Algorithmique et Structures de Données 2

Rapport Projet n°1:

Ensemble

Mamadou DIALLO
Fatma MAOULOUD
Groupe 403 et 419

Introduction

Ce projet consiste à comparer trois différentes implémentations d'ensembles disposant des méthodes décrites dans le sujet du premier TP1 auxquelles s'ajoutent les méthodes d'intersection, d'union et de différence d'ensembles. Le premier ensemble est implémenté à l'aide d'un chaînage simple sans doublons , le deuxième ensemble à l'aide d'un tableau statique trié et le dernier ensemble à l'aide d'un chaînage simple avec doublons respectivement appelé Ensa, Ensb et Ensc.
Ce rapport présentera dans une première partie les ambiguïtés du sujet ,les choix effectués et les limites d'utilisation , ensuite nous parlerons des problèmes rencontrés et des idées d'amélioration et pour finir on parlera du coûts temporels des méthodes de nos différentes classes.

I-Choix effectués et Limites d'utilisation

a)Choix effectués

Pour ce qui est des choix effectués , les trois classes étant toutes différentes alors certains choix sont spécifique en fonction de la classe.

-Pour la première classe celle implémentée à l'aide d'un chaînage simple sans doublons non trié et dont l'ajout ce fait en tête , étant la même que celle du tp1 alors les choix sont plutôt sur les trois méthodes rajoutées (intersectionEns, unionEns , differenceEns) , déjà c'est de dire que c'est trois méthodes sont des procédures dans lesquelles ont modifie l'objet courant , nous avons principalement fais ce choix à cause de sa simplicité.

-Pour la seconde classe celle implémentée à l'aide du tableau statique trié , le premier choix c'est porté sur la taille (capacité) de notre tableau et notre choix c'est porté sur un nombre assez modeste qui est 40000. pour ce qui est des méthodes (intersectionEns, unionEns , differenceEns) les choix ont été les mêmes que celles de la classe précédentes et pour le reste des méthodes seule l'ajout et la suppression se comportent différemment , pour l'ajout ont fais une comparaisons deux à deux des éléments de l'ensemble jusqu'à trouver l'emplacement parfait pour l'élément à rajouter ensuite un décalage est fais et pour la suppression c'est un peu du même ordre ont cherche l'élément à supprimer et on fais décalage ce qui permet d'ôter l'élément.

-Pour notre dernière classe celle implémentée avec un chaînage simple avec doublons et non trié et où l'ajout se fait en tête , elle s'écrit de la même manière que la première classe à quelques méthodes près , les seules méthodes qui diffèrent sont « ajoute » et « retire » , pour la méthode ajoute vu que l'ajout se fait en tête alors son écriture est simple et se compte en temps constant $\theta(1)$ ce qui la rend très efficace, et pour la suppression étant donné l'existence des doublons alors la suppression d'un élément ce fait de t-elle sorte qu'il n y ait plus aucune occurrence de cet élément dans l'ensemble.

b)Limites d'utilisation

Pour les limites d'utilisation il faut souligner qu'il y en a pas beaucoup parmi celle qu'on peut citer c'est de dire que le type T définie dans notre programme fais référence aux type basique (entier , chaîne de caractère...) et en aucun cas ne peux être un objet , on pourrais éventuellement le faire mais cela nécessiterais d'autres modifications et l'ajout d'autres méthodes.Comme autre limite nous

pouvons citer le fait que notre tableau qui sert à implémenter le deuxième ensemble « Ensb » à une capacité de 40000 élément on ne peut aller au-delà de ce nombre sauf si l'utilisateur pense à changer ce nombre !
Et pour finir notre dernière limite concerne nos deux dernières méthodes (intersectionEns, differenceEns) , ici vu que c'est deux méthodes sont des procédures où l'objet courant est modifié , dans ces deux méthodes la comparaison entre les éléments de l'ensemble courant et les éléments de l'ensemble pris en paramètre se fait par rapport aux éléments de l'ensemble courant.

II-Problèmes rencontrés et solutions

Pour ce qui est des problèmes rencontrés déjà pour la classe Ensa vu que la plupart des méthodes étaient ceux du TP1 alors il y a pas eu de problèmes , les problèmes sont apparus avec les trois dernières méthodes (intersectionEns, unionEns , differenceEns) car dans nos premières implémentations on avait convenu de les faire comme des fonctions qui retournent un type ensemble et non comme des procédures ce qui nous posait d'énormes problèmes (des erreurs de segmentations) car on faisaient des affectations entre des ensembles sans utiliser un opérateur d'affectation , aussi on avait pas une fonction pour cloner nos ensembles ce qui nous posaient d'énormes difficultés lors de la destruction de nos objet , et pour remédier à ces problèmes on a préféré procéder très simplement en utilisant des procédures au lieu des fonctions , et on avait qu'à modifier l'objet courant en fonction de nos besoins, voici par exemple la méthode union de notre Ensa et la méthode intersection de notre Ensb.

```
template < typename T>
void Ensa<T>::unionEns(const Ensa<T> & e){
  Maillon *tmp = e.tete;
  for(int i=0; i<e.nb ; i++){
    if(!this->contient(tmp->ch)){
      this->ajoute(tmp->ch);
    }
    tmp = tmp->suiv;
  }
}
```

Ensa

```
template < typename T>
void Ensb<T>::intersectionEns(Ensb<T> & e){
  Ensb<T> ensRes;
  for(int i=0 ;i<this->nb ;i++){
    ensRes.ajoute(this->tab[i]);
  }
  this->nb = 0 ; //initialisation de l'objet courant.
  for(int j=0; j<ensRes.nb ; j++){
    if(e.contient(ensRes.tab[j])){
      this->ajoute(this->tab[j]);
    }
  }
}
```

Ensb

III- Ensembles et coût temporel.

Nos trois différentes Ensembles de ce projet disposent des même méthodes décrites dans le TP1 auxquelles se rajoutent trois autres méthodes qui sont « intersectionEns », « unionEns » et « differenceEns »

intersectionEns :

Cette méthode fait l'intersection entre l'ensemble courant et l'ensemble pris en paramètre , elle modifie l'ensemble courant de sorte à conserver les éléments qui sont à la fois dans les deux ensembles.

unionEns :

Cette méthode fait l'union entre l'ensemble courant et l'ensemble pris en paramètre , elle le fait en rajoutant les éléments du second ensemble dans l'ensemble courant.

differenceEns :

Cette méthode fait la différence entre l'ensemble courant et l'ensemble pris en paramètre , elle modifie l'ensemble courant de sorte à conserver que les éléments qui sont présent dans l'ensemble courant et pas dans l'ensemble pris en paramètre.

1- Premier Ensemble « Ensa »

- estVide :

Pire cas = **Meilleur cas** : que l'ensemble soit vide ou pas le temps mis par le programme pour exécuter cette fonction est de 2t , θ(1).

-contient :

Pire cas: se produit lorsque l'élément se trouve au dernier maillon du chaînage .

Meilleur cas: se produit lorsque l'élément rechercher se trouve en tête du chaînage.

-ajoute:

P**ire cas :** se produit lorsque on ajoute à la fin car on parcours tout le chaînage.

Meilleur cas : se produit lorsque l'ensemble est vide , l'ajout se fait en temps constant θ(1).

-retire :

Pire cas : se produit lorsque l'élément à retirer est à la fin du chaînage.

Meilleur cas : se produit lorsque l'élément à retirer est en tête.

-contenu:

Pire cas = **Meilleur cas**: Il n'y a pas de pire ou de meilleur cas pour cette fonction, le nombre de tour de la boucle 'while' augmente lorsque que le nombre d'éléments augmente.

-intersectionEns :

Pire cas : se produit lorsque tout les éléments de l'ensemble passé en paramètre figurent dans l'ensemble courant.

Meilleur cas : se produit lorsque tous les éléments de l'ensemble passé en paramètre sont différents de ceux présent dans l'ensemble courant.

-unionEns :

Pire cas : se produit lorsque aucun élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

Meilleur cas : se produit lorsque chaque élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

-differenceEns :

Pire cas : se produit lorsque aucun élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

Meilleur cas : se produit lorsque chaque élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

2- Deuxième Ensemble « Ensb »

-estVide :

Pire cas = Meilleur cas : que l'ensemble soit vide ou pas le temps mis par le programme pour exécuter cette fonction est de 2t ,θ(1) .

-contient:

Pire cas: se produit lorsque l'élément se trouve au dernier maillon du chaînage .

Meilleur cas: se produit lorsque l' élément rechercher se trouve à la dernière case du tableau (ou pas mais le nombre de tour serai toujours du nombre de cases occupées dans le tableau).

-ajoute:

Pire cas : se produit lorsque l'élément à ajouter est inférieur à tout les autres dans ce cas tous les éléments qui sont lui sont supérieure devrons être décalés pour faire de la place

Meilleur cas : se produit lorsque l'élément à ajouter est supérieur à tout les autres donc l'ajout sera à la fin du tableau et ne neccessite aucun decalage.

-retire :

Pire cas : se produit lorsque l'élément à retirer est à la première position du tableau dans ce cas on doit le supprimer en copiant tout les autres éléments dans le bon endroit.

Meilleur cas : se produit lorsque l'élément à retirer à la dernière case du tableau (on écrase son contenu en diminuant le nombre de case occupées dans le tableau.

-contenu:

Pire cas = Meilleur cas: Il n'y a pas de pire ou de meilleur cas pour cette fonction, le nombre de tour de la boucle 'while' augmente lorsque que le nombre d'éléments augmente.

-intersectionEns :

Pire cas : se produit lorsque tout les éléments de l'ensemble passé en paramètre figurent dans l'ensemble courant.

Meilleur cas : se produit lorsque tous les éléments de l'ensemble passé en paramètre sont différents de ceux présent dans l'ensemble courant.

-unionEns :

Pire cas : se produit lorsque aucun élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

Meilleur cas : se produit lorsque chaque élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

-differenceEns :

Pire cas : se produit lorsque aucun élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

Meilleur cas : se produit lorsque chaque élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

3- Troisième Ensemble « Ensc »

-estVide :

Pire cas = Meilleur cas : que l'ensemble soit vide ou pas le temps mis par le programme pour exécuter cette fonction est de 2t ,θ(1) .

-contient:

Pire cas: se produit lorsque l'élément se trouve au dernier maillon du chaînage .

Meilleur cas: se produit lorsque l' élément rechercher se trouve en tête du chaînage.

-ajoute:

Pire cas =Meilleur cas : que l'élément soit présent ou pas dans le chaînage on ajoute en tête donc le temps mis pour exécuter ce programme est de 11t ,θ(1) .

-retire :

Pire cas : se produit lorsque l'élément à retirer est à la fin du chaînage.

Meilleur cas : se produit lorsque l'élément à retirer est en tête.

-contenu:

Pire cas = Meilleur cas: Il n'y a pas de pire ou de meilleur cas pour cette fonction, le nombre de tour de la boucle 'while' augmente lorsque que le nombre d'éléments augmente.

-intersectionEns :

Pire cas : se produit lorsque tout les éléments de l'ensemble passé en paramètre figurent dans l'ensemble courant.

Meilleur cas : se produit lorsque tous les éléments de l'ensemble passé en paramètre sont différents de ceux présent dans l'ensemble courant.

-unionEns :

Pire cas : se produit lorsque aucun élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

Meilleur cas : se produit lorsque chaque élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

-differenceEns :

Pire cas : se produit lorsque aucun élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

Meilleur cas : se produit lorsque chaque élément de l'ensemble passé en paramètre existe dans l'ensemble courant.

IV - Tableau récapitulatif des ordres de grandeur des méthodes des classe « Ensa » « Ensb » et « Ensc » en fonction de l'implémentation utilisée :

Ensa = chaînage simple sans doublons non trié ajout en queue
Ensb = tableau trié sans doublons
Ensc = chaînage simple avec doublons non trié ajout en tête

	Ensa	Ensb	Ensc
Ensemble	θ(1)	θ(1)	θ(1)
~Ensemble	O(nb)	θ(1)	O(nb)
estVide	θ(1)	θ(1)	θ(1)
contient	O(nb)	O(nb)	O(nb)
ajoute	O(nb)	O(nb)	θ(1)
retire	O(nb)	O(nb)	$O(nb^2)$
contenu	O(nb)	O(nb)	O(nb)
intersectionEns	$O(nb^2)$	$O(nb^2)$	$O(nb^2)$
unionEns	$O(nb^2)$	$O(nb^2)$	$O(nb^2)$
differenceEns	$O(nb^2)$	$O(nb^2)$	$O(nb^2)$

V-Conclusion

En comparant les différentes implémentations à travers leurs ordres de grandeurs , nous remarquons que les trois se valent à quelques méthodes près , certes celle implémentée avec le chaînage avec doublons non trié « Ensc » est plus efficace lors de l'ajout mais pers si tôt son efficacité lors du retrait d'éléments .Ce qui nous amène à dire que quelques soit l'implémentation choisie l'efficacité est presque la même.