Corrigé

Examen d'algorithmique

EPITA ING1 2015 S1; A. DURET-LUTZ

Durée : 1h30

janvier 2012

1 Bouclettes (3 points)

Combien de lignes sont affichées par les boucles suivantes ? Donnez votre réponse en fonction de N.

```
for (int i = 2; i < N; ++i)
  for (int j = 3; j <= i; ++j)
    puts("Boucle 1");
```

Réponse :

Aucune ligne n'est affichée pour $i < 3$, on peut donc faire commencer la première boucle à $i = 3$.

$$\sum_{i=3}^{N-1} \sum_{j=3}^{i} 1 = \sum_{i=3}^{N-1} (i-2) = \frac{(N-2)(N-3)}{2}$$

```
for (int i = 0; i < N; ++i)
  for (int j = 1; j <= N; j *= 2)
    puts("Boucle 2");
```

Réponse :

Comme j prend des valeurs de la forme 2^k on pose $k = \log_2 j$.
La boucle sur j devient donc `for (int k = 0; k <=` $\log_2$`(N); ++k)`.

$$\sum_{i=0}^{N-1} \sum_{k=0}^{\lfloor \log_2 N \rfloor} 1 = N \times (1 + \lfloor \log_2 N \rfloor)$$

2 Function recursive (4 points)

On considère la fonction suivante.

```
int f(int n)
{
  int i, x;
  if (n < 0)
    return 1;
  x = 0;
```

```
  for (i = 0; i <= n; ++i)
    x = x + i;
  return x + f(n - 1);
}
```

1. **(2pts)** Calculez sa complexité en fonction de n.

Réponse :

On a $\begin{cases} T(0) = \Theta(1) \\ T(n) = \Theta(n) + T(n-1) \end{cases}$

Conclure immédiatement que $T(n) = \Theta(n^2)$ est tout à fait acceptable.
Pour ceux qui veulent une preuve plus détaillée, le théorème général ne applique pas ici car l'équation n'est pas de la bonne forme. On procède donc par itération successives. On commence par remplacer $\Theta(n)$ par cn dans l'équation, puis on remplace $T(n-1)$ par sa définition, en itérant ainsi jusqu'à trouver une forme générale.

$$\begin{aligned} T(n) &= cn + T(n-1) \\ &= cn + c(n-1) + T(n-2) \\ &= cn + c(n-1) + c(n-2) + T(n-3) \\ &\vdots \end{aligned}$$

après $i-1$ substitutions :

$$\begin{aligned} &= cn + c(n-1) + \cdots + c(n-(i-1)) + T(n-i) \\ &= T(n-i) + c\sum_{k=0}^{i-1}(n-k) \end{aligned}$$

on s'arrête lorsque $i = n$ car alors $T(0) = \Theta(1)$

$$\begin{aligned} &= \Theta(1) + c\sum_{k=0}^{n-1}(n-k) \\ &= \Theta(1) + c\sum_{k=1}^{n}(k) \\ &= \Theta(1) + cn(n+1)/2 \\ &= \Theta(n^2) \end{aligned}$$

2. **(2pts)** Proposez une implémentation en $\Theta(1)$ de cette même fonction.

Réponse :

Pour un n donné, la boucle calcule $x = \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

Il faut donc maintenant supprimer la récursion. On a

$$\begin{cases} f(n) = \frac{n(n+1)}{2} + f(n-1) \\ f(0) = 1 \end{cases}$$

En remplaçant la définition de f par itérations successives (comme ci-dessus) on trouve :

$$\begin{aligned} f(n) &= \sum_{k=1}^{n} \frac{k(k+1)}{2} + f(0) \\ &= \frac{1}{2}\sum_{k=1}^{n} k^2 + \frac{1}{2}\sum_{k=1}^{n} k + 1 \\ &= \frac{n(n+1)(2n+1)}{12} + \frac{n(n+1)}{4} + 1 \\ &= \frac{n(n+1)(n+2)}{6} + 1 \end{aligned}$$

Cela nous donne une implémentation en temps constant de f :

```
int f(int n)
{
  if (n < 0)
    return 1;
  return n*(n+1)*(n+2)/6 + 1;
}
```

3 Chaîne de lettres (4 points)

(Note : l'*exécution* du système pyramidal décrit ici est interdite en France ainsi que dans de nombreux pays, mais son *étude* ne l'est pas...)

Vous recevez une "lettre chaîne" contenant une liste de $N \geq 1$ noms (et adresses), avec les instructions suivantes :

- envoyez 1 euro à la personne en tête de la liste ;
- supprimez la tête de la liste, et ajoutez-vous en fin de liste (ainsi la liste contient toujours N personnes) ;
- envoyez une copie de cette liste et de ces instructions à $P \geq 1$ personnes ;
- surveillez votre boîte aux lettres, vous recevrez bientôt beaucoup plus d'argent.

On suppose :

- que ces lettres sont toujours envoyées à P personnes qui ne l'ont jamais reçue auparavant,
- que chaque personne respecte les consignes scrupuleusement.

1. **(1pt)** Quelle est la somme que vous recevrez si $N = 10$ et $P = 2$.

Réponse :

Si l'on représente cette de lettre chaîne par un arbre dont vous être la racine, et dans lequel chaque nœud possède P fils représentant les destinataires des lettres, vous recevrez un euro de chaque nœud à la profondeur N.
Il y a $P^N = 2^{10} = 1024$ personnes à cette profondeur, donc autant d'euros reçus.

2. **(1.5pt)** Après que vous ayez envoyé vos P lettres, combien de personnes vont recevoir une liste contenant votre nom (à n'importe quelle position) ? Donnez votre réponse en fonction de N et P.

Réponse :

P personnes reçoivent la liste avec votre nom en position N,
P^2 personnes reçoivent la liste avec votre nom en position $N-1$,
P^3 personnes reçoivent la liste avec votre nom en position $N-2$,
...
P^N personnes reçoivent la liste avec votre nom en position 1 (en tête).
On calcule donc

$$\sum_{i=1}^{N} P^i = \left(\sum_{i=0}^{N} P^i\right) - 1 = \frac{P^{N+1}-1}{P-1} - 1 = \frac{P^{N+1}-P}{P-1}$$

Note : la formule pour la somme entre parenthèses était dans l'annexe.

3. **(1.5pt)** Dans le cas où $P = 10$, quelle est la plus petite valeur de N à partir de laquelle le nombre de personnes devant recevoir une liste qui possède votre nom (en n'importe quelle position) dépasse la population mondiale (7 milliards de personnes). Non, vous n'avez pas besoin de calculatrice.

Réponse :

On cherche N tel que

$$\frac{P^{N+1} - P}{P - 1} = 7 \cdot 10^9$$

puis on arrondira à l'entier supérieur.

$$\begin{aligned}
\frac{10^{N+1} - 10}{10 - 1} &= 7 \cdot 10^9 \\
10^{N+1} - 10 &= 63 \cdot 10^9 \\
10^N - 1 &= 6,3 \cdot 10^9 \\
10^N &= (6,3 \cdot 10^9) + 1 \\
N &= \log_{10}((6,3 \cdot 10^9) + 1)
\end{aligned}$$

et comme $10^9 < (6,3 \cdot 10^9) + 1 < 10^{10}$

$$9 < N < 10$$

Conclusion : pour $P = 10$ et $N \geq 10$ la population terrestre ne suffit pas.
Sans s'embarquer dans des histoires de logarithmes, il était aussi possible de trouver la réponse en calculant le nombre de personnes touchées ($(P^{N+1} - P)/(P - 1)$) pour différents N :

pour $N =$ 1 on touche	10 personnes
pour $N =$ 2 on touche	110 personnes
pour $N =$ 3 on touche	1 110 personnes
pour $N =$ 4 on touche	11 110 personnes
pour $N =$ 5 on touche	111 110 personnes
pour $N =$ 6 on touche	1 111 110 personnes
pour $N =$ 7 on touche	11 111 110 personnes
pour $N =$ 8 on touche	111 111 110 personnes
pour $N =$ 9 on touche	1 111 111 110 personnes
pour $N =$10 on touche	11 111 111 110 personnes

Les 7 milliards sont donc dépassés à partir de $N = 10$.

4 Omelette (11 points)

Vous êtes au pied d'un grand immeuble avec en poche plusieurs œufs identiques et difficiles à casser (poules élevées en plein air dans une cimenterie). Le but est de concevoir une stratégie pour trouver le plus haut étage duquel on puisse lâcher un œuf sans qu'il ne se casse, en effectuant le moins de lâchers possibles. (Attention, le but n'est pas de casser le moins d'œufs possible, mais bien d'en *lâcher* le moins possible.)

On note H le nombre de niveaux de l'immeuble, et N le nombre d'œufs en poche. On numérotera les étages de 1 (rez-de-chaussée) à H (dernier étage).

On fait plusieurs hypothèses :

- Un œuf qui survit à une chute peut être utilisé à nouveau.
- Tous les œufs du lots se comportent de la même façon.
- Si un œuf lâché de l'étage i se casse, alors tout œuf lâché d'un étage $j \geq i$ se cassera.

- Inversement, si un œuf lâché de l'étage i ne se casse pas, alors aucun œuf lâché d'un étage $j \leq i$ ne se cassera.
- Il n'est pas exclu que les œufs puissent se casser dès le rez-de-chaussée, et il n'est pas exclu non-plus que les œufs puissent survivre à une chute du dernier étage.

Si $N = 1$ (un seul œuf en poche) il n'y a qu'un stratégie possible : on lâche l'œuf depuis l'étage 1, puis depuis l'étage 2, puis 3, etc. jusqu'à ce que soit l'œuf casse, soit il survive à une chute du dernier étage H. Dans le pire des cas, cela demande H lâchers.

2 œufs

Avec $N = 2$ œufs en poche, on peut considérer plusieurs stratégies.

Stratégie 1 : On tente de lâcher le premier œuf tous les $\sqrt{H}$ étages, puis on utilise le second œuf pour trouver l'étage précis avec une recherche linéaire comme précédemment. Par exemple si $H = 36$, on lâche le premier œuf à l'étage 6, puis aux étages 12, 18, etc. jusqu'à ce qu'il casse. Si par exemple il casse à l'étage 24, on utilise le second œuf pour tester les étages 19 à 23 l'un après l'autre.

1. **(1pt)** Combien de lâchers d'œufs la stratégie 1 demande-t-elle dans le pire des cas ? Donnez une formule en fonction de H.

Réponse :

Le pire cas est lorsque l'œuf se casse à partir de l'étage $H - 1$. $\sqrt{H}$ lâchers pour le premier œuf (qui se casse à l'étage H), plus $\sqrt{H} - 1$ pour le second œuf. Cela fait un total de $2\sqrt{H} - 1$ lâchers.

Stratégie 2 : On fait en sorte que plus on a fait de tentatives avec le premier œuf, moins il faudra en faire avec le second. Par exemple si $H = 36$, on effectue les lâchers du premier œuf aux étages $h_1 = 8$, $h_2 = 15$, $h_3 = 21$, $h_4 = 26$, $h_5 = 30$, $h_6 = 33$, $h_7 = 35$, $h_8 = 36$, de façon à ce que si le premier œuf casse à l'étage h_i, il ne reste que $h_1 - i$ tests à réaliser avec le second œuf (remarquez comme l'écart entre h_i et h_{i+1} diminue de un à chaque fois que i augmente).

2. **(2pt)** Expliquez comment trouver h_1 en fonction de H.

Réponse :

Idéalement, on cherche h_1 tel que $h_1 + (h_1 - 1) + (h_1 - 2) + \cdots + 1 = H$.
Évidement, on pourrait faire un programme pour soustraire de H tous les entiers successifs ($H - 1$ puis $H - 1 - 2$ puis $H - 1 - 2 - 3$, etc.) jusqu'à ce que ce calcul devienne nul ou négatif. C'est juste **très dommage** de faire une boucle quand on peut s'en passer.

$$\begin{aligned} h_1 + (h_1 - 1) + (h_1 - 2) + \cdots + 1 &= H \\ \frac{h_1(h_1 + 1)}{2} &= H \\ h_1^2 + h_1 - 2H &= 0 \end{aligned}$$

Magnifique équation du second degré qui rappelle de vieux souvenirs du lycée.

$$h_1 = \frac{-1 \pm \sqrt{1 + 8H}}{2}$$

Évidement la solution négative n'a pas de sens pour nous, il reste donc :

$$h_1 = \frac{\sqrt{1 + 8H} - 1}{2}$$

On peut vérifier que pour $H = 36$ on retrouve bien $h_1 = 8$.

3. **(1pt)** Combien de lâchers d'œufs la stratégie 2 demande-t-elle dans le pire des cas ? Donnez une formule en fonction de h_1.

Réponse :

Si le premier œuf se casse à l'étage h_i, on a déjà effectué i lâchers, et il en reste encore au pire $h_1 - i$ à faire avec le second œuf. Cela fait donc h_1 lâchers au total.

Programmation dynamique pour N œufs

On généralise maintenant le problème à N œufs, et on cherche le nombre minimum de lâchers nécessaire pour couvrir tous les cas. (C'est-à-dire un nombre tel qu'il n'existe pas de pire cas qui en demanderait plus.)

On note $W(n,h)$ le nombre minimum de lâchers qu'il faut faire dans le pire cas lorsqu'on a $n \in [\![0,N]\!]$ œufs en poche, et encore $h \in [\![0,H]\!]$ étages consécutifs à tester.

On a alors :

$$\begin{aligned} & W(n,0) = 0 \\ \forall h > 0, \quad & W(0,h) = \infty \\ \forall n > 0,\ \forall h > 0, \quad & W(n,h) = 1 + \min_{x \in [\![1,h]\!]} \max(W(n-1,x-1), W(n,h-x)) \end{aligned}$$

1. **(3pts)** Justifiez la définition de $W(n,h)$ ci-dessus. (À quoi correspondent les arguments du max ? Pourquoi un max, pourquoi un min ? Que représente x ?)

Réponse :

x compris entre 1 et k, représente un étage où l'on évalue le coût d'un lâcher. On considère la meilleure de toutes les stratégies possibles, d'où le min sur tous les x. Pour chaque position x, deux cas sont possibles :

- soit l'œuf se casse, il reste à tester les $x-1$ étages inférieurs avec un œuf de moins : $W(n-1,x-1)$;
- soit l'œuf ne se casse pas, et il reste à tester les $h-x$ étages suivants avec autant d'œufs : $W(n,h-x)$.

Comme on cherche le pire scénario, on prend le maximum du nombre de lâchers nécessaires dans ces deux cas.
Le $+1$ en tête de formule compte évidement le lâcher qu'on vient de considérer.

2. **(3pts)** Completez le tableau suivant avec les valeurs de $W(n,h)$ pour tout $n \in [\![0,3]\!]$ et $h \in [\![0,10]\!]$.

	$h=0$	1	2	3	4	5	6	7	8	9	10
$n=0$	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$n=1$	0	1	2	3	4	5	6	7	8	9	10
$n=2$	0	1	2	2	3	3	3	4	4	4	4
$n=3$	0	1	2	2	3	3	3	3	4	4	4

3. **(1pts)** Quelle est la complexité de calculer $W(N,H)$, en fonction de N et H ?

Réponse :

Il faut évidement remplir les $(N+1) \times (H+1)$ cases du tableau ci-dessus. Cependant chaque case (x,h) demande $\Theta(h)$ opérations à cause du min sur tous les x.
Cela nous fait donc

$$\sum_{n=0}^{N} \sum_{h=0}^{H} \Theta(h) = \sum_{n=0}^{N} \Theta(H^2) = \Theta(H^2 N)$$