Algorithmique avancée : Partiel du Jeudi 26 Octobre 2017.

Durée 2h. Seul document autorisé : 1 feuille A4 recto-verso.

Le barème est donné à titre indicatif. Toutes les réponses doivent être soigneusement **justifiées**. La compréhension du sujet faisant partie de l'épreuve, **on ne répondra à aucune question**. Si vous rencontrez des ambiguïtés, vous expliquerez **sur votre copie** comment vous les interprétez.

I - Médiane de deux tableaux (17 points)

La médiane d'un tableau de taille paire est un élément M de ce tableau, tel qu'il est possible de partitionner le tableau en 2 tableaux de même taille : le premier ne contient que des éléments inférieurs ou égaux à M, et le deuxième que des éléments supérieurs ou égaux à M. Par exemple, le tableau | 12 | 45 | 3 | 54 | 21 | 45 | 11 | 7 | possède deux médianes possibles, 12 et 21 :

| 12 | 3 | 11 | 7 |

(12 ou 21)

| 45 | 54 | 21 | 45 |

Soient $A[1..m]$ et $B[1..m]$ deux tableaux triés par ordre croissant. On cherche à trouver une médiane des $2 \times m$ éléments de ces deux tableaux. L'algorithme suivant retourne une médiane.

Function MEDIANE($A[1..m], B[1..m]$)

```
Require: A, B : sorted arrays of size m
if m = 1 then
    if A[1] ≤ B[1] then return A[1] else return B[1]
else
    x ← [(m + 1)/2]
    if A[x] ≤ B[x] then
        return MEDIANE(A[(x + 1)..m],B[1..[m/2]])
    else
        return MEDIANE(A[1..[m/2]]),B[(x + 1)..m]
```

On considère les deux tableaux :

$A =$ | 7 | 28 | 46 | 53 | 64 | 84 | 89 | et $B =$ | 3 | 12 | 14 | 21 | 45 | 78 | 122 |

A) Premier algorithme (3 points)

1. Faîtes tourner l'algorithme avec l'appel MEDIANE($A[1..7]$,$B[1..7]$) et vérifiez que l'élément retourné est bien une médiane.

Solution: (1 point : déroulé 0.5, vérif 0.5)

x	A	B	m
4	7 28 46	45 78 122	3
2	46	45	1

renvoie 45. Il y a bien 7 éléments inférieurs ou égaux à 45 et 7 éléments supérieurs ou égaux à 45.

On note $S_{\text{MEDIANE}}(n)$ et $T_{\text{MEDIANE}}(n)$ les complexités respectivement spatiales et temporelles en pire cas de l'algorithme MEDIANE appliqué à des données de taille totale n.

2. En supposant que les appels à MEDIANE sur les sous-tableaux ne recopient pas les tableaux mais utilisent simplement des décalages des indices de début et de fin de ces tableaux, donnez l'expression de la complexité spatiale en pire cas de ce programme en notation asymptotique Θ, c'est-à-dire proposez une fonction g telle que $S_{\text{MEDIANE}} \in \Theta(g)$.

Solution: (1 point : $\Theta(1)$ ou $\Theta(logn)$) Variables utilisées : x (les variables d'indices doivent être fournies à la fonction), complexité en pire cas est donc en $\Theta(1)\times$ nb appels (nombre d'appels= $\log_2(n/2)$).

3. Donnez l'expression de la complexité temporelle en pire cas de ce programme en notation asymptotique Θ, c'est-à-dire proposez une fonction g' telle que $T_{\text{MEDIANE}} \in \Theta(g')$.

Solution: (1 point : 0.5 résultat + 0.5 justif (mention du Master Theorem), si justif = taille divisée par 2 donc log n j'ai mis 0.5 en tout, si résultat faux mais MasterTheorem 0.5 en tout) L'algorithme fait un test (puis soit (un test puis un retour) soit (une affectation puis se relance sur des sous-tableaux de tailles $n/2$) donc $T(n) = T(n/2) + constante$. D'après le Master-Theorem, cas 2), $f(n) = \Theta(1) = \Theta(n^{log_2 1})$ donc $T(n) \in \Theta(\log n)$.

B) Médiane avec un tas binaire (7 points)

Sachant que le m-ième élément par ordre croissant d'une liste de $2m$ éléments en est une médiane, on propose l'algorithme suivant :

Function MEDIANETAS($A[1..m], B[1..m]$)

Require: A, B : `arrays of size` m
  for $i = 1$ *to* m **do**
    $C[i] = A[i]$
    $C[i+m] = B[i]$
  BUILDHEAP(C)
  for $i = 1$ *to* m **do** $r \leftarrow$ REMOVE(C)
  return r

où BUILDHEAP(T) est la fonction qui construit un tas binaire minimal en-place à partir d'un tableau T et REMOVE(T) est une fonction qui supprime et renvoie la racine du tas T.

1. On lance l'algorithme avec l'appel MEDIANETAS(A[1..7],B[1..7]). Décrivez l'évolution du tableau C après le BUILDHEAP et après chacun des deux premiers REMOVE.

Solution: (4 points : BuildHeap 2 (si tas max au lieu de min :-1, si percolate up depuis les feuilles :-1), Remove 1+1) Si erreur et pas de justif : 0, si juste sans justif tous les points

C :	7	28	46	53	64	84	89	3	12	14	21	45	78	122
après BUILDHEAP	3	7	45	12	14	46	89	53	28	64	21	84	78	122
après REMOVE	7	12	45	28	14	46	89	53	122	64	21	84	78	
après REMOVE	12	14	45	28	21	46	89	53	122	64	78	84		

2. En vous servant des complexités des fonctions de traitement des tas binaires vues en cours, exprimez les complexités spatiale $S_{\text{MEDIANETAS}}(n)$ et temporelle $T_{\text{MEDIANETAS}}(n)$ en pire des cas de cet algorithme en fonction de la taille n des données, vous donnerez donc deux fonctions h et h' telles que $S_{\text{MEDIANETAS}} \in \Theta(h)$ et $T_{\text{MEDIANETAS}} \in \Theta(h')$.

Solution: (3 points : temporelle 2points = 0.5 bouclefor+ 0.5 Buildheap+0.5 Remove $\times$ m + 0.5 : aggrégation simplif, spatiale 1 point=0.5 result+0.5 justif)

— Complexité temporelle : BUILDHEAP est en $\Theta(n)$, REMOVE en $\Theta(\log n)$
donc $T_{\text{MEDIANETAS}}(n) = c.\frac{n}{2} + \Theta(n) + \frac{n}{2}\Theta(\log n) \in \Theta(n \log n)$.
— Complexité spatiale : BUILDHEAP et REMOVE sont en $\Theta(1)$
mais on recopie A et B dans un tableau C donc $S_{\text{MEDIANETAS}}(n) \in \Theta(n)$

C) Médiane avec un tas binomial (7 points)

On décide maintenant d'utiliser un tas binomial pour profiter de la fonction qui fusionne deux tas binomiaux. Voici l'algorithme utilisé.

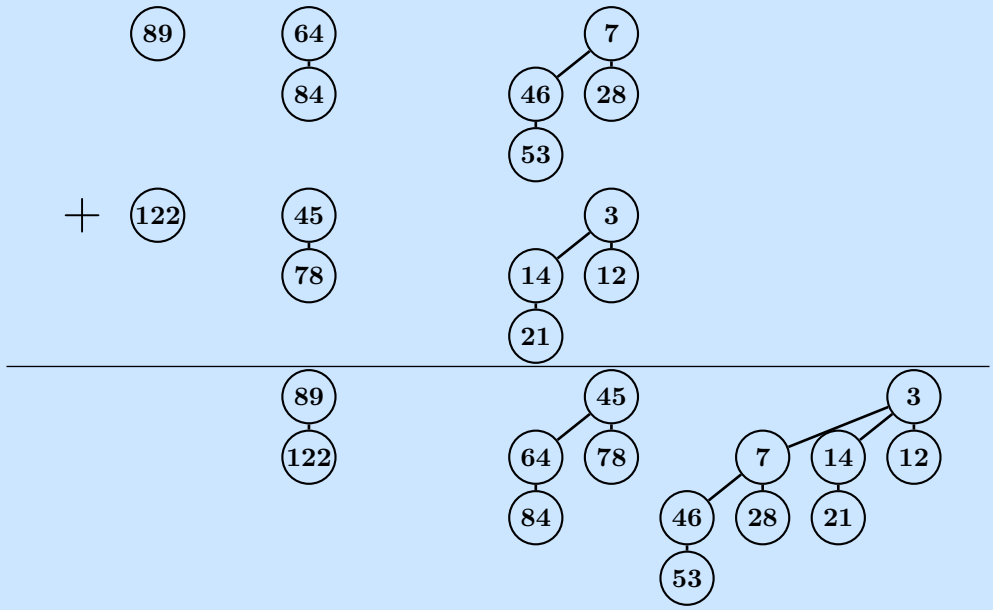
Function MEDIANETASBINOM($A[1..m], B[1..m]$)

Require: A, B : `arrays of size` m
- $H_A \leftarrow$ CREATEBINOMIALHEAP()
- $H_B \leftarrow$ CREATEBINOMIALHEAP()
- **for** $i = 1$ *to* m **do**
  - ADD(H_A,$A[i]$)
  - ADD(H_B,$B[i]$)
- $H \leftarrow$ MERGE(H_A,H_B)
- **for** $i = 1$ *to* m **do** $r \leftarrow$ REMOVEMIN(H)
- **return** r

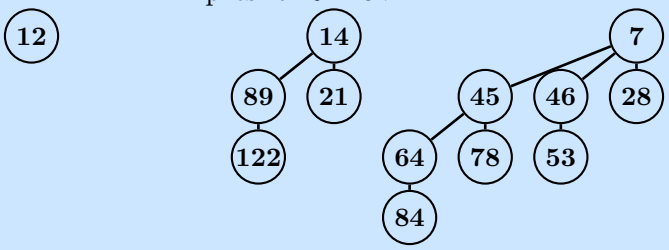
où CREATEBINOMIALHEAP() est la fonction qui crée un tas binomial vide, ADD(H,e) ajoute l'élément e au tas H, MERGE(H_1, H_2) retourne un tas résultant de la fusion de deux tas, REMOVEMIN(H) supprime et retourne l'élément minimum du tas H.

1. Faîtes tourner l'algorithme avec l'appel MEDIANETASBINOM($A[1..7]$,$B[1..7]$) afin de décrire les tas binomiaux H_A et H_B obtenus après la boucle **for** et le tas binomial H issu de MERGE, puis le tas H résultant du premier REMOVEMIN.

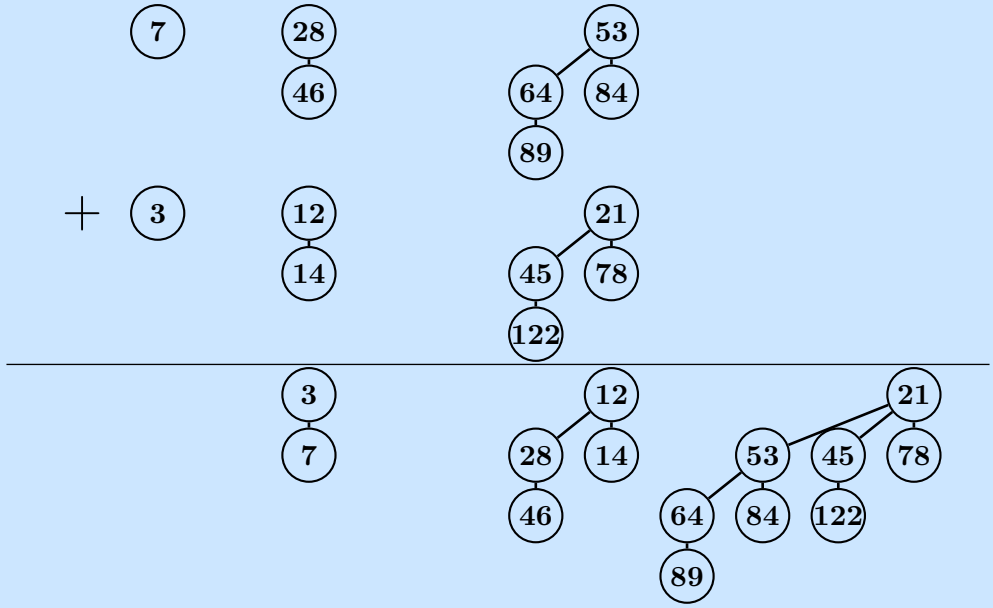
Solution: (5 points : H_A et H_B 2 points (0.5 structure (correcte si fils dans le bon ordre sinon -0.5), 0.5 tas minimal, 1 contenu résulte des m fois Add) ; merge 2 point (0.5 si saute bien première retenue + 3x0.5 par arbre correct dans tas final ; remove 1 point)



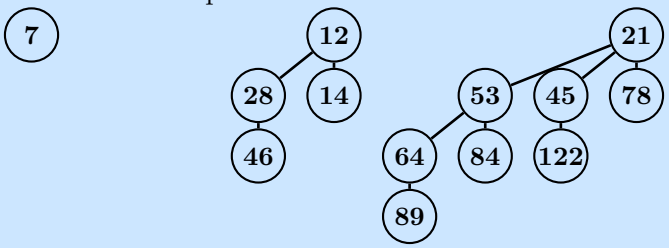
Après le REMOVEMIN :



Cas particulier : ceux qui ont fait la structure sans tenir compte de l'algo (donc sans faire m add par Tas dans l'ordre des tableaux A et B) Barème : **1** pour H_A et H_B, **2** pour Merge, seulement **0.5** pour RemoveMin (car il devient plus facile).



Après le RemoveMin :



2. En vous servant du fait que la complexité temporelle de CreateBinomialHeap est en $\Theta(1)$ et des complexités des fonctions de traitement des tas binomiaux vues en cours, exprimez la complexité temporelle $T_{\text{MedianeTasBinom}}(n)$ en pire des cas de cet algorithme en fonction de la taille n des données, vous donnerez donc une fonction h telle que $T_{\text{MedianeTasBinom}} \in \Theta(h)$.

Solution: (1 point (0.5 resultat+ 0.5 justif) Toutes les fonctions de traitement des tas binomiaux sont en pire des cas en $\Theta(\log n)$ donc
$T_{\text{MedianeTasBinom}}(n) = \Theta(1)+n\Theta(\log n)+\Theta(\log n)+\frac{n}{2}\Theta(\log n)$ pour les $n/2\times 2$ Add, le Merge et les $n/2$ RemoveMin. Donc $T_{\text{MedianeTasBinom}}(n) \in \Theta(n \log n)$.

3. Comparez les complexités temporelles des trois méthodes en tenant compte de la complexité du tri des deux tableaux initiaux nécessaire pour effectuer la première méthode.

Solution: (1 point (0.5 resultat+ 0.5 justif) A) necessite tri des tas de tailles $n/2$ chacun en $\Theta(n \log n)$ puis Mediane en $\Theta(logn)$ donc en $\Theta(n \log n)$ comme B et C. Ces trois programmes ont des complexités temporelles théoriques en pire cas équivalentes.

II - B-arbre des 100 premiers entiers (3 points)

On crée un B-arbre de degré 4 en insérant les 100 premiers entiers dans l'ordre croissant.

1. Donnez la hauteur de cet arbre en justifiant votre réponse.

Solution: (2 points, 1 point si formule $h \leq \log_4(101/2)$) nbnoeuds=100 compris entre min $2.4^h - 1$ et max $8^{h+1} - 1$ or $4^3 = 64$ ce qui donnerait trop de clés : 127, il faut donc une hauteur de 2 seulement qui permettra de stocker entre 31 et jusqu'à $8 \times 8 \times 8 - 1 = 511$ clés.

2. Donnez la première clé de sa racine sans justifier votre réponse.

Solution: (1 point, 17=0.5, 15=1, reste=0) 16