FIT 2 [INF224] – 2021/22

Algorithmique et structures de données
TP 05

Gulfem Isiklar Alptekin – Ozgun Pinarer

Partie 1 : La file d'attente prioritaire

Elle est similaire à la file d'attente dans certains aspects et pourtant elle diffère de la file d'attente ordinaire sur les points suivants:

- Chaque élément de la file d'attente prioritaire est associé à une priorité.
- L'élément avec la priorité la plus élevée est le premier élément à être supprimé de la file d'attente.
- Si plusieurs éléments ont la même priorité, leur ordre dans la file d'attente est pris en compte

Question 1

Définissez la structure pour la file dattente prioritaire

Question 2

En utilisant la structure que vous avez définie, écrivez les fonctions :

- *insert(item, priority)*: Insère un élément dans la file d'attente prioritaire avec une priorité donnée.
- *getHighestPriority()*: Renvoie un élément avec la priorité la plus élevée.
- *deleteHighestPriority()*: Supprime un élément avec la priorité la plus élevée.
- *remove(it)*: supprime un élément pointé par un itérateur it
- *changePriority(it, p)*: change la priorité d'un élément pointé par it à p

En dehors des opérations ci-dessus, nous pouvons également utiliser les opérations de file d'attente normales comme isEmpty(), isFull() et display().

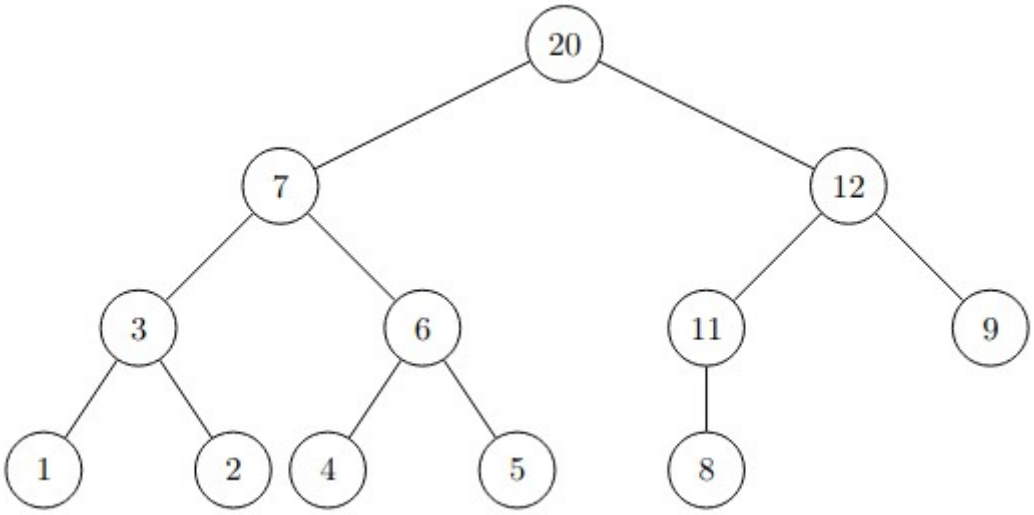
Partie 2 : Le Tas Binaire

Un tas binaire est une structure de données permettant de modéliser une file à priorités.
Cette structure de données maintient un ordre partiel sur les données via un arbre binaire. Cet arbre respecte les propriétés suivantes :

- chaque nœud a une priorité supérieure à celles de ses enfants ;
- un nouveau niveau n’est créé que si le niveau précédent est plein ;
- les niveaux de l’arbre sont remplis de gauche à droite.

Cette structure de données peut être implémentée efficacement dans un tableau : la racine est la première case du tableau, les deux cases suivantes contiennent le premier niveau, les quatre suivantes le second niveau, et ainsi de suite.

La racine de l'arbre a par construction la plus forte priorité, et est stockée dans la première case du tableau. Il est donc possible d'y accéder en temps constant. Pour les autres opérations, l'arbre étant équilibré, toute action qui nécessite de le parcourir de haut en bas ou de base en haut aura donc au pire une complexité en O(log n), la hauteur de l'arbre en fonction du nombre d'éléments.



20	7	12	3	6	11	9	1	2	4	5	8

Question 3
Pour simplifier le problème, votre tas binaire contiendra des entiers, qui correspondent aux priorités des éléments. Le sommet du tas (la première case du tableau) est donc l'entier le plus grand. Vous coderez les fonctionnalités de base du tas binaire :

- void init_tas(tas* t) ;
- void detruire_tas(tas* t) ;
- void inserer_tas(tas* t, int valeur) ;
- int consulter_tas(const tas* t) ;
- void supprimer_tas(tas* t) ;

Question 4
Insérez successivement les valeurs 7, 1, 9, 35, 4, 10, 13, 2, 20, 40 dans un tas, et indiquez à chaque insertion les états par lesquels passe le tas.

Question 5
Supprimez un à un les éléments du tas, et indiquez à chaque itération l’état du tas.

IMPORTANT:

- Vous devez effectuer des études TP en tant que « fichier d'en-tête – fichier source – fichier de test » : écrivez les en-têtes des fonctions que vous avez écrites dans le fichier d'en-tête, les codes des fonctions dans le fichier source et les codes de test dans le fichier d'essai.
- Nous préférons faire des études de TP dans un environnement linux, sans avoir besoin d'IDE supplémentaire.
- Compressez vos fichiers et téléchargez-le sur le système en les nommant « NumeroEtudiant_Prenom_Nom_TPX.zip ».