Structures de Données

Licence Pro GTSBD

Pourquoi ?

- La façon de stocker les données influence grandement les performances de l'algorithme
- Différentes structures de données
  - Prennent plus ou moins de place mémoire
  - Permettent d'optimiser certaines opérations (lectur d'un élément en temps constant, insertion d'un élément en temps constant, ...)

Le Tableau

- Les éléments sont indexés par leur position dans le tableau T[i]
- Écriture d'un élément à la position j en O(1)

```
T[j] <- elem ;
```

- Lecture d'un élément à la position k en O(1)

```
elem <- T[k] ;
```

- Parcours du tableau :

```
Pour i allant de 1 à taille(T) faire {
  Afficher(T[i]) ;
}
```

ou

```
Pour tout e dans T faire {
  Afficher(e) ;
}
```

- Taille fixe - impossibilité d'insérer entre deux éléments (→ décalage, puis écriture)

Le Tableau en PHP

- `$T = array(2, 3, 5, 7) ;`
- `$T2 = array("lun", "MAR", "mer", "jeu", "ven", "sam", "dim") ;`
- Les indices commencent à 0 !!
- `$T2[1] <- "mar" ;`
-
```
foreach($T2 as $jour) {
    echo "$jour\n" ;
}
```
- Attention, un `array` peut croître dynamiquement, ce n'est donc pas une implémentation exacte d'un tableau, mais une structure plus souple (et moins performante).
- `$argv` est un `array`

La Liste

- Structure souple (alors qu'un tableau est rigide
- Lecture/insertion/suppression en début et fin e O(1)
- Lecture/insertion/suppression au milieu de la liste en O(n)

La Liste en PHP

- `$L = array()`
- Ajouter en fin de liste :
  `array_push($L, $elem) ;`
- Ajouter en début de liste :
  `array_unshift($L, $elem) ;` `O(n) !!!`
- Enlever en fin de liste :
  `$elem <- array_pop($L) ;`
- Enlever en début de liste :
  `$elem <- array_shift($L) ;` `O(n) !!!`

La Pile

- La pile de bouquins sur votre bureau. On peut
  - empiler un livre
  - dépiler et récupérer le livre du dessus
  - savoir si la pile est vide ou non
- En anglais Stack
- Appelée LIFO « Last In, First Out »

La Pile en PHP

- $P = array() ;
- array_push($P, $livre) ;
- $livre = array_pop($P) ;
- if (count($P) == 0) { echo "Pile vide\n" ; }

La File

- La file d'attente (, d'impression, ...). On peut :
  - envoyer un nouveau document dans la file
  - traiter le document en tête
  - savoir si la file est vide ou non
- En anglais Queue
- Appelée FIFO « First In, First Out »

La File en PHP

- $F = array() ;
- array_push($F, $document) ;
- $document = array_shift($F) ;
- if (count($F) == 0) { echo "File vide\n" ; }

Table associative

- Tableau dont les éléments sont des couples (clé, valeur). La clé devant être unique dans la table associative. Ex :

```
dicoAnglFr <-
{ "Rabbit" : "Lapin",
  "Fox" : "Renart",
  "Dog" : "Chien" }
```

- `dicoAnglFr["Fox"] <- "Renard" ;`
-
```
Pour toute clé dans dicoAnglFr {
    Écrire(dicoAnglFr[clé]) ;
}
```

- Un tableau peut être vu comme une table associative dont les clés sont des indices

Table associative en PHP

- Un array est une table associative !
-
```
$TA = array("Rabbit" => "Lapin",
"Fox" => "Renard", "Dog" :
"Chien") ;
```
-
```
foreach($TA as $cle => $valeur)
    echo "$cle -- $valeur\n" ;
}
```

Exercice

- Voir feuille d'exercices

Spl* Datastructures

- Depuis PHP 5.3, de vrais types de données ont été implémentés :
  - SplFixedArray : vrai tableau
  - SplDoublyLinkedList : vraie liste chainée
  - SplStack : pile (basée sur SplDoublyLinkedList)
  - SplQueue : file (basée sur SplDoublyLinkedList)
  - SplObjectStorage : map
  - ...