1. Dfs와 dfs 둘 다 사용 가능할 때는 dfs가 낫다.
2. array list와 Collections 사용
3. 코테 정렬 : 정렬 라이브러리 쓰기 (최악이라도 nlogn 보장) , 계수 정렬 (데이터 개수가 100만개 이상일 때)
4. 데이터 정렬 할때는 기준을 compartTo로 잡고 this.데이터로 비교
5. 오름차순 내림차순 반대로 할때는 데이터 add할 때 –(마이너스)를 붙여주면 됨.
6. Radix sort는 필요한 직군이 있고 아닌 직군이 있음
7. 딕셔너리보다는 리스트가 나은게 메모리도 있고, ~~
8. Tostring 은 객체에 정해진 형식이 없기에 정해주는 것

정렬과 그리디 다 풀기

정렬이란 데이터를 특정한 기준에 따라 순서대로 나열하는 것을 말합니다.

문제 상황에 따라 적절한 알고리즘 사용

1. 선택정렬 : 가장 작은 데이터를 선택해서 가장 앞의 것과 바꾸기 (swap 필요) n2
2. 삽입정렬 : 처리되지 않은 데이터를 하나씩 골라 적절한 위치에 삽입 n2, 최선n
3. 퀵정렬 : 기준 데이터를 설정 (피봇)하고, 기준보다 큰 데이터와 작은 데이터의 위치를 바꾼다. 그리고 분할 평균 nlongn 최악 n2 => 너비 x 높이 (분할이기에 logn x n)
4. 계수 정렬 : 특정한 조건이 부합할 때만 사용할 수 있지만 매우 빠르게 동작하는 정렬 알고리즘 입니다. (데이터의 크기의 범위가 제시되어있을 때) n+k
5. 데이터를 하나씩 확인하며 데이터의 값과 동일한 인덱스의 데이터를 1씩 증가시킵니다.
6. 최종리스트에는 각 데이터가 몇 번씩 등장 했는지 그 횟수가 기록됨.
7. 결과할때는 첫번째 데이터부터 하나씩 반복하며 인덱스 출력
8. 반복문 2개로 가능 cnt[arr[i]] += 1;
9. 백준 11650,
10. 나열된 이름은 인접한 두개씩 묶어서 비교하면 됨 (줄세우기 문제) 이름이 사전순으로 증가하는지 감소하는지 체크
11. 삼각형 문제 : 정렬해서 인접한 3개씩 묶어서 삼각형 조건에 부합하면 통과

그리디 알고리즘

현재 상황에서 지금 당장 좋은 것만 고르는 방법

그리디 해법은 정당성 분석이 중요

문제를 풀기위한 최소한의 아이디어를 떠올릴 수 있는 능력

Ex) 거스름돈 문제

가장 큰 화폐 단위부터 거슬러 주면 됨

-곱하기 더하기 문제 = 하나라도 0이나1이면 더하기, 아니면 곱하기

완전 탐색, 구현, 정렬, dfs/bfs = 피지컬 – 코테

Dpp, 그리디 = 뇌지컬 – 대회

뭐 프레임이나 언어 쓴거 테스트에서 면접 물어보면 원래~사용했는데 뭐가 더 효율적이거나, 가장 언어의 뭐 장점 들어서 이유 말하는게 좋음.

포트폴리오는 스토리텔링 중요!

Atm 문제 : 걸리는 시간을 오름차순으로 정렬하고 풀기

백준 18310, 2437