<알고리즘 못 푼 문제들 모음>

[백준]

DP :

11053(가장 긴 증가 부분 수열 DP로직 : 이거 다시 풀어봤는데 로직 잘 기억나서 풀어냈다 근데 이 문제 로직 까먹지 않도옥 간간히 다시 한번 봐보자)

1699(가장 적은 항의 개수의 제곱합 : 다시 푼 결과 로직 잘 짜서 품)

2133(3XN 벽 타일 채우기)

2225(합분해 DP용 배열 이용하는 법)

2580(무엇을 caching질 해야 하는지 감조차 안왔다)

2293(무엇을 caching질 해야 하는지 감조차 안왔다 -> 다시봐도 모르겠었음)

9251(LCS문제 아직 풀이법도 이해 못함 : 다시 블로그에 있는 코드 따라 치면서 논리는 이해함 다음에는 처음부터 직접 코드쳐서 구현헤보자)

9252(LCS 2문제인데 9251번 문제에서 길이뿐만이 아닌 해당 부분 수열 또한 같이 출력해줘야 하는 문제)

5582(이거는 Longest Common SubString!! 9251랑 9252번에서 좀 더 바꾸줘야 SubString 구할 수 있음)

11051(이거 이항계수2 문제인데 점화식 알아야 풀 수 있는듯 결국 점화식을 알아야 한다는 건데 일단 넘겼음)

1309(2차원 배열에서 어떠한 경로로 가야 최대값을 얻는 문제가 아니 그냥 가능한 모든 경우의 수를 묻는 문제 DP배열 조차 제대로 짜지 못했다.... : 다시 처음부터 짜서 풀어보니 너무 쉬운 로직이었다. 아직 나는 DP용 머리가 발전되지 않은 것 같다. 분발하자..)

1010(이거 또한 DP배열을 어떻게 짜야하는지 조차 설계 못했다. : 로직 공부하고 다시 처음부터 내가 짜보긴 했지만 까먹을 거 같으니 다음에 또풀어보자)

2293(이거 또한 DP배열을 어떻게 짜야하는지 조차 설계 못했다.)

1915(가장 큰 정사각형의 크기를 구하는 문제인데 접근방식 조차 세우지 못함)

11066('파일 합치기' : 문제 접근법 조차 시도 못함)

11049("11066"번 문제와 동일한 로직을 기본으로 한다 (2개의 부분문제로 나누는...? 유형) : 코드를 좀 더 깔끔하게 짤 필요 있다고 봄 꾸준함 님 처럼)

2096(메모리 함부로 못 쓰게 했던 문제다 다시 한번 풀어보자)

2631(LIS를 이용하면 풀 수 있는 로직인데 이것을 생각해내질 못했다)

1912(최대 연속합을 구하는 문제인데 SW마에스트로 2차 온라인 테스트 1번 문제로 출제됬었는데 로직 바로 기억 안나서 이 문제 코드 배껴서 사용함)

분할정복 :

11729(하노이 탑 이동순서),

1517(Merge Sort를 이용한 Bubble Sort에서의 SWAP 횟수 count),

2263(트리의 순회(다시 풀어봤는데 세부로직 잘못짯다) : 분할정복할때 함수의 parameter에 InOrder와 PostOrder에 대한 인덱스 값 모두를 갖고 deep하게 들어가는 연습필요 + return할 조건문 설정 까다로움),

2261(가장 가까운 두 점(라인 스위핑 알고리즘 필요))

이분탐색 :

1654(랜선 자르기 : 코드는 기억이 나서 잘 짜긴 했는데, 마지막에 출력할 때 left를 출력해야 하는지 right를 출력해야 하는지 step by step 로직이 아직 부족한 상태였음)

2110(어떤 값을 이분탐색 대상으로 해야하는지 로직을 못세웠음 : 다시 풀어봤는데 카카오 겨울 인턴십 코테 모의고사에서 이분탐색 못 써먹은 것에 대해서 받은 충격인지 이분탐색 대상을 잘 선정해서 풀 수 있었다. : 어떤 것을 이분탐색 대상으로 해야 하는지는 이제 괜춘! 그러나! mid값에 따라 조건을 충족하는지 안하는지 구분하는 로직 세우기 까다로움)

1300(무엇이 이분탐색 대상인지도 눈치채지 못했다)

1981번(최솟값 구하기)랑 2805번(최댓값 구하기) 코드를 비교해서 설명해 놓음 : 이분탐색 BFS를 어떻게 조합해서 풀 수 있는지도 상상도 못했었다.ㄴ

※ 한글파일 1981번 문제에서 이분탐색을 이용한 Parametric Search에서 최솟값 (or) 최댓값 return 필요할 때 코드가 어떻게 다른지

삼분탐색 : 11662

완탐 :

1208 : 1182번 문제의 심화버전인데 Input을 둘로 나누고 lower\_bound랑 upper\_bound를 사용해서 편법...? 사용

1451

2186

2632 : 1차원 배열을 원형으로 논리적으로 매핑해서 푸는 건데 연속이라는 문구가 있어서 투포인터를 쓰려고 시도했다. 근데 O(N^2)로도 풀리는 좀 프리한 문제이기도 하고... 그냥 완탐으로 풀자 원형인 자료구조에서 투 포인터 쓰려니까 못하겠다....

16637 : 괄호 추가하기 문제인데 어떻게 재귀함수를 구현해야 하는지 몰랐었다.....(새로운 유형의 재귀함수 구현이라고 생각이 든다 이런 유형의 재귀함수 구현법도 익히자)

1613 : 처음에는 위상정렬로 풀릴 줄 알았으나 모든 사건들에 대한 순서를 주어지는게 아니기 때문에 적용불가 그냥 재귀함수를 통한 DFS로 풀어야 한다. 이 문제에서 언제 위상정렬을 쓰고 안써야 하는지 경각심을 느낄 수 있었다. 위상정렬 문제들을 봐보면 '스페셜 저지'문제라던가 아니면 동시에 발생가능 조건이 있다.

(중위 표기식 --> 후위 표기식) 연산자 우선순위 및 괄호 문제:

16638 :

자료구조 책을 다시 뒤져보며 중위 표기식을 후위 표기식으로 변환하는 방법을 구현해보는 과정을 경험해보았고 후위 표기식을 통해 결과를 연산하는 과정도 겪어봄

이제 이 방법으로 코테에 중위 표기식으로 나오는 수식을 계산하면 좋을 것 같다. 무기가 하나 더 장착된 기분이다.

16639:

이 문제는 중복된 괄호와 한 괄호내에 여러개의 연산자 포함을 허용하므로 실제로는 연산자간의 우선순위를 고려할 필요가 없다.

(이유 : 어차피 모두 괄호로서 구분할 수 있기 때문에...)

그래서 16638번 문제처럼 중위표기식을 후위표기식으로 변환 후 연산결과를 도출할 필요가 없다.

그냥 모든 연산자의 우선순위를 수열을 통해서 설정하고 이 수열의 순서에 따라 1차원 cache배열과 함께 연산결과를 도출해 내면 된다.

이 방법은 내가 생각해낸 것이다.

그리디 : 10610(솔직히 그리디 문제라기보단 3의 배수 판별법을 묻는 문제), 1783, 2873

BFS :

3108(격자 크기를 \*2 + 1해서 늘려서 점과 선분을 구별하기 활용문제)

13549(BFS할 때 For문으로 한 노드에서 다른 노드로 뻗어나가는 거에서도 순서를 고려해서 push해줘야 했음)

2146(간척사업 문제인데 세부 로직 step에서 놓친부분이 있었다 이것 또한 13549문제와 같이 같은 레벨의 노드에서 뻗어나갈때의 문제였다. 다시 구현까지는 하지 말고 내가 해설 써놓은 것 보고 시험 때 주의하도록 글만 다시 보도록 하자)

2206(3차원 visited[][][]배열을 사용해야 하는 다차원 BFS이다 나는 내가 BFS랑 DFS는 마스터 한 줄 알았는데 아직 멀었다. 기초부터 탄탄히 하자)

위상정렬 :

2252 : 위상정렬 처음으로 풀어보았다 queue이용한 풀이 확인

1766 : 그냥 추가 조건에 의해 그냥 queue 대신에 priority\_queue만 쓰면 됨 단! MIN\_HEAP기반의 priority\_queue 사용해야 함

2056, 1516, 1005 : (Dijkstra + 위상정렬) 기법을 혼용하여 풀어냄

트리의 지름 : 1167, 1967(두 문제 모두 dfs두번만 돌리면 되는 알고리즘이다. 임의의 점에서 dfs를 통해 가장 먼 정점을 찾고 그 정점에서 다시 한번 dfs돌려서 풀 수 있다 이 알고리즘은 그냥 외우자!)

투 포인터 : 2003(투 포인터 로직 세부 step 견고하게 처음부터 직접 다시 짜보자)

다익스트라 : 1261(처음에는 이게 다익스트라로 푸는건지도 몰랐었다)

※ 다익스트라 기본 구현은 종만북 말고 그냥 내가 처음부터 구현한 "1753 최단경로" 문제를 기본으로 삼아서 이용하자

Disjoint-Set(Union\_Find) :

1717(이거는 진짜 그냥 Disjoint Set 기본 익히는 용도의 문제로 사용)

[백준 삼성]

나무 재테크(priority\_queue로 구현했지만 시간초과 -> queue로만 구성하여 구현 필요 -> 다시 풀어보니 priority\_queue1개, 그냥 queue 1개로 구성해도 시간안에 돌아간다. 그니까 딱 필요한 에 pq쓰는거는 허용하는거 같다.)

17825\_주사위 윷놀이(진짜 존나 더러운 문제다 문제 시작전에 세부 로직 안짜고 덤비면 디버깅하다 망한다)

[SWEA]

2105 디저트 카페(풀긴 풀었는데 재귀함수 구과정에서 좀 더럽게 짰다.)

5648 원자 소멸 시뮬레이션(vector<int> vs[4001][4001] 이렇게 vector를 너무 큰 것을 잡아놓으면 메모리 터짐)

2838 점심 식사시간(세세한 로직 step에서 놓침)

1767 프로세서 연결하기(dfs할때 아무것도 안하고 그냥 탐색 이어가는 것을 고려해 줄 생각을 못함)

1824 혁진이의 프로그램 검증(음... 못풀겠다)

[종만북 알고스팟]

완탐 : PICNIC, CLOCKSYNC

[기타]

Dijkstra 문제 : 서머코딩윈터코딩(~2018)의 '배달' 문제

MST(Kruskal's Algorithm) 문제 :

서머코딩/윈터코딩(2019)의 ‘지형 이동’ 문제

백준 삼성 A형 17472 '다리 만들기 2' 문제

Trie 문제 :

2020 KAKAO BLIND RECRUITMENT의 '가사 검색' 문제

2018 KAKAO BLIND RECRUITMENT의 '자동완성' 문제(특히 '자동완성'문제에서 각 TrieNode의 멤버변수로 선언한 int cnt를 어떻게 갱신하고 find함수에서 활용하였는지 문자열의 널문자에 기반하여 한글파일에 정리 잘 놓음)

KMP 알고리즘 : 백준 1786 '찾기' 문제

비트마스크 :

2019 KAKAO BLIND RECRUITMENT의 '후보키' 문제

백준 삼성 A형 기출 '17471게리멘더링' 문제(처음에는 내가 비트마스크를 생각해내지 못해서 그냥 재귀함수를 이용해 조합을 구현해서 풀이해서 AC를 받긴 했지만, 비트마스크가 더 코드가 간결하다)

malloc이용해서 트리만들기 :

2019 KAKAO BLIND RECRUITMENT의 '길 찾기 게임' 문제

Union-Find :

2019 카카오 개발자 겨울 인턴십 '호텔 방 배정' 문제

이분탐색 문제 :

2019 카카오 개발자 겨울 인턴십 '징검다리 건너기' 문제(이분탐색으로 풀린다는 것을 알고는 바로 풀었는데 이문제에 이분탐색을 적용할 생각조차 못했었다.)

라인스위핑...? 문제 :

2019 KAKAO BLIND RECRUITMENT의 '추석 트래픽' 문제

[cin, cout 속도 높히기]

ios\_base::sync\_with\_stdio(0);

cin.tie(0);

--> 위에 처럼 동기화를 해제 해 버리면 절대로 scanf(), printf()와 혼용해서 사용하지 말기

cout << endl(X)

cout << "\n"

--> endl써서 AC못 받은 것을 "\n" 써서 AC받을 정도로 속도 차이 존재

[Dijkstra 최단 경로 알고리즘 [VS] MST 최소 신장 트리]

Dijkstra :

단일 시작점 최단 경로 알고리즘으로

시작 정점 S에서부터 다른 정점들까지의 최단 거리 계산

MST :

그래프의 연결성을 그대로 유지하는 가장 '저렴한' 그래프를 찾는 문제에 사용되며

즉, Dijkstra알고리즘에 비해 숲을 바라보듯이 한점에서 다른점까지 최단 경로가 아니라

모든 정점들을 연결하는 경우 중 가장 저렴한 가중치 값을 구하는 케이스에 사용됨

[주의점]

- 비트마스크 사용할 때 "if((i & target) > 0)" 와 같이 각 인덱스의 요소 판별할 때 &연산자를 이용하는데 연산자 우선순위가 관계연산자인 <, <=, >, >=들이 비트단위의 논리연산자인 &보다 높기 때문에

무조건 괄호 붙혀줘야 한다.

- 함수의 반환 타입을 bool로 안했는데 return값을 설정안하고 돌리면 dev-C++에서는 AC되지만 시험 볼때는 런타임에러난다.

- "2573\_빙산" 문제에서 backup[][] 2차원 배열을 별도로 사용해서 원본인 pan[][] 2차원 배열을 업데이트할 때 사용했어야 했는데 놓쳤었다. backup 필요한지 잘 구분하자

[비트마스크, nCr 조합 각각 언제 사용하면 좋을지에 대한 생각]

비트마스크 구현 : nC1, nC2, nC3, ~ nCn 모두를 확인해야 할 때 유용

재귀함수와 vector를 이용한 nCr 조합 구현 : n개중에 특정개수 r개를 뽑아야 하는 경우에 사용하자

아니근데 다시 생각해보니까 그냥 nCr써서도 nC1, nC2, nC3, ~ nCn 모두를 확인해야 할 때 사용 가능

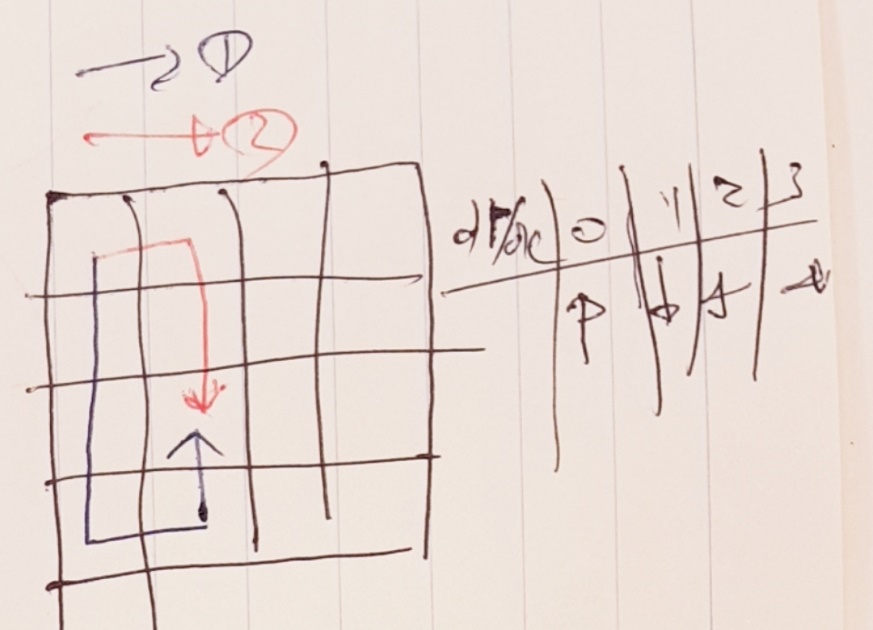
이 둘의 차이점과 각 방법이 어디에 유용한지에 대해 생각해본 결과

비트마스크 : 카카오 기출 "후보키"문제처럼 후보키의 최소성을 만족하기 위한 조합생성에 유리

재귀함수 + nCr : 후보키의 최소성과 같은 고려사항이 없을 경우 유리!

[DFS (VS) BFS for) 최단경로]

가중치가 없는 최단 경로는 무조건 BFS입니다. 왜 DFS가 안 될까요? 그 이유는 당연하게도, 특정 칸에 처음 도달했을 때까지의 경로의 길이가 다른 경로를 통해 도달한 길이보다 짧다는 보장이 전혀 없기 때문입니다.



위와 같이 상,하,좌,우 순서로 탐색을 해서 파란색 경로로 먼저 인덱스가 [2,1]칸에 도착하는데 빨간색 경로로 탐색한 것이 더 빠른 경로인데 파란색 경로로 탐색한것으로만 사용하게 되기 때문!

[priority\_queue]

default는 MAX\_HEAP으로 작동함

MIN\_HEAP으로 작동하게 하려면 아래처럼 하셈

priority\_queue<int, vector<int>, greater<int>> pq;