BOCQUELET Théo
FAN Gao

RAPPORT TP3 NF16
-
GESTION D'UNE BIBLIOTHEQUE

Durant ce TP, nous avons réalisé un programme permettant la gestion d'une bibliothèque.

Ce programme est basé sur les structures de données *Livre*, *Rayon* et *Bilbiothèque*. Chaque structure permet respectivement de :

- gérer les informations propres à chaque livre et pointer sur le livre suivant
- gérer les informations propres à chaque rayon et pointer sur le premier livre du rayon et le rayon suivant
- gérer les informations propres à la bibliothèque et pointer sur le premier rayon

Ces trois structures sont donc des listes chaînées. L'utilisation de listes chaînées nous permet de gérer dynamiquement les données. L'insertion et la suppression, à n'importe quel endroit de la liste, sont plus faciles et moins complexes que pour un tableau classique. En revanche, la recherche reste identique. (à revoir ?)

Pour gérer cette bibliothèque, nous avons utiliser plusieurs fonctions, le nom de chaque fonction explicite son rôle.

Nous allons détailler les cas importants que nous avons gérer sur certaines fonctions :

- *ajouterLivre/ajouterRayon :*
  - dans le main (ajouterLivre) : test si le rayon existe ou non
  - rayon ou bibliothèque vide
  - insertion en tête/au milieu/à la fin
  - le livre/rayon existe déjà
- *afficherBiblio/afficherRayon :*
  - bibliothèque ou rayon vide
  - dans le main (afficherLivre) : test si le rayon existe ou non
- *emprunterLivre :*
  - *le rayon est vide*
  - *le livre n'est pas dans le rayon*
  - *livre disponible ou non*
- *supprimerLivre :*
  - dans le main : test si le rayon existe ou non
  - le rayon est vide
  - suppression en tête/au milieu/à la fin
  - suppression en tête : la tête a-t-elle un successeur ?
  - le livre n'est pas dans le rayon
- *supprimerRayon :*

- la bibliothèque est vide
- suppression en tête/au milieu/à la fin
- suppression en tête : la tête a-t-elle un successeur ?
- le rayon n'est pas dans la bibliothèque
- supprimer tous les livres du rayon avant de supprimer le rayon

Complexité des fonctions :

- *creerLivre/creerRayon/creerBiblio :*

Il n'y a qu'une suite d'affectations, donc une suite d'opérations élémentaires. La complexité est donc constante. $\Omega(k)=O(k)=\theta(k)$ où k est une constante.

- *ajouterLivre :*

Meilleur des cas : cela correspond au fait que le rayon soit vide ou que le titre du livre à ajouter se trouve avant le titre du premier livre du rayon. Il n'y a donc que deux tests à effectuer et des affections, nous ne faisons donc que des opérations élémentaires. C'est pourquoi on est en $\Omega(k)$ où k est une constante.

Pire des cas : cela correspond au cas où il faut rechercher la position du livre à insérer et, en même temps, rechercher si le livre existe déjà dans le rayon. Nous allons donc effectuer au maximum n+1 tests (pour le while) et n tests pour le if dans le while, n étant la taille de notre liste. Il ne reste plus que des opérations élémentaires ensuite (test+affectations). De cette manière nous sommes en $O(2n+k)=O(n)$.

Ω et O étant différentes, on ne peut pas déterminer θ.

La fonction *ajouterRayon* effectue les mêmes opérations que *ajouterLivre,* leur complexité est donc identique.

- *afficherBiblio :*

Meilleur des cas : la bibliothèque est vide, il n'y a donc rien à afficher, un seul test est effectué. La complexité est donc en $\Omega(k)$ où k est une constante.

Pire des cas : s'il y a n rayons, le while effectuera n+1 tests, n printf et n affectations dans la boucle. Il y aura eu précédemment, une affectation, un test et deux printf. Ainsi, la complexité se trouve en $O(3n+k)=O(n)$.

- *afficherRayon :*

Meilleur des cas : la complexité est ici identique à celle d' *afficherBiblio.* En effet, les mêmes opérations sont effectuées. La complexité est donc en $\Omega(k)$ où k est une constante.

Pire des cas : là aussi les mêmes opérations sont effectuées par rapport à la fonction précédente. La seule différence se trouve au sein du while où n tests seront effectués afin de savoir si le livre est disponible ou non. La compléxité est donc en $O(4n+k)=O(n)$.

- *emprunterLivre :*

Meilleur des cas : si le rayon est vide, il n'y a pas de livres à tester. La complexité est donc constante en $\Omega(k)$ où k est une constante.

Pire des cas : cela correspond au cas où le livre se trouve à la fin du rayon ou bien au cas ou le livre n'est pas dans le rayon. En effet, il faudra ainsi faire n+1 tests au niveau du while avec n affectations. Ensuite, un nombre constant de tests et printf sera lancé. La complexité est donc en $O(2n+k)=O(n)$.

- *supprimerLivre :*

Meilleur des cas : si le rayon est vide, aucune recherche n'est à effectuer. La complexité est donc en $\Omega(k)$ où k est une constante.

Pire des cas : cela correspond au cas où le livre à supprimer n'est pas dans le rayon ou au cas où il est à la fin du rayon. Ainsi, n+1 test sont effectués et n affectations. Ensuite, seules des opérations élémentaires sont lancées. La complexité est donc en $O(2n+k)=O(n)$.

- *supprimerRayon :*

La complexité dans le meilleur des cas est identique à celle de la fonction précédente. Les mêmes opérations sont mises en place.

Pire des cas : cela correspond au cas où le rayon à est à la fin de la bibliotèque. Ainsi, n+1 test sont effectués et n affectations afin de rechercher le rayon. Puis il faut supprimer tous les livres du rayon (m livres) d'où l'utilisation d'un autre boucle à m+1 tests et 3 fois m affectations. Enfin, il n'y a plus que des opérations élémentaires. La complexité est donc en $O(2n+4m+k)=O(n+m)$.

- *rechercherLivres :*

La complexité dans le meilleur des cas et celle dans le pire des cas sont les mêmes puisqu'il faudra parcourir chaque livre de chaque rayon pour trouver ce que l'on cherche. Ainsi, nous devrons parcourir les m livres des n rayons et en même temps comparer notre chaîne entrée aux titres de taille t de chaque livre. Enfin, on testera m fois si le titre est bon ou non. Le reste correpond à des opérations élémentaires. La complexité sera donc en $\theta(n*2m*t+k)= \theta(nmt)$.