2021.12.10 부스트캠프 준비 알고리즘 정리

1. 알고리즘 문제 해결 과정

- 지문 읽기

- 요구 복잡도 분석

- 문제 해결을 위한 아이디어 찾기

- 소스 코드 설계 및 코딩

최단경로 같은 경우 ->INF사용 및 10억 미만 일 경우 1e9를 이용

round() 함수 가우스 함수 같은 거임

파이썬에서 / 는 결과가 실수형

//를 사용해야 몫이 나옴

\*\*는 거듭제곱을 뜻한다.

슬라이싱 -> [A:B:C] A 부터 B전까지 C씩 건너뛰면서

리스트 컴프리헨션

array =[ i for \_in ranage(10)] #반복을 적고 If 문을 적어주는 형태도 가능

리스트 2차원 리스트 N\*M 리스트초기화

array =[[0]\*m for \_in range(n)]

반복을 수행하되 반복을 위한 변수 값 없이 사용하려면 \_를 사용하면 된다.

insert() 특정 인덱스에 삽입

tuple 형태 : 한번 선언된건 변경할 수가 없다. 튜플은 소괄호를 사용하여 정의한다.

장점 : 서로 다른 성질의 데이터를 묶어서 관리할 때 편함, 최단 경로 알고리즘에서는 튜플 자료형으로 정의를 많이한다. 데이터 나열을 해싱의 키값으로 사용해야할 때(키 값 사용가능),

리스트보다 메모리를 효율적으로 사용해야 할 때

dict 자료형 : 원하는 “ 변경 불가능한 자료형” 키로 사용 가능

처리 속도 는 O(1) 시간 처리를 한다.

정의 : 중괄호 {} 안에 , 를 기준으로 구분하여 넣어줘도 가능하다.

특징 : 순서가 없으므로 인덱싱 사용이 불가능하다.

파이썬의 입력 받는 방법

1. sys.stdin.readline().rstrip() [ import sys 해줘야함] 단, Enter가 줄바꿈으로 인식되므로 rstrip()를 사용해 주도록 한다.
2. input() 한줄의 문자열을 입력 받는 함수이다.

map() 리스트의 모든 원소에 각각 특정한 함수를 적용할 때 사용한다

ex) list(map(int,input().split())); 각각 원소를 int처리해서 리스트 형태로 받아주는 형태

f-string {변수} 넣어주면 변수 형태 사용가능

and or not -> True/False 반환

X in 리스트 / X not in 문자열 같은 형태 역시 True /False반환

pass 해서 나중에 코드를 작성해주는 형태도 가능함

Continue 건너뛰고 실행

break 반복문 즉시 탈출

0<x<20 이러한 표현이 Python에서만 가능함.

global a 를 사용해야 전역 변수 사용가능 /물론 지역변수를 우선 참조하긴 한다.

람다 함수 : 한번만 함수를 사용해야할 때 유용하다.

lambda x : x[1] <sorted with key

실전 유용한 라이브러리 정도 외우면 될 것 같다. -> 따로 정리

**알고리즘**

Greedy 알고리즘

현재 상황에서 가장 좋은것만 구함

O(k)

DFS, BFS

스택 : 선입 후출 사용메소드) .append() .pop()

큐 : 선입선출 라이브러리 사용 from collection import deque 사용메소드) .append() .popleft()

재귀함수는 종료 조건을 명시해주는 것이 중요하다.

DFS 재귀함수 활용 혹은 스택 자료 구조를 활용한다.

최상단이 계속 변한다. 더 이상 방문 X면 꺼내는 형태

BFS 큐 자료구조 활용

방문하지 않는 노드를 모두 큐에 삽입하고 방문 그림을 그려서 이해하는 것이 편하다.

큐에 넣고 방문처리 -> 큐에서 꺼낸다 -> 인접 노드 모두 큐에 넣고 진행한다.

선택 정렬 알고리즘

가장 작은 데이터를 선택해 맨 앞 데이터와 바꾼다.

arr[i],arr[min\_index]= arr[min\_index],arr[i] #swap문법 같은 형태 사용

시간복잡도 O(N^2)

삽입정렬 -> 적절한 위치에 삽입 하는 방법

시간복잡도 O(N^2) 최선의 경우 O(N)

퀵정렬 -> 기준데이터 Pivot 설정 그 기준보다 큰 데이터와 작은 데이터의 위치를 바꾸는 방법

기본적인 퀵정렬은 첫번째 데이터를 기준으로 잡는 편이다.

왼쪽과 오른쪽이 나뉘어서 왼쪽은 기준데이터보다 작은 것 오른쪽은 기준데이터보다 큰 것이 분류가 된다

이상적인 시간복잡도는 O(NlogN)인데 최악의 경우 O(N^2) 이다.

계수정렬-> 정수형으로 표현 가능할 때만 사용이 가능하지만 속도가 빠르다

최악이여도 시간 복잡도가 O(n+k)이며 테이블을 이용하고 반복을 이용해 최종 출력을 진행한다.

표준 정렬 라이브러리 최악의 경우도 O(NlogN)

총 정리

테이블이(가) 표시된 사진

자동 생성된 설명

버블 정렬 시간 복잡도 O(n^2)

버블 정렬은 앞에서부터 하나씩 비교하면서 크면 넘기고 아니면 제자리에 두는 알고리즘

Merge Sort

쪼갰다가 합치기 그래서 시간 복잡도는 O(nlogn)

**이진탐색**

순차탐색

데이터를 찾기 위해 앞부터 하나씩 확인

시간복잡도 : O(N)

이진탐색

탐색범위를 절반씩 좁혀가며 탐색 (시작점, 끝점, 중간점) 사용

시간 복잡도 : O(logN) 이며 정렬되어 있는 리스트에 적용한다는 것이 포인트

다이나믹프로그래밍

특징 : DP 테이블을 이용

1. 최적부분: 큰 문제를 작게 쪼갤 수 있다.
2. 중복되는 부분문제 -> 동일한 작은 문제를 반복 해결한다.

ex) 피보나치 수열

an= an-1+an-2

top-down 형태

한번 계산한 결과를 메모하는 기법 (메모이제이션) 사용

bottom up 형태

전형적 형태 결과 저장용 리스트(DP테이블)를 만들고 작은 것부터 하나씩 진행하며 진행한 것이라면 표를 확인.

다이내믹프로그래밍과 분할정복의 차이점은 최적부분구조 가질 때 사용

다이내믹은 부분문제의 중복을 계속적으로 본다.

하지만 Pivot 같은 경우 한번 자리를 잡게 되면 그 자리에서 더 이상 작은 문제를 볼 필요가 없다.

다이내믹프로그래밍은 마지막으로 사용하는 것이 좋다.

일단 재귀함수로 구현한뒤 작은 문제의 답이 큰 문제에서 사용되면 코드를 개선하는 방법을 추구한다.

일반적인 코딩테스트에서는 기본 유형이 나오니 기본 유형 숙지!

가장 긴 증가하는 부분 수열(LIS) 알고리즘

D[I] = array[i]를 마지막 원소로 가지는 부분 수열의 길이

점화식 :

모든 0<= j <I 에 대하여 D[i] = max(D[i],D[j]+1 if array[j] <array[i])

**최단 경로 문제**

다익스트라 최단 경로 알고리즘

특정한 노드에서 다른 모든 노드 가는 최단 경로 구함, 테이블 이용

그리디 알고리즘으로도 분류를 한다 : 매 상황에서 가장 비용이 적은 노드를 선택하기 때문에

동작 과정:

1. 출발 노드 설정
2. 최단 거리 테이블을 초기화 #D =[0]\*(n+1)
3. 방문하지 않은 노드들 중 최단 거리가 가장 짧은 노드를 선택
4. 해당 노드를 거쳐 다른 노드로 가는 비용을 계산하여 최단거리 테이블을 갱신
5. 3~4번 반복

이때 최단거리 테이블은 현재까지의 최단거리 정보 보유, 테이블상 작은 것부터 검사하며 같을 경우 노드 번호가 낮은 것부터 검사하는게 국룰

또한 한번 처리된 최단 거리는 고정이다.

구현 : 매 단계마다 가장 작은 것을 구하기 위해 1차원 테이블(순차탐색)을 진행한다.

시간복잡도 O(V^2) 전체 노드 개수가 5000개 이하면 가능하지만 그 이상이면 시간초과

우선순위 큐

우선순위 가장 높은 데이터를 가장 먼저 삭제한다. 보통 표준라이브러리로 지원

힙

최소힙은 값이 낮은 최대힙은 값이 높은것부터 꺼낸다. import heapq

리스트 O(N)

힙(logN)

힙을 이용한 다익스트라 알고리즘은 O(ElogV)

고로 노드가 더 많아져도 처리가 가능함.

플로이드 워셜 알고리즘

모든 노드에서 다른 모든 노드까지 최단경로 모두 계산 (2차원 테이블 이용)

거쳐가는 노드 기준으로 진행하되, 최단 거리 찾는 과정이 아니다.

다이나믹 알고리즘 유형이다. -> 점화식 및 3중포문(O(N^3) )을 이용한다. 그래서 노드수가 적을 때 사용(500개 이하)

ex) 특정한 노드 K를 거쳐가는 경우 확인

a->b->k or a->k 를 검사.

점화식 : Dab = min(Dab,Dak+Dkb)

2차원 테이블 이용 이런형태로…

|  |  |  |  |
| --- | --- | --- | --- |
| \ | 1 | 2 | 3 |
| 1 | 0 |  |  |
| 2 |  | 0 |  |
| 3 |  |  | 0 |

서로소 집합

공통원소가 없는 두집합을 의미

서로소 자료구조

합집합(union) : 두개의 원소가 포함된 집합

찾기(find) : 특정한 원소가 속한 집합 알려주기

국룰: 더 큰 루트 노드가 작은 루트 노드를 가르키게함.

부모테이블을 계속해서 확인하는 방법으로 진행

하지만 편향될경우 O(N)으로 될 수 있어서 문제 -> 경로압축 사용( 재귀호출 사용 , 부모 테이블값 바로 갱신)

ex) 무방향 그래프에서 사이클을 판별할 때 사용된다.

루트 노드가 다르면 두 노드에 대해 합집합 같으면 cycle이 존재하는 것.

최소 신장트리 알고리즘

최소한의 비용으로 그래프의 모든 부분을 포함한다. 사이클이 X 하는 그래프

크루스칼 알고리즘

1. 간선데이터 비용 오름차순 정렬
2. 간선을 하나씩 확인하여 현재의 간선이 사이클을 발생시키는 여부 확인

시간 복잡도 O(ElogE)

위상정렬

사이클이 없는 방향 그래프의 방향성에 거스르지 않도록 순서대로 나열

진입차수: 특정한 노드로 들어오는 간선의 개수

진출차수: 특정한 노드에서 나가는 간선의 개수

큐를 이용

1. 진입차수 0인 모든 노드를 큐에 넣는다.
2. 큐가 빌때까지 반복을 한다.
   1. 큐에서 꺼내 해당 노드에서 나가는 간선을 그래프에서 제거한다.
   2. 새롭게 진입차수가 0이 된 노드를 큐에 넣는다.
   3. 결과 : 큐로 들어온 노드 순서가 위상정렬과 같음

위상정렬은 답이 여러가지 존재가 가능하다.

모든 원소를 방문하기전에 큐가 빈다면 사이클이 존재하는 것으로 처리한다.

O(V+E)

소주 판별 알고리즘

O(x) 일일히 하나씩 확인하면 이렇게 되지만 제곱근을 활용하면 O(N^1/2)까지 줄일 수 있다.

하지만 특정한 범위안에 존재하는 모든 소수는?

에라토스테네스의 체 알고리즘을 사용

1. 2부터 N까지의 모든 자연수 나열
2. 남은 수 중에 아직 처리안된 가장 작은 i를 찾는다.
3. i를 제외한 i의 배수들을 모두 제거한다.

O(NloglogN) 이지만 메모리 사용량이 엄청 높다 별도의 데이블을 이용한다.

투포인터

리스트에 순차적으로 접근해야 할대 두개의 점의 위치를 기록해가면서 처리

ex) 특정한 합을 가지는 부분 연속 수열 찾기

O(N)

1. 시작점과 끝점이 첫번째 원소의 인덱스 0을 가리키도록함.
2. 현재부분합이 M과 같다면 count +1
3. 현재 부분합이 M보다 작으면 end+1
4. 현재의 부분합이 크거나 같으면 start+1
5. 모든 경우를 확인할 때까지 반복

구간 합 문제 ex) 특정 구간의 모든 수를 합한 값을 계산하는 문제

O(N+M)

접두사 합

배열의 맨 앞부터 특정 위치까지의 합을 미리 구해놓는것

P[right]- P[Left-1]

P는 접두사합 저장 되어있는 구조

**자료구조**