Algo

전꺼는 다시 보기

//

12 숫자의 총개수(large)

1자리수 2자리수 3자리수 를 10으로 나눈 나머지로 하면 너무 늦으니까, 여기서는 1의자리 9개 두자리수는 90 개인데 여기에 2를 곱하면 되고 세 자릿수는 똑같이 900개 \*3 하면 된다. 이런식으로 변수를 여러 개 둬서 속도를 빠르게함.

///

21번 카드--> 나는 동점일때 while로 뒤에서부터 체크해서 다시 올 동점인 경우와 아닌 경우들로 나눔

강의에서는 변수 하나를 둠 --> 1,2 로 해서 a가 이기면 1 b가 이기면 2 이렇게 함 이게 훨씬 간경

////

22 온도의 최댓값 🡪 for 문 2개로 하는건 너무 느림 // for문 하나로 처음 숫자 개수 만큼은 더하고 그 뒤부터는 앞에꺼 하나씩 빼가면서 밀면서 더해주면됨// 근데 여기서 중요한 것은 첫번째 값들은 먼저 합쳐놓는다.// 이중 for문 일때는 당연히 처음 i는 n-k 개 까지만 더하고// j는 i+k 전까지만 더함

23 연속 부분 증가 수열 🡪 나는 단지 max값고 카운트 개념으로 풀었음// 강의도 비슷한데 생각이 조금 다른게//강의에서는 pre now로 해서 계속해서 pre now 바꿔 가면서 증가 시킴// 나는 max값을 계속 바꾸면서 카운트 값을 증가해서 그걸 다시 max 카운트로 받아서 계산// 강의 방식대로 하면 처음 값을 받아 둬야함// 내가 22 23 풀면서 느낀게 처음 값을 주고 푸는 걸 잘 못하는거 같다.

24. jolly jumpers 🡪 나랑 비슷하게 품 // 근데 뒤에 더해서 뒷부분 result합 n-1 안하고 그냥 다 탐색하면 yes 출력 해도됨// 그리고 둘다 배열로 받을 필요 없이 처음 값을 따로 받고 싶으면 따로 먼저 cin으로 받으면됨// 🡪 헷갈려서 생각을 못함

25 석차구하기(브루트포스) 🡪 강의랑 내가 푼거랑 비슷 근데 나는 그냥 변수 하나로 큰 정수 개수 세서 거기에 1더해주고, 강의는 배열에다가 저장함 배열 2개 만들어서 순위 배열 숫자배열로 계산함//

26 마라톤 🡪 별로 안어렵 강의에서는 기준수 앞에 큰 수들 더하기 1 했고 , 나는 앞에 기준 수보다 작은 수 세서 기준수의 자리에서 빼는걸로 함 // 나중에 병합정렬로 풀면 더 쉬움//

27 n!의 표현법 🡪 이거 그냥 소인수 분해를 직접 어떻게 구현할지 고민 나는 못품//

28번 n!에서 0의 개수 🡪 이거는 위에서 5의 개수만 찾으면 됨 더 쉬움

29번 그냥 3의 개수 중간에 찾아서 풀면됨

30번 이건 lt rt cur 두는걸 생각해야되고 그리고 cur이 3일 때 3보다 작을 때 클 때 경우를 다 나눠서 계산 해야됨 🡪 3 보다 크면 (lt +1) \*10 이고 , 3보다 작으면 lt \*10 이고 3이랑 같으면 (lt \*10) + (rt+1) 임 // 그리고 k =1 로 놓고 10씩 곱해서 증가시킴 자릿수 마다. 그리고 lt rt cur 변수로 둘때는 🡪 lt = n / (k \*10) , cur = (n/k) % 10 , rt= n % k

31 탄소질량 개수🡪//쉬움 문자열로 체크해서 순서대로 풀기

32 선택정렬

33 3등의 점수🡪 내림차순으로 정렬 사용 그 뒤 앞 뒤로 같은지만 비교해서 구함 근데 나는 헷갈려서 그냥 변수들 2개 pre cur 둬서 구함

35 버블정렬 응용🡪 이게 선택 정렬 같은경우는 그냥 멀리 있는애랑도 바꿔서 순서가 바뀌는데, 여기 문제에서는 양수 음수만 구분해서 음수는 왼쪽 양수는 오른쪽으로 보내니까 버블 정렬이 좋음

37 삽입정렬 응용 LRU 🡪 하.. 이거 일단 삽입정렬의 진짜 의미를 제대로 파악 해야겠고, for문 안에서 j써서 먼저 확인하고 그 다음 for문 써서 다시 확인 하는것도 자연스럽게 느껴야됨// 같은 j여도 일단 체크하는데 끝이면 그 뒤 다시 다르게 써도됨 // 이중 for문 구조에서 어떤 역할 하는지 빠르게 구조를 파악 할 필요가 있음 // 밖에 for는 9개 들어갈 숫자들 그 안for문은 열마다 체크 하는 알고리즘이여야됨//

38 inversion sequence 🡪 어렵다 다시 봐야할 듯

39. 배열 합치기(병합정렬 미리공부) 🡪 나는 그냥 vector 함수 써서 쉽게 품 강의에서는 각 두배열 인덱스 값들 처음부터 비교해서 작은놈 먼저 넣고 그배열 자리 옮겨주고 하면됨(이게 맞는듯 나는 너무 파이썬식)

40 교집합 배열 만들기🡪 거의다 풀었는데 강의에서는 정렬을 하고 난뒤 같은거 안같은거 로 조건 나눈게 아니라 정렬을 했으니까 같은거 또는 크거나 작음 으로 해서 a 가 b보다 작으면 걔를 증가시키면서 모든 탐색 마칠 수 있게함

41연속된 자연수의 합 🡪 말이 되냐 이게// 생각 해낼 수 있을지 모르겠음 포인트는 1,2,3,씩 빠지면서 주어진 수에서 2개면 2빼고 3개면 3을 빼서 그걸 다시 나눠준다 각 수에다가.

42이분검색 🡪while문 조건 이랑 mid 계속 갱신 되는거 기억!

43 뮤직비디오 🡪 못품 이분탐색으로 mid값 갱신인데 중요한 것은 여기서 말한 조건을 만족하게 각 합이 mid 보다 적게해서 함수를 따로 만들어서 체크하고 그 숫자를 넘겨주는 방식

44번 마구간 정하기🡪 위랑 비슷 결국에는 이분탐색인데 중요한건 처음 마구간에 말을 놓고 그 그 뒤에k 만큼 차이 보다 크면 거기다가 말을 넣는다.

45 공주 구하기🡪 딱히 어려운건 없는데 변수 놓는거 생각 잘하기

46🡪 멀티 태스킹🡪 이것도 시뮬레이션 문제 45 번이랑 비슷함 둘다 언제 더하고 언제 뺄지 또는 언제 break할지를 잘 생각해야됨

47 봉우리🡪 이차원 배열 탐색 나는 그냥 vector 만들어서 if문 안에 여러 개 놔둬서 품 근데 강의는 dx dy라는 배열 따로 만들어서 3중 루프 만들어서 각 방향마다 2중 for문에서 하나 더 들어가서 4방향 체크하고 있으면 break하는 방식

48 🡪각 행의 평균값과 가까운값 찾기🡪 딱히 어려운 것 없음 이차원 배열 각 행 마다 평균값 들 가지는 일차원 배열 나는 따로 만들어서 비교 강의 에서는 그냥 변수에 안넣고 바로 변수에 넣어서 콘솔 아웃풋 값으로 바로 나오게함

49 🡪 블록의 최댓값 나는 우선 행과 열 에서 기준 행으로 잡고 앞, 우측 에서 본 값들을 각각 일차원 배열에 넣음 // 그 다음 이중 for문으로 행을 i 열을 j로 해서 앞 우측 행렬 값 같으면 이차원 배열에 값 넣고 둘 중에 작은 값 있으면 그때는 작은 값을 넣음 // 강의 에서는 처음에 정면에서 본 값으로 이차원 배열을 j를 행으로 잡고 열을 i로 잡아서 채우고🡪 우측값 배열 따로 받아서 비교 하면서 우측 값보다 큰 값 있으면 우측 값으로 값을 바꾸는 과정 진행함

50,51 은 51은 dP로품

52 ugly numbers 🡪 투포인트 알고리즘인데 여기서는 3포인트

55 기차운행🡪 이거 stack인데 원포인트 변수 두고 품 그냥 숫자 커지면서 예를들어 4 면 1 2 3 4 순으로 들어가야 되니까 무조건 1다음 2 되서 2나올 때까지 ㅇㅇ//

59 부분집합 개수🡪 아 dfs 이용해서 푸는건데 재귀 개념이 더 중요함 // 결국 앞쪽 함수를 쭉 들어갔다가 결론 조건에 도달 하면 처리를 하고 그 뒤 오른쪽으로 이동하는게 결국 그 뒤 함수를 다시 실행 하는건데 위의 과정을 또 거친다고 보면됨// 여기서는 ch라는 배열 사용하여서 체크해가면서 숫자 있을 때 없을 때 구분함//

60 합이 같은 부분집합🡪 재귀함수에 sum값을 주어서 계속적으로 업데이트함// 그래서 total 값 차랑 같은 부분집합 경우 있을 때 yes 나옴 //재귀 나눌때는 숫자 포함 했나 안했나로 구분//

61 특정 수 만들기 🡪 위에 꺼랑 다시 풀기 dfs에 대한 개념 다시 공부 // 간단하게 설명하면 3가지 경우로 나뉘는거임 이진트리아님// 재귀작동원리 더 익숙해질 필요 있음

62 병합정렬🡪

63 인접행렬//

64 경로탐색 🡪 문제는 내가 푼거랑 비슷 근데 실수는 변수 선언에서 나옴// 나는 전역 변수로 두고 계속 재귀 할 때 사용했는데 이렇게 하면 기존에 i값으로 안돌아가고 그냥 i에 더해져서 계속 for문돔 조건보다 큰데 어떻게 가능하냐고 물을 수 있는데 , 재귀 때문에 i가 조건에 걸려 나가게 되면 전역 변수인 i는 커진상태로 다시 들어옴 그래서 계속 큰 i가 찍힘// 그리고 path보여줄때는 그냥 ch배열 1일 때 값들 보여주면됨🡪 강의 에서는 L 레벨을 줘서 레벨마다 i값을 줘서 도달 했을 때 레벨들의 값 보여줌 굳이 이럴필요 없을 듯

65 미로탐색//

68 최소비용 경로🡪 강의에서는 재귀함수에 sum값을 넘겨주어서 계속 업데이트 되고 다시 돌아오면 값 빠지지만 // 나는 sum 변수가 따로 있어서 D갔다가 돌아오면 ch 값 빼주고 sum값 빼줌 같은 원리 같음// 이걸 통해서 알 수 있는게 함수의 인자로 줘버리면 내가 따로 관리할 필요 없을거 같음 , 만약 값이 재귀적으로 동작할 필요가 있다면//

69넓이 우선 탐색//

70 그래프 최단거리

71 송아지 찾기🡪 일단 최단 거리 같은거는 BFS라 생각 하는게 빠름 // 이 문제 포인트는 방향 -1,1,5 를 가지는 배열을 for문으로 더해주면서 나오는 pos값을 가지고 풀어나가는 것/ 또한 이번에는 그냥 ch값에 원래 값 더하기 1 한다 전에 그래프 최단거리처럼 따로 dis배열 없이 ch자체에 구분하면서 더함// 그리고 출력값이 pos가 아니라 x인 이유는 이미 1더해서 시작되서 그렇다.//

72🡪 공주 구하기 큐로 풀기🡪 나랑 차이점은 나는 변수들을 둬서 cntr가 3일 때만 push를 안해주는 걸로 풀었는데// 강의는 아예 큐의 함수로만 품 for문 을 k전 만큼만 돌려서 그때까지는 푸쉬하고 팝하고 k일때는 팝만 한다. 그리고 size가 1일 때 그 값 보여줌

77🡪 교집합 서로소 집합 개념//

78🡪 이건 77에서 배운 find union 개념 이용함 크루스칼 알고리즘(edge 기준) // 포인트는 엣지 기준이니 엣지의 값이 낮은 것 순으로 정렬 해줘야됨 // 그리고 차례로 비교하면서 최소신장 트리 만들어감; // 그리고 엮여 있는지 union 배열 중요

79🡪 이건 우선순위 큐로 최소힙 사용해서 구현(노드 기준) // 포인트는 벡터 만들어 줄 때 make\_pair로 다음노드와 엣지 값 넣어줌 // 당연히 넣어 줄때는 우선순위 큐에 넣는데, 비교값은 그 둘만 받아서 비교함// 그리고 체크 배열 중요 // 그리고 여기서 헷갈렸던데 operator 오버로오 할 때 < 대소 비교연산자 헷갈렸음 // 그냥 정렬사용하기 위해서 오버로딩 할때는 return < 같은 방향이면 오름차순 > 반대면 내림차순인데// 우선순위 큐에 사용하기 위해서는 또 반대로 사용된다. > 이상태가 최소힙이 되고, < 이게 최대힙

80 다익스트라

81 벨만 포드

Bfs , dfs 차이점🡪 지금까지의 생각으로는 처음부터 끝까지 가서 반복되는 코드를 짤 거면 dfs 이고 bfs는 일단 한 부분에서 끝까지 다돌고 다음거 넘어가는식// 여기서 섬 아일랜드 개수 구하기 문제가 bfs의 특징을 보여주었음// **근데 섬아일랜드에서 dfs쓸려면 잘 생각해보면 for문 여러 번 돌아야 되는데 dfs안에서 하면 성립이 안됨 그러면 main 함수에서 이중for문 돌려서 그 뒤 dfs에서는 8가지 방향으로 dfs 다시 반복 해서 다돌때까지 하면 main에서 count 개수를 세어주니까 지도 처음부터 차례대로 돌면서 1이뜨면 끝까지 들어가고 나와서 개수 세고 하는 식으로 똑같이 구할 수 있다.**

**동**적프로그래밍🡪 최대 부분 증가수열 🡪 응용 해보면 선을 긋는 문제도 같은 방식 // 여기서 최대로 선을 긋는 개수를 물어 보는게 아니라// 전체 연결 된 선에서 몇 개를 제거하면 최대 연결이 되는 가를 물어보면 🡪 그냥 전체 개수 n에서 최대 연결된 수를 빼면 된다.

벽돌 쌓기 문제🡪 기준이 여러 개여도 푸는 법은 똑같음 여러 개중 하나를 기준으로 잡고 정렬 시킨다음에 정렬되지 않은 기준으로 최대 개수를 쌓으면 됨

11번 최대 점수 구하기 knapsack문제는 다른 문제들이랑 다르게 한 유형을 중복되게 사용 할 수 없음 그래서 유형별로 1개씩만 사용가능해서 2차원 배열로 dy행렬을 만들어야됨(dy[I] = dy[i-1][j] +val 이런식임 왜냐면 중복되서 들어가면 안되니까 전꺼 행의 값을 사용// 근데 이렇게 하면 메모리 너무 많이 잡아먹으니까 // 원래 dy테이블 만드는 순서의 반대로 뒤에서부터 만들면 중복안되계 계산할 수 있음

플로이드 와샬 🡪 모든 노드의 모든 거리 구하는 건데 최소를 구할 수가 있음// dy[i][j] = dy[i][k]+dy[k][j] 이렇게 해서 k가 기준으로 3단 루프 를 만들어서 이차원 행렬 돈다.

회장 뽑기 문제에서는 이렇게 해서 행마다 최대값들을 뽑아서 점수로 만들고 그중에 최솟값 가진 애들이 회장후보임

위상정렬 🡪 기존 그래프랑 다르게 진입차수로 dy행렬 만들어서 queue를 이용해서 품