Leçon 901

Coudert Patrice

I] Introduction et structure de base

1) Opération désirée

find : trouver un élément de notre structure

create : créer la structure.

union : unifier deux structures

insert : ajouter un element de notre structure.

delete : enlever un element de notre structure

find-min : trouver le plus petit selon un ordre donnée

find-earliest : trouver le dernier element ajouter

2) liste

Def 1 : Une liste simplement chainée est un structure de données où on a accés à la dernière donnée inserée qui a un pointeur vers la donnée suivante et ainsi de suite.

Proposition 2 : temps des différentes opérations

n représentera la taille de la liste

find : $O(n)$ — insert : $O(1)$

create : $O(n)$ — delete : $O(n)$

union : $O(\min(n_1, n_2))$ — find-min : $O(n)$

find-earliest : $O(1)$.

3) tableau de taille fixée

Définition 3

On considère ici qu'on tableau a une taille fixée et que donc ajouter (ou enlever) un élément oblige à recréer le tableau.

Proposition 4 Coût des opérations

n represente la taille du tableau

find : $O(1)$ — insert : $O(n)$

create : $O(n)$ — delete : $O(n)$

union : $O(n_1 + n_2)$ — find-min : $O(n)$

find-earliest : impossible

4) Noms des types de structure de données

Définition 5. On donne les noms suivants au problèmes de trouver une structure de donnée efficace pour les opérations listées.

Pile : insert, find-earliest, delete-earliest

File : insert, find-oldest, delete-oldest

File de priorité : insert, find-min, delete-min

Dictionnaire : insert, delete, find

Union-find : union, find, find-structure

Proposition 6 : Une liste doublement chaînée remplit parfaitement les problèmes Pile et File.

Tout les operations sont en $O(1)$.

Constant Fabrice
Leçon 910

II Une file de priorité : tas binaire

DEV1

Définition 7 : Arbre binaire équilibré
Un arbre binaire est un arbre dont chaque noeud a **exactement** deux fils. Ces fils peuvent être vide à la hauteur maximale et seulement les derniers.

Définition 8 : tas binaire
On appelle descendant d'un noeud l'ensemble de ses fils et des descendant de ceux-ci.
Un tas binaire est un arbre binaire dont les descendant d'un noeud sont tous supérieur ou égale à ce noeud.

Algorithme 9 Insertion dans un tas binaire
Entrée : arbre A, élément e
Sortie : A tas binaire avec e dedans
Insérer e à la place disponible
père = A.père[e]
Boucle(père, e)
  si père > e alors terminer
  sinon echanger e et son père
    et recommencer avec A.père[père] etc

Proposition 10 Coût du tas binaire
n est le nombre d'élément
Insert : $O(\log n)$ find_min : $O(1)$
Create : $O(n)$ delete_min : $O(\log n)$

Proposition 11 Principale file de priorité

	Insert	Find_min	Delete_min
liste	$O(1)$	$O(n)$	$O(1)$
liste triée	$O(n)$	$O(1)$	$O(1)$
tas binaire	$O(\log n)$	$O(1)$	$O(\log n)$
tas binomial	$O(\log n)$	$O(1)$	$O(\log n)$
tas de fibonacci	$O(1)$	$O(1)$	$O(\log n)$ amorti

Application 12 tri par tas binaire
Boucle tanque non vide
  find_min, delete_min, renvoyer min

III Dictionnaire : arbre binaire de recherche

Définition 13 : Arbre binaire de recherche
Un arbre binaire de recherche est un arbre dont le sous arbre droit d'un noeud lui est inférieur et le sous arbre gauche lui est supérieur

Proposition 14

Coût des opérations : h est la hauteur de l'arbre
insert $O(h)$
find $O(h)$
Delete $O(h)$
Find-min $O(h)$

DEUX

Définition 15 : Arbre Zig-Zag
Un arbre Zig-Zag est un arbre binaire de recherche équilibré

Problème 16 Stockage d'une base de données

Le problème est de stocker un ensemble de données et de pouvoir gérer cet ensemble efficacement.

IV] Union-find

Définition 17 On a ici un ensemble de structure et on veut à la fois pouvoir savoir si un element apartient à une structure et pouvoir unifier deux structure

Problème 18 Arbre couvrant de poid minimal

Etant donné un graphe non orienté dont les arrêtes ont un poid, on cherche un arbre couvrant de poids minimal

Algorithme 19 Kruskal

trier les arrêtes par ordre de poids

Boucle

Prendre l'arrête de poids min
Si elle ne crée pas de cycle
rajouter cette arrête à la structure

Proposition 20 cet algorithme s'implemente en $O(n \log n)$ avec une structure d'union find..

V] Conclusion

Créer la structure de données en fonction des problèmes

Candan Patrice
Leçon 902