Structures de données usuelles

Option Informatique

Anthony Lick
Lycée Janson de Sailly

Introduction

Introduction

Option informatique

Un fil rouge du programme des deux années d'option informatique est l'utilisation de **structures de données abstraites** classiques dans des algorithmes.

Plus basiquement, pour résoudre un problème donné, un informaticien se tournera naturellement vers une structure de données bien connue s'il peut en faire usage efficacement.

Introduction

Exemple: pile ou file

Lors de l'exploration d'un graphe, il faut stocker les sommets découverts mais dont les voisins n'ont pas encore été examinés.

Le choix de la structure (**pile** ou **file**) mène à une exploration du graphe dite "**en profondeur**" ou "**en largeur**".

Introduction

Exemple: file de priorité

Pour un graphe dont les arêtes sont munies de poids positifs, la distance entre deux sommets est la somme des poids des arêtes sur un chemin entre les deux sommets (on prend le chemin minimisant cette somme).

Une généralisation du parcours en largeur explorant les sommets depuis une origine fixée, par distance à l'origine croissante, se fait via l'algorithme de Dijkstra : une **file de priorité** est utilisée pour gérer les sommets en cours d'exploration.

Introduction

Exemple: dictionnaire

L'informaticien d'une entreprise souhaite référencer les clients, et pouvoir accéder rapidement aux informations concernant un client à l'aide de son identifiant unique, attribué à la création de la fiche client.

Une structure de **dictionnaire** est toute indiquée pour stocker les informations sur les clients.

Structures de données abstraites

Définition

Une **structure de données abstraite** est la donnée d'un **type**, et des **opérations** que l'on peut effectuer dessus.

Structure abstraite vs implémentation concrète

Cette définition est un peu évasive, mais ce qu'il faut retenir est qu'une structure abstraite est indépendante d'une implémentation concrète.

Ce qui est important est la description des opérations que le type permet, plutôt que la manière dont ces opérations sont réalisées.

Il y a plusieurs avantages à cela.

Structures de données abstraites

Avantages

- Si on possède déjà une implémentation d'une structure de données abstraites, on peut programmer des algorithmes avec sans savoir comment elles sont implémentées : on les voit comme des "boites noires" complexes.
  $\hookrightarrow$ C'est le point de vue du programmeur utilisateur : il n'a pas besoin de savoir comment ça marche pour utiliser une structure abstraite.
- Si on change l'implémentation d'une structure pour une autre implémentation, les algorithmes faisant usage de la structure que l'on a déjà implémentés seront compatibles avec la nouvelle implémentation.

En pratique

Ce qui permet de juger si une implémentation est meilleure qu'une autre est essentiellement la **complexité**.

Lorsqu'on voudra implémenter concrètement une structure abstraite (on parle de structure concrète), on cherchera à avoir la meilleure complexité (en temps et/ou en mémoire) possible.

Piles

Piles

Pile

Une **pile** est une structure de donnée linéaire, dans laquelle les éléments sont insérés ou supprimés suivant le principe **LIFO** (last in, first out).

Visuellement, le seul élément accessible est celui situé au sommet de la pile, qu'il faudrait retirer pour accéder à l'élément situé en dessous.

Inversement, un élément qu'on rajoute à la pile sera rajouté au sommet.

Piles

Définition (Pile)

Une **pile** est une structure abstraite, supportant les opérations suivantes :

- **création** d'une pile vide ;
- test d'**égalité** au vide ;
- **accès** au sommet d'une pile non vide ;
- **retrait** de l'élément au sommet d'une pile non vide ;
- **ajout** d'un élément au sommet de la pile.

Piles

Exemple

La pile est un outil de base en informatique, dont les utilisations sont multiples. Voici quelques exemples :

- la gestion des appels de fonctions dans l'exécution d'un programme se fait via une pile d'appels : lorsqu'une fonction est appelée, les informations relatives à l'appel (adresse de retour, notamment) sont empilées sur la pile d'appels, et seront dépilées lorsque la fonction terminera son exécution ;
- les boutons "page précédente" et "page suivante" d'un navigateur utilise deux piles ;
- le parcours en profondeur d'un graphe, déjà évoqué, fait usage d'une pile.

Files

Files

File

La file est une autre structure linéaire, assez semblable à la pile, mais qui fonctionne sur le principe FIFO (first in, first out).

Lorsqu'on insère un élément dans une file, celui-ci ne pourra être retiré qu'après que tous les éléments insérés avant l'aient été.

Files

Définition (File)

Une **file** est une structure abstraite, supportant les opérations suivantes :

- **création** d'une file vide ;
- test d'**égalité** au vide ;
- **retrait** de l'élément en tête d'une file non vide ;
- **ajout** d'un élément en queue de file.

Files

Exemple

La file n'est pas d'un usage aussi courant que la pile en informatique, citons néanmoins quelques exemples :

- une imprimante "bas de gamme" (qui ne gère pas les priorités, cf. section suivante) à qui l'on envoie des documents à imprimer les traitera séquentiellement : le premier document à être envoyé sera imprimé en premier ;
- le parcours en largeur d'un graphe traite les sommets par distance à l'origine (nombre minimal d'arêtes à parcourir depuis l'origine) croissante : il suffit de rajouter les sommets dans une file lorsqu'ils sont découverts pour respecter cet ordre dans le parcours.

Files de priorité

Files de priorité

File de priorité

Une **file de priorité** suit le même principe qu'une file, mais à chaque élément inséré dans la file est attaché une **priorité**, en général représentée par un entier.

La file de priorité stocke donc des couples (e, p), où e est un élément et p un entier (la priorité).

Le prochain élément à sortir de la file est celui qui a la plus grande priorité (on parle de file de priorité max ou min suivant si on prend le plus grand ou le plus petit entier p).

Files de priorité

Définition (File de priorité)

Une **file de priorité** est une structure abstraite, supportant les opérations suivantes :

- **création** d'une file de priorité vide ;
- test d'**égalité** au vide ;
- **retrait** de l'élément de plus grande priorité d'une file non vide ;
- **ajout** d'un élément avec une priorité donnée ;
- **modification** de la priorité d'un élément (cette opération a un sens seulement si on impose l'unicité des éléments e stockés dans la file de priorité, ce qui est le cas en général).

Files de priorité

Exemple

Quelques exemples d'utilisation :

- le système d'exploitation gère les processus à exécuter via des priorités. Une tâche de moindre importance sera mise en attente pour être exécutée par le processeur après celles de plus grande priorité ;
- il en va de même pour les documents gérés par une imprimante "haut de gamme" : il est possible d'augmenter ou de diminuer la priorité d'un document ;

Files de priorité

Exemple

Quelques exemples d'utilisation :

- le parcours en largeur d'un graphe se généralise à des **graphes pondérés** (les arêtes sont munies d'un poids supposé positif) via l'algorithme de **Dijkstra** : les sommets à explorer en priorité sont ceux qui ont la plus petite distance à l'origine, l'ordre d'exploration est donc géré par une file de priorité min.
  Cet algorithme permet de calculer toutes les distances entre l'origine du parcours et chacun des sommets.

Dictionnaires

Dictionnaires

Dictionnaire

Le **dictionnaire** est également une structure très courante en informatique.

Il stocke des couples (k, e) où k est la **clé**, et e l'**élément** associé.

Dans un dictionnaire, les clés k sont supposées toutes **distinctes**.

Dictionnaires

Définition (Dictionnaire)

Un **dictionnaire** est une structure abstraite, supportant les opérations suivantes :

- **création** d'un dictionnaire vide ;
- test d'**égalité** au vide ;
- test de **présence** d'un élément ayant une clé k donnée ;
- **retrait** de l'élément ayant une clé k donnée (s'il existe) ;
- **ajout** d'un élément e avec une clé k donnée (s'il n'y a pas déjà d'élément de clé k).

Dictionnaires

Exemple

Quelques exemples d'utilisation :

- un dictionnaire (au sens usuel) peut être vu comme un dictionnaire informatique : les clés sont les mots de la langue, les éléments sont les définitions ;
- un logiciel de programmation fait usage d'un dictionnaire pour stocker les variables définies par l'utilisateur, et leurs valeurs ;

Dictionnaires

Exemple

Quelques exemples d'utilisation :

- dans la réalisation concrète d'une file de priorité, on peut faire usage d'un dictionnaire pour implémenter l'opération "augmenter ou diminuer la priorité d'un élément".
  Les clés de ce dictionnaire sont les éléments x de la file de priorité ;
- plus généralement, tout stockage de données où on attribut un identifiant unique (la clé) aux éléments stockés peut être implémenté via un dictionnaire.