Algorithmique et programmation

Piles, files et arbres

R.Gosswiller

Sommaire

Piles et Files

Principes généraux

Principes

Les piles et files sont des structures de données aux utilisations spécifiques.
Une pile est une structure où seul le **dernier** élément ajouté est accessible.

Une file est une structure où seul le **premier** élément ajouté est accessible.

Différentes représentations possibles, par tableaux/listes chaînées.

Structures limitées, mais optimisées !

Piles

Principe

Une pile ne permet d'accéder qu'au dernier élément ajouté.
Représentation en *'pile d'assiettes'*
Aussi nommée FILO - First Input Last Output (le Premier à avoir été Ajouté sera le Dernier à être Retiré)

Exemples

Algorithmes de parcours/tri
Pile d'instructions

Implémentation des piles

Solution

```
1  struct pile_ {
2          myData data;
3          struct pile_* next;
4  };
5  typedef struct pile_* pile;
6
7  void push(pile p, myData d) {
8          while (p->next != NULL) { p = p->next; }
9          p->next = createNew(d);
10 }
```

Implémentation des piles

Solution

```
1  myData pop(pile p) {
2           pile prev;
3           while (p->next != NULL) {
4                   prev = p;          //on garde le precedent
5                   p = p->next;
6           }
7           prev->next = NULL;  //on 'decroche' le dernier
8           myData d = p->data; //on retiens la valeur
9           free(p);            //on free() le dernier
10          return d;
11 }
```

Files

Principe

Une file ne permet d'accéder qu'au premier élément ajouté.
Représentation en *'file d'attente'*
Aussi nommée FIFO - First Input First Output (le Premier à avoir été Ajouté sera le Premier à être Retiré)

Exemples

Accès concurrent (à un serveur internet par exemple)
Installation d'un programme

Implémentation des files

Solution

```
1  struct pile_ {
2          myData data;
3          struct pile_* next;
4  };
5  typedef struct pile_* pile;
6
7  void push(pile p, myData d) {
8          while (p->next != NULL) { p = p->next; }
9          p->next = createNew(d);
10 }
11
12 pile pop(pile p, myData* value) {
13         pile prev;
14         if(p->next!=NULL){pile prev = p->next;}
15         *value = p->data;          //on retiens la valeur
16         free(p);                          //on free() le premier
17         return prev;
18 }
```

Remarques

A noter :

- Une 'bonne' implémentation des piles/files nécessite de connaître le facteur critique
- File : besoin de parcourir la chaine en push ou en pop
- Pile : passer par un tableau + realloc() économise de la mémoire
- Autre facteurs : alternance écriture/lecture, type de données...

Arbres

Les arbres

Définition

En mathématiques, un arbre est un graphe orienté, connexe et acyclique

Principe

Structure de données hiérarchisée
On parle de la racine (root), les feuilles (leaves), et les noeuds intermédiaires

- Racine : pas d'élément précédent ('père')
- Feuille : Pas d'éléments suivants ('enfants')

Vocabulaire

Etiquettes

Chaque noeud peut être associé à une valeur (étiquette)
On parle alors d'arbre étiqueté
L'étiquette peut être de n'importe quel type (int, float, char, voiture, ...)

Hauteur

La hauteur (ou profondeur) d'un noeud n représente le nombre d'arrêtes à parcourir entre la racine et n

Chemin

On appelle chemin du noeud n la liste des noeuds traversés entre la racine et n

Les arbres

Exemple

```
1   struct tree_ {
2       struct tree_ * sonL;
3       struct tree_ * sonD;
4       struct tree_ * sonR;
5       int val;
6   };
7
8   struct tree_ {
9       struct tree_ sons[10];
10      int val;
11  };
```

Utilisation des arbres

- Systèmes de fichiers
- Arbre de décision (IA...)
- Outil algorithmique

Arbres binaires

Définition

Arbre où chaque noeud possède $0, 1$ ou 2 fils.

Syntaxe

```
1 typedef struct tree_
2 {
3     element e;
4     struct tree_ *left;
5     struct tree_ *right;
6 } tree ;
```

Attention

Lors de la gestion d'un arbre, les fonctions de retrait de noeud sont à développer de manière précautionneuse !

Parcours en profondeur (DFS - Depth First Search)

Principe

Parcours d'une branche jusqu'à son extrémité, puis l'autre
Approche récursive possible ou construction d'une pile.

A voir

depthFirstTraversal.pdf

Parcours en largeur (BFS - Breadth First Search)

Principe

Analyse de chaque niveau de l'arbre avant de descendre au niveau suivant
Construction d'une file avec tous les voisins d'un sommet S

A voir

breadthFirstTraversal.pdf

Conclusion

- Piles et files
- Arbres
- Algorithmes d'arbres binaires