Ce calcul fait donc exactement m multiplications. Le calcul de $g^r = z_1$ fait exactement $b-1$ multiplications. On peut éliminer la multiplication par 1 au début du calcul des y_i et la multiplication par 1 au début du calcul de g^r. Remarquons qu'il n'est pas nécessaire de stocker les y_i dans un tableau, ce qui économise de la mémoire...

4.c) Ce nouvel algorithme est de performances $[m+b-3, 0, t]$. On a $m = t(1-\frac{1}{b})$, donc si b est suffisamment grand, $m \simeq t$. Le nombre total d'opérations dans G est donc environ $t+b$, à comparer avec $t+3k/2$ ou $t(b-1)/2$ pour la question précédente.

Exp4 est meilleur que Exp3 si $t+b-3 < t(b-1)/2$, c'est-à-dire $b > \frac{t-6}{t-2}$, ce qui est vrai pour tout t.

Exp4 est meilleur que Exp et donc que Exp3' si $t+b < 3k/2$, c'est-à-dire $t > \frac{2^{l+1}}{3l-2}$, ce qui est vrai dès que b est suffisamment inférieur à $3k/2$

4.d) Si b est suffisamment petit, c'est Exp4 le meilleur des algorithmes vus précédemment.

Pour une valeur k donnée, on veut minimiser le nombre d'opérations $t+b = k/l+2^l$. Ce minimum est atteint pour l tel que $k = l^2 2^l \ln 2$, c'est-à-dire $b \simeq k/(\log_2 k)^2/\ln 2$. Par exemple pour $k = 1024$, cela fait $b \simeq 2^7$, et comme performances environ $[275, 0, 150]$, ce qui est mieux que Exp2 qui nécessite $[512, 0, 1024]$.

5.a) Notons $\gamma_{j,k} = e^{-2\pi i k \alpha_j/M}$. On a ainsi $e^{-2\pi i k r/M} = \prod_{j=1\ldots n} \gamma_{j,k}^{u_j}$. Comme les u_j sont tirés indépendamment, l'espérance du produit est égal au produit des espérances.

Étant donné que u_j vaut 0 avec probabilité $1-\beta$ et chaque valeur entre 1 et $b-1$ avec probabilité $\frac{\beta}{b-1}$, l'espérance de $\gamma_{j,k}^{u_j}$ est égale à $(1-\beta)+\frac{\beta}{b-1}\sum_{\ell=1\dots b-1}\gamma_{j,k}^{\ell}$.
On en déduit la formule de l'énoncé, qui est vraie aussi pour $k=0$.

5.b) Il faut faire attention car les E_k ne sont pas des variables indépendantes. Par exemple, s'il existe $k>0$ tel que $E_k=1$, alors c'est vrai pour tout k. Mais cette non-indépendance n'empêche pas que l'espérance d'une somme soit égale à la somme des espérances. Il suffit donc de calculer l'espérance des normes $|E_k|^2$, qui sont le produit de n variables indépendantes, donc il suffit de calculer l'espérance des variables $X_{j,k}$ ci-dessous.
$X_{j,k}=\left|(1-\beta)+\frac{\beta}{b-1}\sum_\ell \gamma_{j,k}^{\ell}\right|^2=(1-\beta)^2+\frac{(1-\beta)\beta}{b-1}\sum_\ell(\gamma_{j,k}^{\ell}+\gamma_{j,k}^{-\ell})+\frac{\beta^2}{(b-1)^2}\sum_{\ell,\ell'}\gamma_{j,k}^{\ell-\ell'}$.
Or l'espérance de $\gamma_{j,k}$ étant nulle, il ne reste que le premier terme et ceux où $\ell=\ell'$.
L'espérance de $X_{j,k}$ est donc égale à B, d'où on déduit la formule de l'énoncé.

5.c) Si X est de probabilité uniforme δ il contient δM éléments, et $\hat{f}(0)=\delta M$. De plus $E_0=1$, donc la différence cherchée est $(\sum_{k=0}^{M-1}\frac{1}{M}\hat{f}(k)E_k)-\delta=\sum_{k=1}^{M-1}\frac{1}{M}\hat{f}(k)E_k\le\frac{1}{M}\sum_{k=1}^{M-1}|\hat{f}(k)|\cdot|E_k|$.
Le carré de cette somme de produits est inférieur au produit des sommes des carrés (Cauchy-Schwartz), c'est-à-dire $\frac{1}{M^2}\sum_{k=1}^{M-1}|\hat{f}(k)|^2\sum_{k=1}^{M-1}|E_k|^2$.
En outre, $\|f\|^2=\frac{1}{M}\sum_q\epsilon_q=\delta$, donc $\sum_{k=0}^{M-1}|\hat{f}(k)|^2=\|\hat{f}\|^2=M^2\delta$. En posant $\lambda(\alpha_1,\cdots,\alpha_n)=\sum_{k=1}^{M-1}|E_k|^2$ on obtient le résultat cherché.

5.d)
```
/* V[i]=g^{\alpha_i} est de taille n et précalculé */
Exp5(){
    /* Calcul des valeurs u_i */
    for(i=1;i<=n;i++)
        u[i]=RANDOM(b,\beta);
    /* Calcul des valeurs y_i */
    y[b]=1;
    for(j=b-1;j>0;j--){
        y[j]=y[j+1];
        for(i=1;i<=n;i++)
            if (u[i]==j)
                y[j]=(y[j]*V[i]);
    }
    /* Calcul de r */
    r=0;
    for(j=1;j<=n;j++)
        r=(r+a[j]*u[j])%M;
    /* Calcul de g^r */
    a=1;
    for(j=1;j<b;j++)
        a=(a*y[j]);
    return (r,a);
}
```
Cet algorithme dépend de trois paramètres : l'entier n, la puissance de deux b et la probabilité β. Il est de performance moyenne $[\beta n+b-1,0,n]$ et chaque couple (r,g^r) possible apparaît avec une probabilité de l'ordre de $\frac{1}{M}\pm B^{n/2}$ où $B=(1-\beta)^2+\frac{\beta^2}{b-1}$.
Pour que cette méthode permette de produire un couple (r,g^r) avec une distribution suffisamment proche de la distribution uniforme, il faut que B soit assez proche de 0. Si b est fixé, la valeur minimale de B est atteinte pour $\beta=1-\frac{1}{b}$, ce qui correspond à un tirage uniforme des u_j et n'est donc pas meilleur que `Exp4`. La dégradation correspondante est $B^{n/2}\simeq\frac{1}{\sqrt{M}}$.
En revanche, prenons comme précédemment $k=1024$ et $n=150$. Si $b=32$ et $\beta=1/2$, alors $B\simeq 1/4$ et la dégradation est environ 2^{-150}, ce qui est suffisant pour certaines applications, et l'algorithme est sensiblement plus rapide que `Exp4` : $[110,0,150]$