STRUCTURE DE DONNEE

DJOSSOU Kokou Armand Light

TONGNI Rebecca

❖ LES DIFFERENTS TYPES DE STRUCTURE DE DONNEE

Il existe deux types de structures de données en C qui sont : les structures de données linéaires et les structures de données et les structures de données non linéaires

A. **LES STRUCTURES DE DONNEES LINEAIRES (confère cours)**
B. **LES STRUCTURES DE DONNEES NON LINEAIRE**

a. Table de hachage

- Role : Stocker des paires clé-valeur
- Caractéristiques :

. Utilise une fonction de hachage pour mapper les clés à des indices

. Collisions gérées par chainage(liste chainée) ou sondage

Exemple :

```
struct HashNode {
    int key;
    int value;
    struct HashNode* next;
};
struct HashTable {
    struct HashNode** table;
    int size;
};
```

b. Tas

- Role : Structure arborescente pour gérer des priorités ou trier des données
- Caractéristiques :

. Max-Heap : le parent est plus grand que ses enfants

. Min-Heap : le parent est plus petit que ses enfants

. Utilisé pour les files de priorités , algorithmes comme Dijkstra ou tri par tas

Exemple de Tas :

```
struct Heap {
    int* array;
    int size;
    int capacity;
};
```

c. Trie

Rôle : Stocker des chaînes de caractères pour une recherche rapide par préfixe.

Caractéristiques :

•Chaque nœud représente un caractère.

•Utilisé pour les dictionnaires, autocomplétion.

•Temps : O(m) où m est la longueur de la chaîne

Exemple :

```
struct TrieNode {
    struct TrieNode* children[26];
    int isEndOfWord;
};
```

C. AUTRES STRUCUTRES DE DONNES SPECIALISEES

a. Skip lists

- Structure probabiliste pour accélérer les recherches dans une liste chaînée ordonnée
- Combine les avantages des listes chaînées et des arbres équilibrés.
- Chaque nœud peut avoir plusieurs pointeurs vers des nœuds plus loin (niveaux

Exemple :

```
struct SkipNode {
    int data;
    struct SkipNode** next; // Tableau de pointeurs pour les niveaux
    int level;
};
```

b. Union-find

Gére des ensembles disjoints et vérifier si des éléments appartiennent au même ensemble

Exemple :

```
struct DisjointSet {
    int* parent;
    int* rank;
    int size;
};
```

c. Bit arrays

Stocke des données booléennes de manière compacte

Exemple :

```
unsigned char bit_array[100]; // Chaque octet stocke 8 bits
```

d. Filtres de Bloom

Vérifie rapidement si un élément appartient probablement à un ensemble

Exemple :

```
struct BloomFilter {
    unsigned char* bit_array;
    int size;
    int num_hash_functions;
};
```

❖ TABLEAU RECAPITULATIF DES TYPES DE STRUCTURE DE DONNEES

Structure	Rôle Principal	Complexité(moyenne)
Tableau	Stockage indexé	Accès O(1), Insertion/Suppression O(n)
Liste chaînée	Gestion dynamique	Accès O(n), Insertion/Suppression O(1)
Pile	LIFO (historique, appels)	Push/Pop O(1)
File	FIFO (tâches, files d'attente)	Enqueue/Dequeue O(1)
Arbre binaire	Organisation hiérarchique	Recherche O(log n) si équilibré
Graphe	Relations complexes	Dépend de l'algorithme
Table de hachage	Accès rapide clé-valeur	O(1) pour opérations de base
Tas	Gestion des priorités	O(log n) pour insertion/suppression
Trie	Recherche de chaînes	O(m) où m est la longueur de la chaîne

❖ LES TYPES DE LISTE CHAINEE

Liste Chainée	Pointeurs par Noeud	Navigation	Avantages	Inconvénients
Simple	1 (suivant)	Unidirectionnelle	Simple, moins de mémoire	Pas de navigation arrière, accès O(n)
Double	2 (suivant, précédent)	Bidirectionnelle	Navigation facile, suppression rapide	Plus de mémoire, gestion complexe
Circulaire	1 ou 2	Cyclique	Parcourt en boucle	Risque de boucles infinies
Avec sentinelle	Comme simple/double	Dépend du type	Simplifie les algorithmes	Surcharge mémoire légère