비선형 자료구조 는 데이터 요소들이 일렬로 배열되지 않고 계층적이거나 그래프 형태로 연결되어 있는 자료구조를 의미한다. 하나의 데이터 요소가 여러 개의 다른 요소와 연결될 수 있으며 이러한연결 관계가 복잡하게 얽혀 있는 구조를 갖는다.

비선형 자료구조의 종류

1. 트리
• 이진 트리
• 이진 탐색 트리
2. 그래프
• 인접 리스트
• DFS (깊이 우선 탐색)
• BFS (너비 우선 탐색)
3. 해시 테이블
• 해시 함수
• 충돌 해결 기법 (체이닝, 개방 주소법)

비선형 자료구조의 특징

1. 복잡한 데이터 관계를 효율적으로 표현할 수 있다.
2. 다양한 문제 해결에 활용될 수 있다.
3. 특정 연산에 있어 높은 효율성을 보일 수 있다.
4. 구현이 복잡하고, 알고리즘 설계가 어려울 수 있다.
5. 메모리 사용량이 많을 수 있다.

배열 은 동일한 자료형의 값들을 순서대로 저장하는 기본적인 선형 자료구조이다.

배열의 특징

1. 순차적 저장 : 메모리 상에서 연속된 공간에 데이터가 순차적으로 저장이 된다.
2. 인덱스 기반 : 각 요소는 인덱스를 가지며 인덱스를 통해서 요소에 접근 할 수 있다.
3. 동일 자료형 : 배열에 저장되는 모든 요소는 동일한 자료형으로 이루어져야 한다.
4. 고정된 크기 : 배열의 크기는 생성 시에 정해지며, 이후에 크기를 변경하는 것은 어렵다.
5. 중복 허용 : 동일한 값을 여러 개 저장할 수 있다.

자바에서 배열 선언시

```java
//정수형 배열 선언 및 초기화
int[] numbers = new int[4]; //크기가 5인 정수형 배열 생성

//배열 요소에 값 할당
numbers[0] = 10;
numbers[1] = 20;
numbers[2] = 30;
numbers[3] = 40;

//배열 요소 출력
for (int i = 0; i < numbers.length; i++) {
    System.out.printl(numbers[i]);
}

//배열 초기화 시 값 할당
String[] names = {"대한", "민국", "만세};
```

다차원 배열 선언시

```java
int[][] matrix = new int[3][4]; //3행 4열의 2차원 배열
```

[0][0]	[0][1]	[0][2]	[0][3]
[1][0]	[1][1]	[1][2]	[1][3]
[2][0]	[2][1]	[2][2]	[2][3]

matrix 변수 초기화 형태

`int[][]` : 2차원 정수형 배열을 의미한다.
`matrix` : 배열의 이름이다.

`new int[3][4]` : 3행 4열의 2차원 배열을 생성한다.

그래프는 노드와 간선으로 구성된 자료구조이다. 노드는 데이터를 저장하는 공간이고 간선은 노드 간의 관계를 나타낸다.

1. 노드 : 데이터를 저장하는 기본 단위이다.
2. 간선 : 두 노드를 연결하는 선이다. 방향성이 있을 수도 있고 없을 수도 있다.
3. 인접 : 간선으로 연결된 두 노드를 서로 인접하다고 한다.

그래프의 종류

1. 무방향 그래프 : 간선에 방향성이 없는 그래프이다. A에서 B로 갈 수 있다면 B에서 A로도 갈 수 도있다.
2. 방향 그래프 : 간선에 방향성이 있는 그래프이다. A에서 B로 갈 수 있다고 해서 B에서 A로 갈 수 있는 것은 아니다.
3. 가중치 그래프 : 간선에 가중치가 부여된 그래프이다. 이 가중치는 거리, 비용 등을 나타낼 수 있다.

인접 리스트

그래프를 표현하는 방법 중 하나이다. 각 노드에 연결된 인접 노드들을 리스트 형태로 저장한다.

1. 장점 : 간선의 수가 적은 희소 그래프에서 효율적이다. 메모리 공간을 절약할 수 있다.
2. 단점 : 모든 노드 간의 연결 여부를 확인하려면 모든 리스트를 순회해야 한다.

깊이 우선 탐색(Depth-First Search, DFS)

그래프 탐색 알고리즘 중 하나로 하나의 가지를 끝까지 탐색한 후 다른 가지를 탐색하는 방식이다. 스택 자료구조를 이용하여 구현하며 미로 탐색, 사이클 검출 등에 사용된다.

너비 우선 탐색(Breadth-First Search, BFS)

그래프 탐색 알고리즘 중 하나로 현재 노드와 인접한 모든 노드를 먼저 방문하고 그 다음에 그 노드들과 인접한 노드들을 방문하는 방식이다. 큐 자료구조를 이용하여 구현하며 최단 경로 찾기, 연결 요소 찾기 등에 사용된다.

DFS와 BFS의 차이점

특징	DFS	BFS
탐색 방식	깊이 우선	너비 우선
자료 구조	스택	큐
주요 용도	미로 탐색, 사이클 검출	최단 경로 찾기, 연결 요소 찾기

비선형 자료구조 그래프의 활용

1. 소셜 네트워크 분석 : 친구 관계, 추천 시스템 등을 구현하는 데 사용된다.
2. 최단 경로 찾기 : 길찾기 앱, 네트워크 라우팅 등에 활용된다.
3. 파일 시스템 : 파일과 디렉토리의 관계를 트리 형태의 그래프로 표현할 수 있다.

해시 테이블은 키와 값 쌍으로 데이터를 저장하고 매우 빠른 검색 속도를 제공하는 비선형 자료 구조이다. 마치 사전을 찾는 것처럼 특정 키를 입력하면 해당하는 값을 거의 순식간에 찾아낼 수 있다.

해시 테이블의 작동 원리

1. 해시 함수 : 임의의 길이의 키를 고정된 길이의 해시 값으로 변환하는 함수이다. 이 해시 값은 배열의 인덱스로 사용되어 데이터가 저장될 위치를 결정한다.
2. 배열(버킷) : 해시 값을 인덱스로 사용하는 배열이다. 각 버킷에는 하나 이상의 데이터가 저장될 수 있다.
3. 저장 : 키를 해시 함수에 넣어 생성된 해시 값을 이용하여 해당 버킷에 데이터를 저장한다.
4. 검색 : 찾고자 하는 키를 해시 함수에 넣어 해시 값을 생성하고 해당 버킷에서 데이터를 검색한다.

해시 테이블 사용 이유

1. 빠른 검색 : 해시 함수를 통해 직접 데이터의 저장 위치를 계산하므로 거의 상수 시간(O(1)) 안에 데이터를 찾을 수 있다.
2. 유연한 크기 조절 : 데이터의 양에 따라 배열의 크기를 동적으로 조절할 수 있다.
3. 다양한 응용 : 캐시, 딕셔너리, 심볼 테이블 등 다양한 분야에서 사용된다.

해시 테이블의 장단점

장점
검색, 삽입, 삭제 연산이 평균적으로 O(1)의 시간 복잡도를 가지므로 매우 빠르다.
다양한 자료 구조를 구현하는 데 활용될 수 있다.
단점
해시 함수의 선택에 따라 성능이 크게 달라질 수 있다.
충돌이 자주 발생하면 성능이 저하될 수 있다.

충돌 해결 기법

충돌이 발생했을 때 데이터를 효율적으로 저장하기 위한 다양한 기법들이 있다.

1. 체이닝(Chaining)

각 버킷에 연결 리스트를 연결하여 충돌이 발생한 데이터를 연결 리스트에 추가하는 방법이다.
장점 : 구현이 간단하고 충돌이 자주 발생하더라도 효율적으로 처리할 수 있다.

단점 : 연결 리스트를 사용하기 때문에 추가적인 메모리 공간이 필요한다.

2. 개방 주소법(Open Addressing)

충돌이 발생하면 다른 버킷에 데이터를 저장하는 방법이다.
선형 탐색 : 충돌이 발생한 버킷의 다음 버킷부터 순차적으로 빈 공간을 찾아 데이터를 저장한다.
제곱 탐색 : 충돌이 발생한 버킷에서 특정 함수(ex: i^2)를 이용하여 이동할 버킷을 계산한다.
이중 해싱 : 두 개의 해시 함수를 사용하여 충돌 시 이동할 버킷을 계산한다.
장점 : 추가적인 메모리 공간이 필요하지 않다.
단점 : 클러스터링 문제가 발생할 수 있으며 삭제 연산이 복잡한다

해시 테이블의 활용 예시

1. 캐시 : 데이터를 빠르게 조회하기 위해 해시 테이블을 사용한다.
2. 연관 배열 : 키를 이용하여 값을 빠르게 찾아야 할 때 사용한다.
3. 집합 : 중복되지 않는 요소들을 저장하고 빠른 멤버십 테스트를 수행해야 할 때 사용한다.