Bracquier Benjamin Cateau Pierrick Soler Lilian

TP 1 Algorithmique Avancée

Introduction

Pour ce TP,nous avons programmé en langage C des fonctions sur les piles,files,ABR et AVL.Il existe 4 fichiers de test pour chaque type.

2 Piles

La pile utilise le système *LIFO (Last in First Out)* qui signifie que le dernier élément entré à l'empilement dans la pile est le premier à en sortir au dépilement.

Nous avons implémenter des fonctions sur des piles. Le fichier pile.h contient les descriptions nécessaires pour le type pile_t. Les piles contiennent des pointeurs sur des nœuds de l'arbre. Les piles seront utilisées pour des traitements sur les arbres ABR qui ne seront pas récursives.

1)creer_pile:

Pour créer cette pile, nous allouons de la place en mémoire,puis nous mettons le sommet de la pile à 0 et nous retournons la pile.

2)detruire_pile:

Pour détruire une pile,nous vérifions tout d'abord que la pile n'est pas vide puis si elle ne l'est pas ,nous libérons cette pile.

3)pile_vide:

Nous retournons le sommet de la pile s' il est différent de 0.

4)pile_pleine:

Nous retournons le sommet si il est égal au maximum de la taille de la pile possible.

5)depiler:

Tout d'abord si la pile est vide,on n'a pas besoin de dépiler autrement on crée une autre pile qui prend dans la pile le sommet de l'ancienne pile puis on baisse le sommet de 1 et on retourne la nouvelle pile.

6)empiler:

Pour empiler,nous avons besoin tout d'abord que la pile ne soit pas pleine puis si elle ne l'est pas,nous augmentons le sommet de 1 et nous affectons le noeud du paramètre là où on veut .

3 Files

La file utilise le système *FIFO (First in First Out)* qui signifie que le premier élément entré à l'ajout dans la file est le premier à en sortir au défilement.

Pour gérer les éléments de la pile et vérifier que la taille de celle-ci ne dépasse pas le maximum autorisé (ici 32 éléments), on utilise deux champs tête et queue afin d'avoir le point de départ de la file et son dernier élément. Tant que l'écart entre les deux est inférieur à 32, on peut enfiler. De même, on peut défiler si tête != queue.

Nous implémentons des fonctions sur des files. Le fichier `file.h` contient les descriptions nécessaires pour le type `file_t`. Les files contiendront des pointeurs sur des nœuds de l'arbre. Les piles seront utilisées pour des parcours en largeur.

7)creer_file:
Pour créer cette fille,nous allouons de la place en mémoire et nous mettons la tête et la queue à 0 puis nous renvoyons la file.
8)detruire_file:
Pour détruire une file,nous vérifions tout d'abord que la file n'est pas vide puis si elle ne l'est pas ,nous libérons cette pile.
9)file_vide:
La file est vide si la queue est égale à la tête et que la case de la tête est null.
10)file_pleine:
La file est pleine si la différence entre la queue et la tête est égale à 1 ou que la différence entre la tête et la queue est égale à la taille maximum de la file.
11)defiler:
Tout d'abord si la file est vide,on n'a pas besoin de dépiler autrement on crée une autre file qui prend dans la file la tête de l'ancienne file moins 1 puis on affecte à la tête de la file NULL et on retourne la nouvelle file.
12)enfiler:
Pour enfiler,nous avons besoin tout d'abord que la file ne soit pas pleine puis si elle ne l'est pas,nous regardons si la queue égale à 0 et on affecte à la queue la taille maximale de la file moins 1 sinon on enlève 1 de la queue puis on affecte à la case de la queu le noeud p.

4 Arbres ABR

Nous avons utilisé la structure ABR présentée en cours et définie dans le fichier abr.h.
15)parcourir_arbre_largeur:
Nous parcourons l'arbre en largeur à l'aide d'une file. Nous commençons par enfiler le nœud de l'arbre passé en paramètre. Puis tant que la file n'est pas vide, nous défilons l'élément dans la pile puis si ses fils existent, nous enfilons les fils de cet élément.
16)hauteur_arbre_r:
Nous calculons la hauteur de l'arbre récursivement. Nous calculons pour cela le maximum entre la hauteur du fils gauche de l'arbre et la hauteur du fils droit de l'arbre. Les conditions d'arrêt possibles de l'algorithme sont : le nœud est null.
hauteur_arbre_nr:
Nous calculons la hauteur de l'arbre à l'aide d'une file. Nous commençons par empiler le nœud de l'arbre passé en paramètre. Puis tant que la pile n'est pas vide ,nous dépilons la pile puis si un fils n'est pas nul,nous empilons le fils correspondant et nous empilons la deuxième pile avec h+1.Après cela,on retourne la hauteur.
17)nombre_cle_arbre_r:

Nous calculons le nombre de clés dans l'arbre récursivement.Ainsi si l'arbre n'est pas nul,on retourne la fonction avec le fils droit plus le le fils gauche plus un.

nombre_cle_arbre_nr:

Nous calculons le nombre de clés dans l'arbre à l'aide d'une file. Nous commençons par créer la file puis si l'arbre n'est pas nul,on enfile la racine de l'arbre passé en paramètre.Tant que la file n'est pas vide,on défile ,on incrémente le nombres de nodes et si un fils n'est pas nul,on enfile ce fils.Et enfin on retourne le nombre de nodes.

18)imprimer_liste_cle_triee_r:

Nous affichons la liste des clés triées dans l'arbre récursivement. Nous appelons pour cela cette fonction sur le fils gauche, puis nous affichons la clé de la racine actuelle puis nous appelons récursivement cette fonction récursive sur le fils droit. Par conséquent, avant d'afficher la clé de l'arbre courant, la fonction affichera d'abord son fils gauche et les propres fils de ce dernier. La condition d'arrêt de l'algorithme est : l'arbre est null.

19)arbre_plein:

Les conditions d'arrêt sont que l'arbre soit nul ou que les 2 fils soient null.
Si ces conditions ne sont pas respectées,on retourne la fonction avec le fils gauche et le fils droit.

20)arbre_parfait:

Si l'arbre est un arbre plein, alors l'arbre est parfait. Si l'arbre ayant n nœuds a pour hauteur , alors l'arbre est parfait. Sinon, il n'est pas parfait ce qui est représenté par les conditions de l'algorithmes.

21)rechercher_cle_sup_arbre:

On cherche le nœud ayant une clé directement supérieure à celle donnée. Il y a donc plusieurs cas à vérifier : Tout d'abord, si la clé donnée n'appartient pas à l'arbre, on retourne null. Ensuite, on parcourt récursivement les fils droits si la cle est supérieure à la valeur. Si la cle est supérieure à la valeur etq ue a_sup est null alors on retourne l'arbre sinon on retourne a_sup. Autrement on retourne NULL.

22)rechercher_cle_inf_arbre:

Même chose que le sup sauf qu'on parcourt les fils gauches à la place des fils droits.

23)intersection_deux_arbres:

Tout d'abord,si 1 des 2 arbres est nul ,on retourne NULL.Sinon on crée un autre arbre qui est null puis si la cle des 2 arbres sont égaux on crée un arbre avec comme cle celle de a1 et pour le fils gauche ,on appelle la fonction avec les fils gauches des 2 arbres et pareil pour le fils droit.Si la cle de a1 est inférieur de a2 alors on appelle la fonction pour le fils droit de a1 et si c'est supérieur ,on appelle avec le fil droit de a2.
Et enfin,on retourne le nouvel arbre.

24)union_deux_arbres:

Tout d'abord,si 1 des 2 arbres est nul ,on retourne cet arbre.Sinon on crée un autre arbre qui est null puis si la cle des 2 arbres sont égaux on crée un arbre avec comme cle celle de a1 et pour le fils gauche ,on appelle la fonction avec les fils gauches des 2 arbres et pareil pour le fils droit.Si la cle de a1 est inférieur à celle de a2 alors on crée un arbre avec comme cle de a1 et le fils gauche de cet arbre devient le fils gauche de a1 et le fils droit est affecté par la fonction avec comme paramètre le fils droit de a1 et a2.La même chose pour le cas inverse puis on retourne le nouvel arbre.

26)detruire_cle_arbre:

On cherche à supprimer une clé donnée.Si l'arbre est null alors on retourne null.Si la cle de a est égale à la cle alors si il n'y a pas de fils on libére l'arbre et on retourne null .Si il n'y a qu'une des feuilles qui est null alors on crée un nouvel arbre qui est le fils qui existe puis on libére l'arbre et on retourne le nouvel arbre.Sinon on recherche la cle supérieur dans le fils droit et la cle de l'arbre devient le résultat de la focntion précédente puis on appele récursivement la fonction avec comme arbre,le fils droit et comme cle celle qu'on veint de voir.