Nom :

Prénom :

Examen d'algorithmique

EPITA ING1 2015 S1; A. DURET-LUTZ

Durée : 1h30

janvier 2012

Consignes

- Cette épreuve se déroule **sans document** et **sans calculatrice**.
- Répondez sur le sujet dans les cadres prévus à cet effet.
- Soignez votre écriture, et ne donnez pas plus de détails que ceux nécessaires à vous justifier.
- Il y a 5 pages d'énoncé et une page d'annexe.
- Le barème, indicatif, correspond à une note sur 22.

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 :

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

Réponse :

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 :

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

Réponse :

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 :

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 :

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 :

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 :

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 :

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 :

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 :

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										
$n=3$	0										

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

Réponse :

C'est déjà fini...

Notations asymptotiques

$$\mathrm{O}(g(n)) = \{f(n) \mid \exists c \in \mathbb{R}^{+\star}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq f(n) \leq cg(n)\}$$
$$\Omega(g(n)) = \{f(n) \mid \exists c \in \mathbb{R}^{+\star}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq cg(n) \leq f(n)\}$$
$$\Theta(g(n)) = \{f(n) \mid \exists c_1 \in \mathbb{R}^{+\star}, \exists c_2 \in \mathbb{R}^{+\star}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \iff g(n) \in \mathrm{O}(f(n)) \text{ et } f(n) \notin \mathrm{O}(g(n)) \qquad f(n) \in \mathrm{O}(g(n)) \iff g(n) \in \Omega(f(n))$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \iff f(n) \in \mathrm{O}(g(n)) \text{ et } g(n) \notin \mathrm{O}(f(n)) \qquad f(n) \in \Theta(g(n)) \iff \begin{cases} f(n) \in \Omega(g(n)) \\ g(n) \in \Omega(f(n)) \end{cases}$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \in \mathbb{R}^{+\star} \iff f(n) \in \Theta(g(n))$$

Ordres de grandeurs

constante	$\Theta(1)$	
logarithmique	$\Theta(\log n)$	
polylogarith.	$\Theta((\log n)^c)$	$c > 1$
	$\Theta(\sqrt{n})$	
linéaire	$\Theta(n)$	
	$\Theta(n \log n)$	
quadratique	$\Theta(n^2)$	
	$\Theta(n^c)$	$c > 2$
exponentielle	$\Theta(c^n)$	$c > 1$
factorielle	$\Theta(n!)$	
	$\Theta(n^n)$	

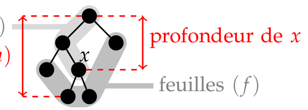
Théorème général

Soit à résoudre $T(n) = aT(n/b) + f(n)$ avec $a \geq 1$, $b > 1$

- Si $f(n) = \mathrm{O}(n^{\log_b a - \varepsilon})$ pour un $\varepsilon > 0$, alors $T(n) = \Theta(n^{\log_b a})$.
- Si $f(n) = \Theta(n^{\log_b a})$, alors $T(n) = \Theta(n^{\log_b a} \log n)$.
- Si $f(n) = \Omega(n^{\log_b a + \varepsilon})$ pour un $\varepsilon > 0$, et si $af(n/b) \leq cf(n)$ pour un $c < 1$ et tous les n suffisamment grands, alors $T(n) = \Theta(f(n))$.

(Note : il est possible de n'être dans aucun de ces trois cas.)

Arbres



(nœuds $n = ni + f$)

Pour tout arbre binaire :

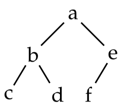
$n \leq 2^{h+1} - 1$ $\qquad h \geq \lceil \log_2(n+1) - 1 \rceil = \lfloor \log_2 n \rfloor$ si $n > 0$

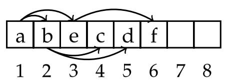
$f \leq 2^h$ $\qquad h \geq \lceil \log_2 f \rceil$ si $f > 0$

$f = ni + 1$ (si l'arbre est *complet* = les nœud internes ont tous 2 fils)

Dans un arbre binaire équilibré une feuille est soit à la profondeur $\lfloor \log_2(n+1) - 1 \rfloor$ soit à la profondeur $\lceil \log_2(n+1) - 1 \rceil = \lfloor \log_2 n \rfloor$. Pour ces arbres $h = \lfloor \log_2 n \rfloor$.

Un arbre parfait (= complet, équilibré, avec toutes les feuilles du dernier niveau à gauche) **étiqueté** peut être représenté par un tableau.





Les indices sont reliés par :

Père$(y) = \lfloor y/2 \rfloor$

FilsG$(y) = y \times 2$

FilsD$(y) = y \times 2 + 1$

Identités utiles

$$\sum_{k=0}^{n} k = \frac{n(n+1)}{2}$$

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1} \qquad \text{si } x \neq 1$$

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \qquad \text{si } |x| < 1$$

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2} \qquad \text{si } |x| < 1$$

$$\sum_{k=1}^{n} \frac{1}{k} = \Theta(\log n)$$

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

Définitions diverses

La complexité d'un problème est celle de l'algorithme le plus efficace pour le résoudre.

Un tri stable préserve l'ordre relatif de deux éléments égaux (au sens de la relation de comparaison utilisée pour le tri).

Un tri en place utilise une mémoire temporaire de taille constante (indépendante de n).

Rappels de probabilités

Espérance d'une variable aléatoire X **:** C'est sa valeur attendue, ou moyenne. $\mathrm{E}[X] = \sum_x \Pr\{X = x\}$

Variance : $\mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2] - \mathrm{E}^2[X]$

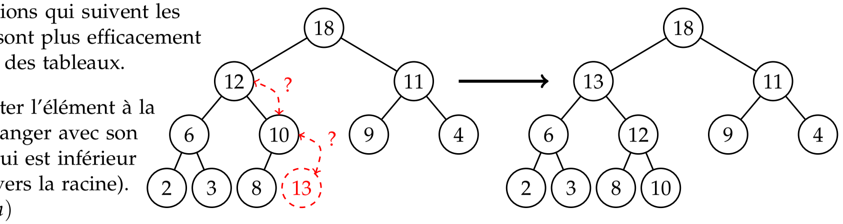
Loi binomiale : On lance n ballons dans r paniers. Les chutes dans les paniers sont équiprobables ($p = 1/r$). On note X_i le nombre de ballons dans le panier i. On a $\Pr\{X_i = k\} = \mathrm{C}_n^k p^k (1-p)^{n-k}$. On peut montrer $E[X_i] = np$ et $\mathrm{Var}[X_i] = np(1-p)$.

Tas

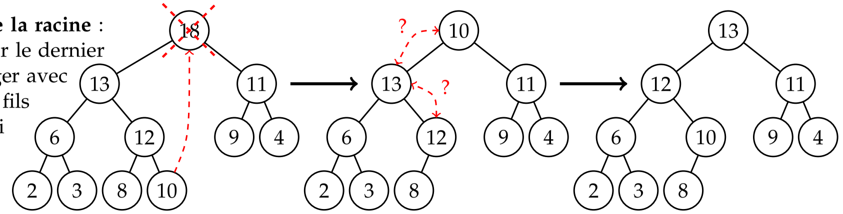
Un tas est un arbre parfait partiellement ordonné : l'étiquette d'un nœud est supérieure à celles de ses fils.

Dans les opérations qui suivent les arbres parfaits sont plus efficacement représentés par des tableaux.

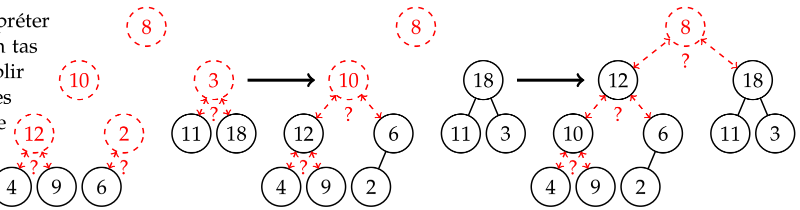
Insertion : ajouter l'élément à la fin du tas, l'échanger avec son père tant qu'il lui est inférieur (en remontant vers la racine). $T_{\text{insert}} = \mathrm{O}(\log n)$



Suppression de la racine : la remplacer par le dernier nœud ; l'échanger avec son plus grand fils tant que celui-ci est plus grand. $T_{\text{rem}} = \mathrm{O}(\log n)$



Construction : interpréter le tableau comme un tas (incorrect) puis rétablir l'ordre en partant des feuilles (vues comme des tas corrects). $T_{\text{build}} = \Theta(n)$

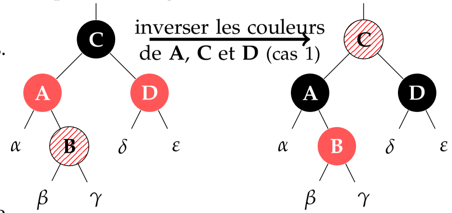


Arbres Rouge et Noir

Les ARN sont des arbres binaires de recherche dans lesquels : (1) un nœud est rouge ou noir, (2) racine et feuilles (NIL) sont noires, (3) les fils d'un nœud rouge sont noirs, et (4) tous les chemins reliant un nœud à une feuille (de ses descendants) contiennent le même nombre de nœuds noirs (= la *hauteur noire*). Ces propriétés interdisent un trop fort déséquilibre de l'arbre, sa hauteur reste en $\Theta(\log n)$.

Insertion d'une valeur : insérer le nœud avec la couleur rouge à la position qu'il aurait dans un arbre binaire de recherche classique, puis, si le père est rouge, considérer les trois cas suivants dans l'ordre.

Cas 1 : Le père et l'oncle du nœud considéré sont tous les deux rouges. **Répéter** cette transformation à partir du grand-père si l'arrière grand-père est aussi rouge.



(Dans tous ces cas, les lettres grecques représentent des sous-arbres avec la même hauteur noire.)

Cas 2 : Si le père est rouge, l'oncle noir, et que le nœud courant n'est pas dans l'axe père–grand-père, une rotation permet d'aligner fils, père, et grand-père.

Cas 3 : Si le père est rouge, l'oncle noir, et que le nœud courant est dans l'axe père–grand-père, un rotation et une inversion de couleurs rétablissent les propriétés des ARN.