**자료구조 기초 유형 정리**

자료구조란?

* 개념 : 데이터 단위(데이터를 구성하는 덩어리)와 데이터 자체 사이의 물리/논리적인 관계이다. 즉, 자료구조는 데이터(자료)를 효율적으로 이용할 수 있도록 컴퓨터에 저장하는 방법이다.
* 선형 자료구조(앞뒤 자료간 1:1 관계)와 비선형 자료구조(앞뒤 자료간 1:n 관계)로 나뉜다.
* 메모리 공간 기반의 연속방식과, 포인터 기반의 연결방식으로 구현을 한다.

선형 자료구조

1. 배열 (O1)

* 값 or 변수의 집합으로 구성된 구조로, 하나 이상의 index or key로 식별된다.
* 자료구조 中 연속방식의 자료형으로, 크기를 지정하고 해당 크기만큼의 메모리 공간을 할당받는 작업을 수행한다.
* 크기가 고정되어 있으며, 한번 생성하면 크기 변경이 불가능하다.

1. 연결 리스트(On : 특정인덱스 접근 위해 전체를 순서대로 읽어야함)

* 데이터의 선형 집합으로, 데이터 순서가 메모리에 물리적인 순서대로 저장되지 않는다.   
  (따라서 관리가 쉽다.)
* 자료구조 中 연결방식의 자료형으로, 동적으로 새로운 노드를 간편하게 삽입/삭제할 수 있다.

1. 스택/큐
   1. 스택
      1. 쌓아올린다는 뜻으로, 접시처럼 차곡차곡 쌓아 둔 형태의 자료구조이다.
      2. LIFO(후입선출) 방식으로, 역순 문자열 & 실행취소 등에 활용된다.
      3. 정해진 한 곳(TOP)을 통해서만 자료의 삽입과 삭제가 이루어진다. (접시의 맨 위)
   2. 큐 : FIFO(선입선출) / 줄서기
      1. 줄 혹은 줄을 서서 기다리는다는 뜻으로, 일반적인 줄서기와 같은 형태의 자료구조이다.
      2. FIFO(선입선출) 방식으로, 인쇄 등 작업 예약, 콜센터 고객 대기, BFS 등에서 활용된다.
      3. 한쪽 끝에서 삽입이 이루어지고 다른 쪽 끝에서 삭제가 이루어진다.
2. 해시 테이블(O(1))

* Key, Value로 데이터를 저장하는 자료구조이다.
* 임의 크기의 데이터를 고정 크기 값으로 매핑하는 데 사용할 수 있는 해시함수가 핵심이며, 이를 통해 각각의 Key값에 고유한 index를 생성한다. 이 index를 통해 값을 가능한 한 빠르게 저장하고 검색한다.
* 로드 / 벡터 : 해시테이블 고정 한계치가 넘어가면 ~
* 해시값(위에서 해시 함수를 통해 생성하는 index)이 충돌하는 경우가 종종 발생한다.  
  충돌값에 대해 연결 리스트로 연결하는 개별 체이닝과(자바),  
  충돌시 빈 공간을 다시 찾아 집어넣는 오픈 어드레싱 방식을 통해 충돌을 해결한다.

비선형 자료구조

1. 그래프

* 정점과 간선으로 이루어져, 정점 간 관계를 표현하는 자료구조이다.
* 정점(=노드) : 데이터가 저장되는 곳  
  간선(=링크) : 노드간의 관계  
  인접 정점 : 간선에 의해 서로 연결된 정점  
  단순 경로 : 경로 중 반복되는 정점이 없는 것  
  차수 : 하나의 정점에 인접한 정점의 수
* 그래프의 노드를 2차원 배열로 표현한 인접행렬 방식과 리스트로 표현한 인접리스트 방식이 있다.
* 각 그래프의 정점을 방문하는 그래프 순회로 DFS, BFS 알고리즘이 있다.

1. 트리

* 하나의 뿌리에서 위로 뻗어 나가는 형상을 뜻하는, 나무의 형상과 같은 자료구조이다. (실제로 표현할 때는 나무의 형상과는 반대 방향으로 표현한다.)
* 루트에서 시작하여, 자식 노드와 간선으로 연결되어 있는 형태이다. 자식 노드 역시 서브 트리 구조를 가지면서 전체 구조가 확장된다.
* 그래프 VS 트리의 차이
  + 1. **트리는 ‘순환 구조를 갖지 않는 그래프’이다.** 즉, 그래프의 범주에는 포함되나 부모가 자식을 가리키는 단방향의 형태만을 가진다.

1. 다양한 트리의 형태
   1. 이진트리
      1. 모든 노드의 차수가 2 이하인 트리
      2. 모든 노드가 0개 OR 2개의 자식을 가진 정 이진 트리, 마지막 레벨 제외 모든 레벨이 완전히 채워진 완전 이진 트리, 모든 노드가 2개의 자식 노드를 지닌 포화 이진 트리가 있다.
   2. 이진 탐색 트리
      1. 노드의 왼쪽 서브트리(자식부터 이어지는 트리)에는 자신보다 적거나 같은 값,  
         노드의 오른쪽 서브트리에는 자신보다 크거나 같은 값을 지닌 트리이다.
2. 힙

* 트리 기반의 자료구조 중 하나로, 완전 이진 트리의 일종이다.  
  (완전 이진 트리 : 마지막 레벨을 제외 한 모든 레벨이 완전히 채워진 트리)
* 정렬된 형태가 아니라는 것 기억하기!
* 항상 균형을 유지하려는 특성을 지니고 있어, 우선순위 큐 / 다익스트라 알고리즘 / 중앙 근사값을 구하는 알고리즘 등에 활용된다.
* 최대 힙 : 부모노드 키값 >= 자식노드 키 값  
  부모가 자식보다 크기에, 결국 루트가 가장 큰 값을 갖게 된다.
* 최소 힙 : 부모노드 키값 <= 자식노드 키값  
  부모가 자식보다 작기에, 결국 루트가 가장 작은 값을 갖게 된다.

1. 트라이

* 트리 기반의 자료구조 중 하나로, 키가 문자열인 것이 특징이다.
* 다진 트리의 형태이며, 문자의 개수만큼 자식이 있어 문자열 탐색에 용이하다.

(포탈 검색 과정 => 각 알파벳별 많이 쓰인 단어대로 파악해서)

알고리즘

1. DFS(깊이 우선 탐색)

* 일반적으로 스택 또는 재귀함수를 이용해 구현하며, 백트래킹에 뛰어난 효용을 보인다.
* 백트래킹이란?
  1. 탐색을 하다가 더 갈 수 없으면(현재 길이 가능성이 없다고 판단하면) 왔던 길을 되돌아가 다른 길을 찾는 방식으로, DFS의 한 종류이다. 모든 경로를 한번씩 이동하는 브루트포스보다 효율적이다.
  2. 수많은 제약 조건을 충족하는 특정 상태를 찾아내는 제약 충족 문제를 푸는데 필수적이다.

1. BFS(너비 우선 탐색)

* 일반적으로 큐를 통해 구현한다. (DFS와 달리 재귀함수로 구현하지 않는다.)
* 최단 경로를 찾는 다익스트라 알고리즘 등에 사용된다.

1. 최단 경로 문제 – 그래프

* 각 간선의 가중치 합이 최소가 되는 두 정점(노드) 사이의 경로를 찾는 문제이다.
* 대표적인 예로 다익스트라 알고리즘이 있다.  
  (음의 간선이 없다는 가정 하에, 그래프의 한 노드에서 각 모든 노드까지의 최단거리를 구하는 알고리즘)

1. 정렬 알고리즘

* 목록의 요소를 특정 순서대로 넣는 알고리즘, 데이터의 집합을 어떠한 기준에 따라 대소관계를 따져 일정한 순서로 줄지어 세우는 알고리즘이다.
* 버블 정렬 : 가장 기초적인 정렬 알고리즘, 배열의 길이만큼 반복하여 각 배열의 아이템을 살펴보고, 필요에 따라 앞뒤 아이템의 순서를 뒤바꾼다. (On2)
* 병합 정렬 : 일정한 실행속도와 안정성으로 상용으로 많이 쓰이는 알고리즘. 쪼갤 수 없을 때까지 배열을 분할한 후, 분할이 끝나면 정렬하면서 정복해 나간다. (분할정복 방식)  
   + 설명 or 장점(안정성이 좋다. 무조건 반반무마니로 답을 구하기 떄문에) (nlogn)
* 퀵 정렬 : 기존 병합정렬에 피벗 개념을 더한 정렬 알고리즘. 리스트 안에 한 요소를 선택해 ‘피벗’으로 설정하며, 피벗보다 작은 요소와 큰 요소를 좌우 구분하여 정렬한다. (nlogn~On2) => 이미 정렬된 데이터에 대해서는 오히려 On2 속도
* 팀 소트 :
* 안정정렬 / 불안정정렬 :

1. 이진 검색(logn)

* 정렬된 배열(리스트)에서 검색 범위를 줄여 나가면서 특정 값을 찾아내는 알고리즘.  
  계속해서 리스트의 중간 값을 기준으로 특정 값을 비교하며 반복한다.

1. 슬라이딩 윈도우

* ‘고정 사이즈’의 윈도우가 이동하면서 윈도우 내에 있는 데이터를 이용해 문제를 풀이하는 알고리즘. 투 포인터와 방식은 비슷하나, 윈도우 사이즈(포인터 사이 값의 수)가 고정이며 이동 역시 단방향으로만 가능하다는 차이가 있다.
* 투 포인터 : 시작과 끝점을 두개의 포인터로 지정하여 문제를 푸는 전략

1. 그리디 알고리즘

* 바로 눈 앞의 이익만을 쫓는 알고리즘. 선택의 순간마다 당장 눈앞에 보이는 최적의 상황을 쫓아 최종적인 해답에 도달하는 알고리즘이다. 한번 선택한 것을 다시 고려하지 않고 지나간다.

1. 분할 정복

* 해결할 수 있을 정도로 간단한 문제가 될 때 까지 문제를 재귀적으로 쪼갠 다음, 하위 문제 결과들을 조합해서 다시 원래 문제의 결과로 만드는 알고리즘. 정렬 알고리즘의 병합 정렬이 대표적인 예이다.

1. 다이나믹 프로그래밍

* 문제를 각각의 작은 문제로 나누어, 해결한 결과를 저장한 뒤 나중에 합하여 최종 문제의 답을 구하는 알고리즘. 분할정복 & 그리디 알고리즘과 풀이 방식이 유사하나, ‘중복된 하위 문제들’의 결과를 저장해 두었다가 풀이해 나간다는 차이점이 있다.
* ‘중복된 하위 문제들’이란, 쪼갠 작은 문제들이 반복된다는 의미이다. 피보나치 수가 그 대표적인 예시이다.
* 피보나치 수 : 0,1,1,2,3,5,8,13….앞뒤 숫자의 합이 그 다음 숫자인 case가 반복되는 수열