



uOttawa

CSI 2510 - Structures de données et algorithmes

AUTOMNE 2021 SECTION A

PROFESSEURE DORRA RIAHI

Laboratoire 4

Kien Do (ID: 300163370)

Question 1 (4 points)

Soit l'arbre monceau min montré ci-dessous sous forme d'un tableau. Effectuer les opérations suivantes. Dessiner le tableau et l'arbre correspondant après chacune des opérations.

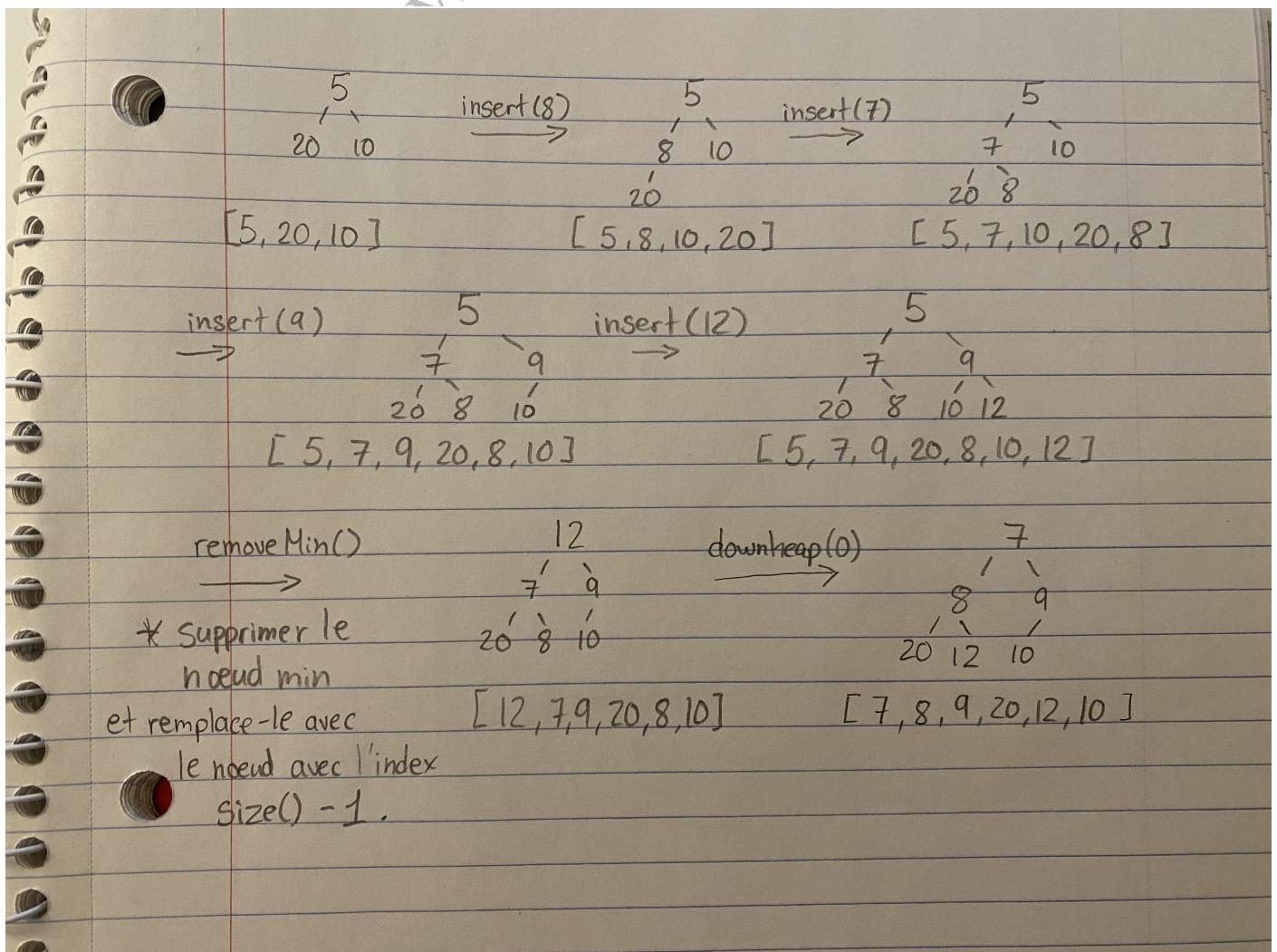
1.

5	20	10	-	-	-	-
---	----	----	---	---	---	---



```

insert(8);
insert(7);
insert(9);
insert(12);
removeMin();
  
```



Question 2 (4 points)

Soit l'arbre monceau min montré ci-dessous sous forme d'un tableau. Transformer ce monceau min en monceau max en utilisant l'algorithme de construction ascendante $O(N)$ vu en classe. Bien montrer l'état du tableau après chaque itération (i.e. après chaque traitement de sous-arbre).

2.

```
for (i=(n-2)/2; i>=0; i--)
    downheap(A, i); /* downheap for a maxheap */
```

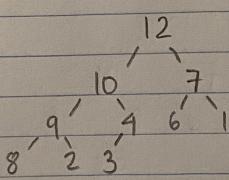
0	1	2	3	4	5	6	7	8	9
1	2	6	8	3	12	7	10	9	4

La taille n est 10. On suit le boucle for et commence à $\frac{n-2}{2} = \frac{10-2}{2} = 4$. Alors, on commence le noeud à l'index 4 puis on continue le processus "i--" du boucle.

Remarque que pour le downheap(i), on échange i avec compare $(2i+1, 2i+2)$, met le nouveau index à compare $(2i+1, 2i+2)$, puis on continue tant que hasLeft(i) est true.

l'index →	0	1	2	3	4	5	6	7	8	9
l'original →	1	2	6	8	3	12	7	10	9	4
($i=4$)	1	2	6	8	<u>4</u>	12	7	10	9	<u>3</u>
($i=3$)	1	2	6	<u>10</u>	<u>9</u>	12	7	<u>8</u>	<u>9</u>	<u>3</u>
($i=2$)	1	2	<u>12</u>	<u>10</u>	<u>4</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>3</u>
($i=1$)	1	<u>10</u>	<u>12</u>	<u>2</u>	<u>4</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>3</u>
	1	<u>10</u>	<u>12</u>	<u>9</u>	<u>4</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>2</u>	<u>3</u>
($i=0$)	<u>12</u>	<u>10</u>	<u>1</u>	<u>9</u>	<u>4</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>2</u>	<u>3</u>
final →	<u>12</u>	<u>10</u>	<u>7</u>	<u>9</u>	<u>4</u>	<u>6</u>	<u>1</u>	<u>8</u>	<u>2</u>	<u>3</u>

Donc, l'arbre monceau max et le tableau final est


[12, 10, 7, 9, 4, 6, 1, 8, 2, 3]

Question 3. (4 points)

Soit un tableau A de dimension n et un entier k dans l'intervalle $\{1, \dots, n\}$. Vous devez écrire un algorithme aussi efficace que possible permettant d'obtenir le kième plus petit élément du tableau.

Par exemple:

pour $A=[9,6,1,100,3,8,5,80]$ et $k=3$, la réponse sera 5.

L'algorithme ci-dessous permet de résoudre ce problème avec une efficacité de $O(n \log n)$ ce qui n'est pas très bon lorsque $k \ll n$.

3.

```
kSmallest(int A[], int n, int k) {
    // assume k has already been tested to be in {1, ..., n}
    heapSort(A); /* takes O(n log n) */
    return A[k-1];
}
```

Proposer un algorithme plus efficace (sous forme de code Java ou de pseudo-code) et donner son efficacité Grand O. Utiliser le TAD monceau-min et/ou monceau max pour votre solution.

Indice : Afin d'obtenir tous vos points, votre algorithme devrait avoir une complexité de $O(n + k \log n)$ alors qu'une solution $O(n \log k)$ vaudra 3 points.

```
int kSmallest (int A[], int n, int k) {
    // assume k is between 1 and n, inclusive

    // build a min heap from the given array A
    Heap minHeap = new Heap(); // create empty min heap
    minHeap.buildMinHeap(A); // build min heap -> O(n)

    // get the kth smallest element
    int kthSmallestNum = 0;
    // call removeMin() k times -> O(k)
    for (int i = 0; i < k; i++) {
        kthSmallestNum = minHeap.removeMin(); // recall removeMin() is O(log n)
    }

    // This algorithm is O(n + k log(n))

    return kthSmallestNum;
}
```