|  |
| --- |
| **[증빙] 문제풀이 활동 내용** |





|  |
| --- |
|  |

1. 20210101 (금)
2. 백준 - 10867. 중복빼고 정렬하기

N개의 정수가 주어진다. 이때, N개의 정수를 오름차순으로 정렬하는 프로그램을 작성하시오. 같은 정수는 한 번만 출력한다. 첫째 줄에 수의 개수 N (1 ≤ N ≤ 100,000)이 주어진다. 둘째에는 숫자가 주어진다. 이 수는 절댓값이 1,000보다 작거나 같은 정수이다. 첫째 줄에 수를 오름차순으로 정렬한 결과를 출력한다. 이때, 같은 수는 한 번만 출력한다.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 박민정, 백준

✔ 풀이과정

|  |
| --- |
| // 빅민정님 코드  #include <iostream>  #include <vector>  #include <set>  **using** **namespace** std**;**  int main**()**  **{**  int num**,** input**;**  vector**<**int**>** arr**;**  set**<**int**>** answer**;**  cin **>>** num**;**  **for** **(**int i **=** 0**;** i **<** num**;** i**++)**  **{**  cin **>>** input**;**  arr**.**push\_back**(**input**);**  **}**  **for** **(**int i **:** arr**)**  **{**  answer**.**insert**(**i**);**  **}**  **for** **(**int i **:** answer**)**  **{**  cout **<<** i **<<** " "**;**  **}**  cout **<<** endl**;**  **return** 0**;**  **}** |

1. 백준 – 1431. 시리얼 번호

전화번호부에 적힌 전화번호를 담은 배열 phone\_book이 solution함수의 매개변수로 주어질 때, 어떤 번호가 다른 번호의 접두어인 경우가 있으면 false, 그렇지 않으면 true를 return 하도록 solution함수를 작성하세요. (단, phone\_book의 길이는 1이상 1,000,000 이하이고, 각 전화번호의 길이는 1 이상 20 이하입니다.)

다솜이는 기타를 많이 가지고 있다. 그리고 각각의 기타는 모두 다른 시리얼 번호를 가지고 있다. 다솜이는 기타를 빨리 찾아서 빨리 사람들에게 연주해주기 위해서 기타를 시리얼 번호 순서대로 정렬하고자 한다. 모든 시리얼 번호는 알파벳 대문자 (A-Z)와 숫자 (0-9)로 이루어져 있다. 시리얼번호 A가 시리얼번호 B의 앞에 오는 경우는 다음과 같다.

1. A와 B의 길이가 다르면, 짧은 것이 먼저 온다.

2. 만약 서로 길이가 같다면, A의 모든 자리수의 합과 B의 모든 자리수의 합을 비교해서 작은 합을 가지는 것이 먼저온다. (숫자인 것만 더한다)

3. 만약 1,2번 둘 조건으로도 비교할 수 없으면, 사전순으로 비교한다. 숫자가 알파벳보다 사전순으로 작다.

시리얼이 주어졌을 때, 정렬해서 출력하는 프로그램을 작성하시오.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 정보라, 백준

✔ 풀이과정

|  |
| --- |
| // 정보라님 코드  #include <iostream>  #include <queue>  #include <string>  **using** **namespace** std**;**  struct cmp **{**  bool is\_int**(**char a**)** **{**  int code **=** **(**int**)**a**;**  **return** a **>=** 48 **&&** a **<=** 57**;** // 아스키코드  **}**  // true면 a와b가 위치를 swap  bool **operator()(**string a**,** string b**)** **{**  **if** **(**a**.**length**()** **!=** b**.**length**())** // 길이가 다르면 짧은 것이 먼저온다.  **return** a**.**length**()** **>** b**.**length**();**  // 길이가 같다면  int sumA **=** 0**,** sumB **=** 0**;**  **for** **(**int i **=** 0**;** i **<** a**.**length**();** i**++){**  **if** **(**is\_int**(**a**[**i**]))** sumA **+=** **(**a**[**i**]** **-** '0'**);**  **if** **(**is\_int**(**b**[**i**]))** sumB **+=** **(**b**[**i**]** **-** '0'**);**  **}**  **if** **(**sumA **!=** sumB**)**  **return** sumA **>** sumB**;** // 각 자리수의 숫자의 합이 작은 것이 먼저온다  // 숫자의 합이 같다면 사전순  **for** **(**int i **=** 0**;** i **<** a**.**length**();** i**++)** **{**  **return** **(**int**)**a**[**i**]** **>** **(**int**)**b**[**i**];**  **}**  **return** **true;**  **}**  **};**  int main**()**  **{**  int n**;** cin **>>** n**;**  string**\*** a **=** **new** string**[**n**];**  **for** **(**int i **=** 0**;** i **<** n**;** i**++)**  cin **>>** a**[**i**];**  priority\_queue**<**string**,** vector**<**string**>,** cmp**>** myQ**;**  **for** **(**int i **=** 0**;** i **<** n**;** i**++)**  myQ**.**push**(**a**[**i**]);**  **while** **(!**myQ**.**empty**())**  **{**  cout **<<** myQ**.**top**()** **<<** endl**;**  myQ**.**pop**();**  **}**  **}** |

1. 백준 – 16401. 과자 나눠주기

명절이 되면, 홍익이 집에는 조카들이 놀러 온다.  떼를 쓰는 조카들을 달래기 위해 홍익이는 막대 과자를 하나씩 나눠준다. 조카들이 과자를 먹는 동안은 떼를 쓰지 않기 때문에, 홍익이는 조카들에게 최대한 긴 과자를 나눠주려고 한다. 그런데 나눠준 과자의 길이가 하나라도 다르면 조카끼리 싸움이 일어난다. 따라서 반드시 모든 조카에게 같은 길이의 막대 과자를 나눠주어야 한다. M명의 조카가 있고 N개의 과자가 있을 때, 조카 1명에게 줄 수 있는 막대 과자의 최대 길이를 구하라.단, 막대 과자는 길이와 상관없이 여러 조각으로 나눠질 수 있지만, 과자를 하나로 합칠 수는 없다. 단, 막대 과자의 길이는 양의 정수여야 한다.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 서예진, 백준

✔ 풀이과정

|  |
| --- |
| // 서예진님 코드  #include <iostream>  #include <vector>  #include <algorithm>  #include <string>  #include <sstream>  **using** **namespace** std**;**  int main**()** **{**  int Pnum**,** Snum**;**  cin **>>** Pnum**;**  cin **>>** Snum**;**  vector**<**int**>** snackArr**(**Snum**);**  **for(**int i **=** 0**;** i **<** Snum**;** i**++){**  cin **>>** snackArr**[**i**];**  **}**    sort**(**snackArr**.**begin**(),** snackArr**.**end**());**    int start **=** 0**,** end **=** Snum**-**1**,** mid**;**  int ans**,** cnt**;**  // cout << snackArr[Snum-1]<< endl;  **while(**start **<=** end**){**  mid **=** **(**start**+**end**)/**2**;**  cnt **=** 0**;**  **for(**int i **=** 0**;** i **<** Snum**;** i**++){**  cnt **+=** snackArr**[**i**]/**mid**;**  //해당 과자를 mid로 나눈후의 몫 (받을수 있는 사람수)  **}**  **if(**cnt **>** Pnum**){** //사람수보다 나눠준 과자가 더 많음.  start **=** mid **+**1**;** //더 긴 과자를 나눠줄수 있는지 확인.  ans **=** mid**;**  **}**  **else** **{** // 사람수보다 나눠준 과자가 적음.  end **=** mid **-**1**;** //더 짧은 과자를 나눠줘야 함.  **}**  **}**  cout**<<** ans **<<** endl**;**  **}** |

1. SWEA – 1204. 최빈수 구하기

어느 고등학교에서 실시한 1000명의 수학 성적을 토대로 통계 자료를 만들려고 한다. 이때, 이 학교에서는 최빈수를 이용하여 학생들의 평균 수준을 짐작하는데, 여기서 최빈수는 특정 자료에서 가장 여러 번 나타나는 값을 의미한다.  
다음과 같은 수 분포가 있으면,  
10, 8, 7, 2, 2, 4, 8, 8, 8, 9, 5, 5, 3  
최빈수는 8이 된다.  
최빈수를 출력하는 프로그램을 작성하여라 (단, 최빈수가 여러 개 일 때에는 가장 큰 점수를 출력하라).

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 삼성 Expert Academy

✔ 풀이과정

|  |
| --- |
| // 차준영님 코드  #include<iostream>  **using** **namespace** std**;**  int main**(**int argc**,** char**\*\*** argv**)**  **{**  int test\_case**;**  int T**;**  cin**>>**T**;**  **for(**test\_case **=** 1**;** test\_case **<=** T**;** **++**test\_case**)**  **{**  int i**,** m**,** score**,** scores**[**101**];**  **for(**i**=**0**;** i**<=**100**;** i**++)** **{**  scores**[**i**]=**0**;**  **}**    scanf**(**"%d"**,** **&**m**);**  **for(**i**=**1**;** i**<=**1000**;** i**++)** **{**  scanf**(**"%d"**,** **&**score**);**  scores**[**score**]++;**  **}**    int max**=**0**,** idx**;**  **for(**i**=**0**;** i**<=**100**;** i**++)** **{**  **if(**scores**[**i**]** **>=** max**)** **{**  max **=** scores**[**i**];**  idx **=** i**;**  **}**  **}**    printf**(**"#%d %d\n"**,** test\_case**,** idx**);**  **}**  **return** 0**;**//정상종료시 반드시 0을 리턴해야합니다.  **}** |

1. 20210108 (금)
2. BOJ – 1037. 약수

첫째 줄에 N의 진짜 약수의 개수가 주어진다. 이 개수는 50보다 작거나 같은 자연수이다. 둘째 줄에는 N의 진짜 약수가 주어진다. 1,000,000보다 작거나 같고, 2보다 크거나 같은 자연수이고, 중복되지 않는다.

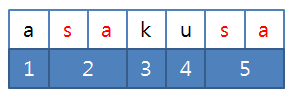
🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 정보라, 백준

✔ 풀이과정

|  |
| --- |
| // 정보라님 코드  #include <iostream>  #include <vector>  #include <algorithm>  **using** **namespace** std**;**  int main**()**  **{**  int size**;**  scanf\_s**(**"%d"**,** **&**size**);**  vector**<**int**>** factors**(**size**);**  **for** **(**int i **=** 0**;** i **<** size**;** i**++)**  scanf\_s**(**"%d"**,** **&**factors**[**i**]);**  sort**(**factors**.**begin**(),** factors**.**end**());**  int n **=** factors**[**0**]** **\*** factors**[**size **-** 1**];**  printf**(**"%d"**,** n**);**  **}** |

1. SWEA – 3143. 가장 빠른 문자열 타이핑

그냥 한 글자씩 타이핑 한다면 A의 길이만큼 키를 눌러야 할 것이다. 여기에 속도를 조금 더 높이기 위해 어떤 문자열 B가 저장되어 있어서 키를 한번 누른 것으로 B전체를 타이핑 할 수 있다. 이미 타이핑 한 문자를 지우는 것은 불가능하다. 예를 들어 A=”asakusa”, B=”sa”일 때, 다음 그림과 같이 B를 두 번 사용하면 5번 만에 A를 타이핑 할 수 있다.  


A와 B가 주어질 때 A 전체를 타이핑 하기 위해 키를 눌러야 하는 횟수의 최솟값을 구하여라.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 서예진, 삼성 Expert Academy

✔ 풀이과정

|  |
| --- |
| // 서예진님 코드  #include<iostream>  #include <string>  #include <algorithm>  **using** **namespace** std**;**  int main**(**int argc**,** char**\*\*** argv**)**  **{**  int test\_case**;**  int T**;**  cin**>>**T**;**    **for(**test\_case **=** 1**;** test\_case **<=** T**;** **++**test\_case**)**  **{**  int ans**,** cnt **=** 0**;**  string mystr**,** slicestr**;**  cin **>>** mystr **>>** slicestr**;**  int slicesize **=** slicestr**.**size**();**    **for(**int k **=** 0**;** k **<** mystr**.**size**();** k**++){**  **if(**mystr**.**substr**(**k**,**slicesize**)** **==** slicestr**)** **{**  cnt**++;**  k**+=** **(**slicesize**-**1**);**  **}**  **}**  ans **=** mystr**.**size**()** **-** **(**cnt**\***slicesize**)** **+** cnt**;**  cout **<<** "#" **<<** test\_case **<<** " "**<<** ans **<<** endl**;**  **}**  **return** 0**;**//정상종료시 반드시 0을 리턴해야합니다.  **}** |

1. BOJ – 2504. 괄호의 값

4개의 기호 ‘(’, ‘)’, ‘[’, ‘]’를 이용해서 만들어지는 괄호열 중에서 올바른 괄호열이란 다음과 같이 정의된다.

한 쌍의 괄호로만 이루어진 ‘()’와 ‘[]’는 올바른 괄호열이다.

만일 X가 올바른 괄호열이면 ‘(X)’이나 ‘[X]’도 모두 올바른 괄호열이 된다.

X와 Y 모두 올바른 괄호열이라면 이들을 결합한 XY도 올바른 괄호열이 된다.

예를 들어 ‘(()[[]])([])’ 의 괄호값을 구해보자.  ‘()[[]]’ 의 괄호값이 2 + 3×3=11 이므로  ‘(()[[ ]])’의 괄호값은 2×11=22 이다. 그리고  ‘([])’의 값은 2×3=6 이므로 전체 괄호열의 값은 22 + 6 = 28 이다. 주어진 괄호열을 읽고 그 괄호값을 앞에서 정의한대로 계산하여 출력하라.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 차준영, 백준

✔ 풀이과정

|  |
| --- |
| // 차준영님 코드  // 괄호의 결합관계로 묶여 있는 수식을 길게 풀어서 생각하는 것이 관건.  // ex) (() [[]]) ([]) = 2\*(2 + 3\*3) + 2\*3 = 2\*2 + 2\*3\*3 + 2\*3  // 1-1. stack에 (를 추가해야 하는 경우, tmp에 2를 곱함  // 1-2. stack에 [를 추가해야 하는 경우, tmp에 3을 곱함  // 2. stack에서 빼와야 하는 경우 (닫힌 괄호)  // 2-1. stack에 남은게 없으면 error  // 2-2. ()로 matching되는 경우, tmp로 쌓아올린 값을 res에 모두 더한다.  // 2-2-1. 괄호가 바로 matching 되었기 때문에 이후 들어오는 값은 matching된 괄호와 따로 생각해야 한다.  // 2-2-2. '('괄호가 들어오면서 2를 곱했기 때문에, matching된 이후에는 2로 나눠준다.  // 2-3. 2-2와 동일하다. []로 matching되는 경우, res에 모두 더한다.  // 2-3-1. 다만 []로 matching되지 않고 ]]이거나 )]인 경우, 결과값을 모두 더한 상태이므로 3으로 나눠주기만 한다.  // 2-4. 괄호가 matching 되지 않으면 error  // 3. stack이 비어있으면 완성된 괄호이므로 결과값을 출력한다.  #include <iostream>  #include <vector>  #include <stack>  **using** **namespace** std**;**  char a**[**31**];**  int main**()** **{**  stack**<**char**>** s**;**  scanf**(**"%s"**,** **&**a**);**  int res **=** 0**,** i**,** tmp **=** 1**;**  **for** **(**i **=** 0**;** a**[**i**]** **!=** '\0'**;** i**++)** **{**  **if** **(**a**[**i**]** **==** '('**)** **{**  tmp **\*=** 2**;**  s**.**push**(**a**[**i**]);**  **}**  **else** **if** **(**a**[**i**]** **==** '['**)** **{**  tmp **\*=** 3**;**  s**.**push**(**a**[**i**]);**  **}**    **else** **{**  **if** **(**s**.**empty**())** **{**  printf**(**"0"**);**  **return** 0**;**  **}**  **if** **(**s**.**top**()** **==** '(' **&&** a**[**i**]** **==** ')'**)** **{**  **if** **(**a**[**i **-** 1**]** **==** '('**)** res **+=** tmp**;**  tmp **/=** 2**;**  s**.**pop**();**  **}**  **else** **if** **(**s**.**top**()** **==** '[' **&&** a**[**i**]** **==** ']'**)** **{**  **if** **(**a**[**i **-** 1**]** **==** '['**)** res **+=** tmp**;**  tmp **/=** 3**;**  s**.**pop**();**  **}**  **else** **{**  printf**(**"0"**);**  **return** 0**;**  **}**  **}**  **}**    **if(!**s**.**empty**())** printf**(**"0"**);**  **else** printf**(**"%d"**,** res**);**  **return** 0**;**  **}** |

1. 백준 – 14716. 현수막

저번 학기 영상처리 수업을 듣고 배웠던 지식을 최대한 응용 해보고 싶은 혁진이는 이 현수막에서 글자가 몇 개인지 알아보는 프로그램을 만들려 한다. 혁진이는 우선 현수막에서 글자인 부분은 1, 글자가 아닌 부분은 0으로 바꾸는 필터를 적용하여 값을 만드는데 성공했다. 그런데 혁진이는 이 값을 바탕으로 글자인 부분 1이 상, 하, 좌, 우, 대각선으로 인접하여 서로 연결되어 있다면 한 개의 글자라고 생각만 하였다. 혁진이가 필터를 적용하여 만든 값이 입력으로 주어졌을 때, 혁진이의 생각대로 프로그램을 구현하면 글자의 개수가 몇 개인지 출력하여라.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 박민정, 백준

✔ 풀이과정

|  |
| --- |
| // 박민정님 코드  #include <iostream>  #include <vector>  **using** **namespace** std**;**  // 행, 열, 현수막 정보, 글자갯수  int col**,** row**,** val**,** cnt **=** 0**;**  int dy**[**8**]** **=** **{** **-**1**,** **-**1**,** **-**1**,** 0**,** 0**,** 1**,** 1**,** 1 **};** // y축(col) 변화량  int dx**[**8**]** **=** **{** **-**1**,** 0**,** 1**,** **-**1**,** 1**,** **-**1**,** 0**,** 1 **};** // x축(row) 변화량  int flag**[**255**][**255**]** **=** **{** 0**,** **};**  void DFS**(**int fcol**,** int frow**);**  int main**()**  **{**  /\* 입력값 넣기 \*/  cin **>>** col**;**  cin **>>** row**;**  **for** **(**int i **=** 0**;** i **<** col**;** i**++)**  **{**  **for** **(**int j **=** 0**;** j **<** row**;** j**++)**  **{**  cin **>>** val**;**  flag**[**i**][**j**]** **=** val**;**  **}**  **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  **for** **(**int i **=** 0**;** i **<** col**;** i**++)**  **{**  **for** **(**int j **=** 0**;** j **<** col**;** j**++)**  **{**  **if** **(**flag**[**i**][**j**]** **==** 1**)**  **{**  DFS**(**i**,** j**);**  cnt**++;**  **}**  **}**  **}**  cout **<<** cnt**;**  **}**  void DFS**(**int fcol**,** int frow**)**  **{**  flag**[**fcol**][**frow**]** **=** 0**;**  **for** **(**int i **=** 0**;** i **<** 8**;** i**++)**  **{**  int ncol **=** fcol **+** dy**[**i**];**  int nrow **=** frow **+** dx**[**i**];**  **if** **(**ncol **>=** 0 **&&** nrow **>=** 0 **&&** ncol **<** col **&&** nrow **<** row **&&** flag**[**ncol**][**nrow**]==**1**)**  DFS**(**ncol**,** nrow**);**  **}**  **}** |

1. 20210115 (금)
2. SWEA – 5521. 상원이의 생일파티

상원이의 생일 파티가 곧 열린다! 그렇기에 상원이는 반 친구들에게 생일 파티 초대장을 주려고 한다. 그러나 파티가 어색하게 되는 것을 원치 않는 상원이는 모든 친구들에게 초대장을 줄 생각이 없다. 같은 반에 있지만, 서로 친한 친구가 아닐 수도 있기 때문이다. 상원이는 우선 자신과 친한 친구들에게는 모두 초대장을 주기로 했다. 여기에 더해서 친한 친구의 친한 친구인 경우에도 초대장을 주기로 했다. 총 몇 명의 친구들에게 초대장을 주어야 하는지 구하는 프로그램을 작성하라. 상원이의 반에는 N명이 있으며, 각 학생들은 1번에서 N번까지의 번호가 붙어 있다. 여기서 1번 학생이 상원이다.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 서예진, 삼성 Expert Academy

✔ 풀이과정

먼저 배열 array를 commands에서 제시한 i와 j의 위치에 맞게 잘라 임시 벡터에 저장한다. 임시 벡터를 오름차순으로 정렬한 후, k번째 위치에 있는 수를 answer에 넣는다. 제시한 조건에 크게 제약을 받을 필요가 없다.

|  |
| --- |
| // 서예진님 코드  #include <iostream>  #include <vector>  #include <set>  **using** **namespace** std**;**  int main**()** **{**  int n**,** m**,** cnt **=** 0**;**  cin**>>** n **>>** m **;**    vector**<**pair**<**int**,**int**>>** vec**(**m**);**  set**<**int**>** finalfriend**;**  vector**<**int**>** myfriend**;**  **for(**int i **=** 0**;** i **<** m **;** i**++){**  cin **>>** vec**[**i**].**first**>>** vec**[**i**].**second **;**  **if(**vec**[**i**].**first **==** 1**)** **{**  myfriend**.**push\_back**(**vec**[**i**].**second**);** //상원이 친한친구목록 X  finalfriend**.**insert**(**vec**[**i**].**second**);** //초대친구목록 X  **}**  **else** **if(**vec**[**i**].**second **==** 1**)** **{**  myfriend**.**push\_back**(**vec**[**i**].**first**);** //상원이 친한친구목록 X  finalfriend**.**insert**(**vec**[**i**].**first**);** //초대친구목록 X  **}**  **}**  **if(**myfriend**.**size**()** **!=**0 **){** //상원이가 초대장 줄 친구가 있음.  **for(**int i **=** 0**;** i **<** myfriend**.**size**();** i**++){**  **for(**int j **=** 0**;** j **<** m**;** j**++){**  **if(**myfriend**[**i**]** **==** vec**[**j**].**first **&&** 1**!=**vec**[**j**].**second**){**  finalfriend**.**insert**(**vec**[**j**].**second**);**  **}**  **else** **if(**myfriend**[**i**]==** vec**[**j**].**second **&&** 1**!=**vec**[**j**].**first**){**  finalfriend**.**insert**(**vec**[**j**].**first**);**  **}**  **}**  **}**  cnt **=** finalfriend**.**size**();**  **}**  cout **<<** cnt **<<** endl **<<** endl**;**  set**<**int**>::**iterator iter**;**  **for(**iter **=** finalfriend**.**begin**();** iter **!=** finalfriend**.**end**();** iter**++){**  cout **<<** **\***iter **<<** " " **;**  **}**  cout **<<** endl**;**  **}** |

1. 백준 - 9372, 상근이의 여행

첫 번째 줄에는 테스트 케이스의 수 T(T ≤ 100)가 주어지고, 각 테스트 케이스마다 다음과 같은 정보가 주어진다.

첫 번째 줄에는 국가의 수 N(2 ≤ N ≤ 1 000)과 비행기의 종류 M(1 ≤ M ≤ 10 000) 가 주어진다. 이후 M개의 줄에 a와 b 쌍들이 입력된다. a와 b를 왕복하는 비행기가 있다는 것을 의미한다. (1 ≤ a, b ≤ n; a ≠ b) 주어지는 비행 스케줄은 항상 연결 그래프를 이룬다. 상근이가 모든 국가를 여행하기 위해 타야 하는 비행기 종류의 최소 개수를 출력한다.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 차준영, 백준

✔ 풀이과정

이 문제의 가장 핵심 포인트는 많은 수들을 정렬할 때, 시간초과가 나지 않도록 유의해야 한다는 점이다. 문제를 읽어봤을 때 당장 떠올릴 수 있는 방법으로는 순열을 이용해 가능한 모든 경우의 수를 만들고 가장 큰 것을 찾아내는 방법을 떠올릴 수 있지만, 100,000개나 되는 수의 모든 경우의 수를 모두 구하려면 적어도 100,000 factorial번 계산해야 한다. 따라서 우리는 이 시간 복잡도를 줄이는 방향으로 계산을 하는 또 다른 방법을 생각해봐야 한다.

더하기(+)연산을 통해 a와 b가 그대로 ab로 이어 붙여진다고 생각해보자. ab와 ba를 비교해봤을 때 ba가 더 크게 나왔다면 numbers 배열 안에서 a와 b의 순서를 b, a로 하는 것이 합리적이다. 이 과정을 numbers의 배열을 정렬하기 위한 특별한 규칙으로 만들어 정렬한 후, 배열의 순서대로 그대로 이어 붙이면 된다.

|  |
| --- |
| // 차준영 님 코드  #include <iostream>  #include <vector>  #include <algorithm>  **using** **namespace** std**;**  // Union - Find (합집합 찾기)  // https://blog.naver.com/PostView.nhn?blogId=ndb796&logNo=221230967614&parentCategoryNo=128&categoryNo=&viewDate=&isShowPopularPosts=false&from=postList  // 부모 노드를 찾는 함수  int getParent**(**int parent**[],** int x**)** **{**  **if(**parent**[**x**]** **==** x**)** **return** x**;**  **return** parent**[**x**]** **=** getParent**(**parent**,** parent**[**x**]);**  **}**  // 두 부모 노드를 합치는 함수  int unionParent**(**int parent**[],** int a**,** int b**)** **{**  a **=** getParent**(**parent**,** a**);**  b **=** getParent**(**parent**,** b**);**  **if(**a **<** b**)** **return** parent**[**b**]** **=** a**;**  **else** **return** parent**[**a**]** **=** b**;**  **}**  // 같은 부모를 가지는지 확인  int findParent**(**int parent**[],** int a**,** int b**)** **{**  a **=** getParent**(**parent**,** a**);**  b **=** getParent**(**parent**,** b**);**  **if** **(**a**==**b**)** **return** 1**;**  **else** **return** 0**;**  **}**  // 간선 클래스 선언 (정점 2개와 간선의 가중치 정보 포함)  class Edge **{**  public**:**  int node**[**2**];**  int distance**;**  Edge**(**int a**,** int b**,** int distance**)** **{**  **this->**node**[**0**]** **=** a**;**  **this->**node**[**1**]** **=** b**;**  **this->**distance **=** distance**;**  **}**    bool **operator** **<(**Edge**&** edge**)** **{**  **return** **this->**distance **<** edge**.**distance**;**  **}**  **};**  int main**()** **{**  int test\_case**;**  int T**;**  cin**>>**T**;**  **for(**test\_case **=** 1**;** test\_case **<=** T**;** **++**test\_case**)**  **{**  int i**,** n**,** m**,** p**,** q**;**  scanf**(**"%d %d"**,** **&**n**,** **&**m**);**    vector**<**Edge**>** v**;**  v**.**clear**();**    // 간선 입력 받고 정렬  **for(**i**=**0**;** i**<**m**;** i**++)** **{**  scanf**(**"%d %d"**,** **&**p**,** **&**q**);**  v**.**push\_back**(**Edge**(**p**,** q**,** 1**));**  **}**    // 간선 비용을 기준으로 오름차순 정렬  // sort(v.begin(), v.end());  // 문제 상 가중치가 모두 1이기 때문에 오름차순으로 정렬할 필요가 없음    // 각 정점이 포함된 그래프가 어디인지 저장  int parent**[**n**];**  **for(**i**=**0**;** i**<**n**;** i**++)** **{**  parent**[**i**]** **=** i**;** // 기본적으로 모든 정점은 자기 자신을 가리킴  **}**    int sum**=**0**;**  **for(**int i**=**0**;** i**<**v**.**size**();** i**++)** **{**  // 싸이클이 발생하지 않는 경우 그래프에 포함시킴함  // 두 노드가 서로 연결되어 있지 않은 경우  **if(!**findParent**(**parent**,** v**[**i**].**node**[**0**]-**1**,** v**[**i**].**node**[**1**]** **-** 1**))** **{**  sum **+=** v**[**i**].**distance**;**  // 두 노드를 합치면서 가중치 더함  unionParent**(**parent**,** v**[**i**].**node**[**0**]-**1**,** v**[**i**].**node**[**1**]-**1**);**  **}**  **}**    printf**(**"%d\n"**,** sum**);**  **}**  **return** 0**;**  **}** |

1. SWEA - 8457, 알 덴테 스파게티

가게에는 N개의 모래시계가 있고, i번째 모래시계는 정확히 xi초를 측정할 수 있다.모래가 다 내려오면 바로 모래시계를 뒤집는 것으로 2xi초, 3xi초, …도 측정할 수 있다. 그러나 나머지 시간은 측정할 수 없다. 선표는 모래시계로는 정확한 시간을 측정할 수 없을 것 같아서, 조금 타협하여 E초 정도의 오차를 허용하려고 한다. 그러므로, 어떤 하나의 모래시계를 구입하여 B – E초에서 B + E초 중 하나를 측정해낼 수 있으면 된다. 구입해도 되는 모래시계의 개수를 구하는 프로그램을 작성하라.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 박민정, 삼성 Expert Academy

✔ 풀이과정

citiations을 먼저 오름차순으로 정렬한다. 정렬 한 결과를 생각해봤을 때, i번째 index에서 최소한으로 인용되어야 하는 횟수는 citations 배열 크기에서 i를 뺀 값과 같다. 예를 들어, 어느 과학자가 쓴 논문의 개수가 5개라고 생각하고 논문을 인용한 횟수 순으로 정렬했다면, index가 2에 위치해 있는 논문은 5 – 2번 즉, 적어도 3번 이상 인용되어야 한다. 이를 코드를 통해 나타내면 다음과 같다.

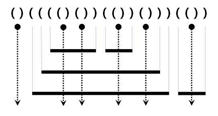
|  |
| --- |
| // 박민정 님 코드  #include <iostream>  **using** **namespace** std**;**  bool recursion**(**int init\_sand**,** int target**,** int mult\_sand**);**  int main**()**  **{**  int test\_case**;**  cin **>>** test\_case**;**    **for** **(**int tc **=** 1**;** tc **<=** test\_case**;** tc**++)**  **{**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  int sandNum **=** 0**;** // 모래시계 갯수  int b **=** 0**;** // 정확한 값  int e **=** 0**;** // 오차  cin **>>** sandNum **>>** b **>>** e**;**  int cnt **=** 0**;** // count  int sand**[**100**];** // 상점에 있는 모래시계 수  **for** **(**int i **=** 0**;** i **<** sandNum**;** i**++)**  **{**  cin **>>** sand**[**i**];**  **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  **for** **(**int i **=** 0**;** i **<** sandNum**;** i**++)**  **{**  int target **=** b **-** e**;** // 오차 범위 내 수  **for** **(**int j **=** 0**;** j **<** e **\*** 2 **+** 1**;** j**++)**  **{**  **if** **(**recursion**(**sand**[**i**],** target**,** sand**[**i**]))**  **{**  cnt**++;**  **break;**  **}**  **else**  **{**  target**++;**  **}**  **}**  **}**  cout **<<** '#' **<<** tc **<<**" " **<<** cnt **<<** endl**;**  **}**  **return** 0**;**  **}**  bool recursion**(**int init\_sand**,** int target**,** int mult\_sand**)**  **{**  **if** **(**mult\_sand **>** target**)**  **return** **false;**  **else** **if** **(**mult\_sand **==** target**)**  **return** **true;**  **else**  recursion**(**init\_sand**,** target**,** mult\_sand **+** init\_sand**);**  **}** |

1. 20210129 (금)

완전 탐색은 모든 경우의 수를 전부 찾아서 답을 찾는 알고리즘으로 다른 말로는 Brute-Force, 브루트 포스라고도 부른다. 모든 자료를 탐색해야 하기 때문에 특정 구조를 전체적으로 탐색할 수 있는 방법을 필요로 하며, 대표적으로 선형 구조를 탐색하는데 사용되는 순차 탐색, 비선형 구조를 탐색하는데 사용되는 깊이 우선 탐색(DFS), 너비 우선 탐색(BFS) 등이 있다. 주로 BFS가 완전탐색과 관련이 있으며, DFS는 back-tracking과 관련이 있다.

1. SWEA – 5432. 쇠막대기 자르기

여러 개의 쇠막대기를 레이저로 절단하려고 한다. 효율적인 작업을 위해서 쇠막대기를 아래에서 위로 겹쳐 놓고, 레이저를 위에서 수직으로 발사하여 쇠막대기들을 자른다. 수평으로 그려진 굵은 실선은 쇠막대기이고, 점은 레이저의 위치, 수직으로 그려진 점선 화살표는 레이저의 발사 방향이다.



이러한 레이저와 쇠막대기의 배치는 다음과 같이 괄호를 이용하여 왼쪽부터 순서대로 표현할 수 있다.  
    1. 레이저는 여는 괄호와 닫는 괄호의 인접한 쌍 “()” 으로 표현된다. 또한, 모든 “()”는 반드시 레이저를 표현한다.  
    2. 쇠막대기의 왼쪽 끝은 여는 괄호 ‘(’ 로, 오른쪽 끝은 닫힌 괄호 ‘)’ 로 표현된다.  
 쇠막대기와 레이저의 배치를 나타내는 괄호 표현이 주어졌을 때, 잘려진 쇠막대기 조각의 총 개수를 구하는 프로그램을 작성하라.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 서예진, 삼성 Expert Academy

✔ 풀이과정

Answers의 길이만큼 돌면서 하나씩 문제와 각 사람들이 찍은 문제들의 답이 일치하는지 확인해 봐야 한다. 한 사람 씩 정답 개수를 count 해주고, 가장 많이 정답을 맞춘 사람을 찾아 반환하고자 하는 벡터에 넣어준다

|  |
| --- |
| // 서예진님 코드  #include<iostream>  #include <vector>  #include <string>  #include <stack>  **using** **namespace** std**;**  int main**(**int argc**,** char**\*\*** argv**)**  **{**  ios\_base**::**sync\_with\_stdio**(false);**  cin**.**tie**(NULL);** cout**.**tie**(NULL);**  int test\_case**;**  int T**;**  cin**>>**T**;**  /\*  여러 개의 테스트 케이스가 주어지므로, 각각을 처리합니다.  \*/  **for(**test\_case **=** 1**;** test\_case **<=** T**;** **++**test\_case**)**  **{**  int cnt**=**0**;**  string mystr**;**  cin **>>** mystr**;**  stack**<**int**>** S**;**  **for(**int i **=** 0**;** i **<** mystr**.**size**();** i**++){**  string tmp**,** ttmp**;**  tmp **=** mystr**[**i**];**  ttmp **=** mystr**[**i**-**1**];**    **if(**tmp **==** "("**){**  S**.**push**(**i**);** // 막대기의 시작 위치를 저장  **}**  **else** **{**  **if(**ttmp **==** "(" **){** // () 연결된거는 자르는 부분임.  S**.**pop**();** // ( 를 빼냈음. == 레이저 끊어냈음.  cnt **+=** S**.**size**();** //막대기 갯수만큼 더함.  **}**  **else{** //막대기 부분임.  S**.**pop**();** //막대기 꺼냈고,  cnt**++;** //맨끝조각 더함.  **}**  **}**  **}**  cout **<<** "#" **<<** test\_case **<<** " "**<<** cnt **<<** "\n"**;**  **}**  **return** 0**;**//정상종료시 반드시 0을 리턴해야합니다.  **}** |

1. 백준 - 1655, 가운데를 말해요

수빈이는 동생에게 "가운데를 말해요" 게임을 가르쳐주고 있다. 수빈이가 정수를 하나씩 외칠때마다 동생은 지금까지 수빈이가 말한 수 중에서 중간값을 말해야 한다. 만약, 그동안 수빈이가 외친 수의 개수가 짝수개라면 중간에 있는 두 수 중에서 작은 수를 말해야 한다.

예를 들어 수빈이가 동생에게 1, 5, 2, 10, -99, 7, 5를 순서대로 외쳤다고 하면, 동생은 1, 1, 2, 2, 2, 2, 5를 차례대로 말해야 한다. 수빈이가 외치는 수가 주어졌을 때, 동생이 말해야 하는 수를 구하는 프로그램을 작성하시오.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 정보라, 백준

✔ 풀이과정

|  |
| --- |
| // 정보라 님 코드  #include<iostream>  #include<queue>  **using** **namespace** std**;**  int main**()** **{**  int n**;**  cin **>>** n**;**  priority\_queue**<**int**>** smaller**;** // 중간값보다 작은 숫자들 저장 , Max Heap  priority\_queue**<**int**,** vector**<**int**>,** greater**<**int**>>** bigger**;** // 중간값보다 큰 숫자들 저장 , Min Heap  int number**;**  scanf**(**"%d"**,** **&**number**);**  smaller**.**push**(**number**);**  cout **<<** number **<<** endl**;**  **for** **(**int i **=** 1**;** i **<** n**;** i**++)** **{**  scanf**(**"%d"**,** **&**number**);**  **if** **(**number **<** smaller**.**top**())** smaller**.**push**(**number**);**  **else** bigger**.**push**(**number**);**  **if** **(**smaller**.**size**()** **<** bigger**.**size**())** **{**  smaller**.**push**(**bigger**.**top**());**  bigger**.**pop**();**  **}**  **else** **if** **(**smaller**.**size**()** **>** bigger**.**size**()** **+** 1**)** **{**  bigger**.**push**(**smaller**.**top**());**  smaller**.**pop**();**  **}**  printf**(**"%d\n"**,** smaller**.**top**());**  **}**  **}** |

1. 백준 - 2263, 트리의 순회

n개의 정점을 갖는 이진 트리의 정점에 1부터 n까지의 번호가 중복 없이 매겨져있다. 이와 같은 이진 트리의 인오더와 포스트오더가 주어졌을 때, 프리오더를 구하는 프로그램을 작성하시오.

첫째 줄에 n(1≤n≤100,000)이 주어진다. 다음 줄에는 인오더를 나타내는 n개의 자연수가 주어지고, 그 다음 줄에는 같은 식으로 포스트오더가 주어진다. 첫째 줄에 프리오더를 출력한다.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 차준영, 백준

✔ 풀이과정

노란색 타일은 중앙에 몰려있다고 했으므로, 노란색 타일이 우선 어떤 식으로 분포되어 있는지 확인해봐야 한다. 노란색 타일 개수의 약수를 구하면 노란색 타일이 배치되어 있는 가로와 세로의 길이가 나오고, 노란색 타일을 감싸고 있는 형태로 갈색 타일이 배치되어 있으므로 주어진 갈색 타일의 개수와 노란색 타일의 세로와 가로의 길이를 이용해 구한 갈색 타일의 개수가 동일한지 확인하는 과정을 통해 전체 카펫의 가로와 세로의 길이를 구할 수 있다.

|  |
| --- |
| // 차준영 님 코드  #include <iostream>  #include <vector>  #include <queue>  **using** **namespace** std**;**  int n**;**  vector**<**int**>** inorder**(**100000**);**  vector**<**int**>** postorder**(**100000**);**  void preorder**(**int in\_st**,** int in\_en**,** int po\_st**,** int po\_en**)** **{**  // 종료 조건 (index 범위를 벗어났을 때)  **if** **(**in\_st **<** 0 **||** po\_st **<** 0 **||** in\_en **>=** n **||** po\_en **>=** n**)** **return;**  **if** **(**in\_st **>** in\_en **||** po\_st **>** po\_en**)** **return;**  printf**(**"%d "**,** postorder**[**po\_en**]);**  int index **=** **-**1**;**  **for** **(**int i **=** in\_st**;** i **<=** in\_en**;** i**++)** **{**  **if** **(**inorder**[**i**]** **==** postorder**[**po\_en**])**  index **=** i**;**  **}**    // 종료 조건 (찾고자 하는 원소가 없는 경우)  **if(**index **==** **-**1**)** **return;**  // 왼쪽 subtree  preorder**(**in\_st**,** index **-** 1**,** po\_st**,** po\_st **+** **(**index **-** in\_st**)** **-** 1**);**  // 오른쪽 subtree  preorder**(**index **+** 1**,** in\_en**,** po\_st **+** **(**index **-** in\_st**),** po\_en **-** 1**);**  **}**  int main**()** **{**  int i**;**  scanf\_s**(**"%d"**,** **&**n**);**  **for** **(**i **=** 0**;** i **<** n**;** i**++)** scanf\_s**(**"%d"**,** **&**inorder**[**i**]);**  **for** **(**i **=** 0**;** i **<** n**;** i**++)** scanf\_s**(**"%d"**,** **&**postorder**[**i**]);**  preorder**(**0**,** n**-**1**,** 0**,** n**-**1**);**  **return** 0**;**  **}** |

1. SWEA - 1859, 백만 장자 프로젝트

25년 간의 수행 끝에 원재는 미래를 보는 능력을 갖게 되었다. 이 능력으로 원재는 사재기를 하려고 한다.  
다만 당국의 감시가 심해 한 번에 많은 양을 사재기 할 수 없다.  
다음과 같은 조건 하에서 사재기를 하여 최대한의 이득을 얻도록 도와주자.  
    1. 원재는 연속된 N일 동안의 물건의 매매가를 예측하여 알고 있다.  
    2. 당국의 감시망에 걸리지 않기 위해 하루에 최대 1만큼 구입할 수 있다.  
    3. 판매는 얼마든지 할 수 있다.  
예를 들어 3일 동안의 매매가가 1, 2, 3 이라면 처음 두 날에 원료를 구매하여 마지막 날에 팔면 3의 이익을 얻을 수 있다.

첫 번째 줄에 테스트 케이스의 수 T가 주어진다.  
각 테스트 케이스 별로 첫 줄에는 자연수 N(2 ≤ N ≤ 1,000,000)이 주어지고,  
둘째 줄에는 각 날의 매매가를 나타내는 N개의 자연수들이 공백으로 구분되어 순서대로 주어진다. (각 날의 매매가는 10,000이하이다.)

🧑 문제풀이자 : 전원

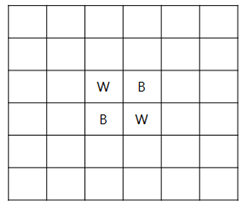
👨‍💻 문제선정자 및 출처 : 박민정, 삼성 Expert Academy

✔ 풀이과정

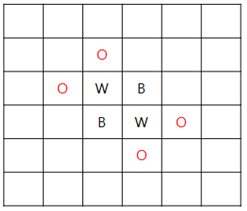
|  |
| --- |
| // 박민정님 코드  #include <iostream>  #include <vector>  **using** **namespace** std**;**  int main**()**  **{**  ios\_base**::**sync\_with\_stdio**(false);**  cin**.**tie**(NULL);** cout**.**tie**(NULL);**    int T**;**  cin **>>** T**;**  **for** **(**int tc **=** 0**;** tc **<** T**;** tc**++)**  **{**  vector**<**int**>** arr**(**1000000**)** **;**  int n **=** 0**;**  cin **>>** n**;**  //vector<int> arr(n) ;  int ans **=** 0**;**    **for** **(**int i **=** 0**;** i **<** n**;** i**++)**  **{**  cin **>>** arr**[**i**];**  **}**  int flag **=** arr**[**n **-** 1**];**  **for** **(**int i **=** n **-** 2**;** i **>=** 0**;** i**--)**  **{**  **if** **(**flag **>** arr**[**i**])**  ans **+=** flag **-** arr**[**i**];**    **else**  flag **=** arr**[**i**];**  **}**    cout **<<** "#" **<<** tc **+** 1 **<<** " " **<<** ans**<<**endl**;**  **}**  **return** 0**;**  **}** |

1. 20210204 (목)
2. SWEA - 4615, 오셀로 게임

오셀로라는 게임은 흑돌과 백돌을 가진 사람이 번갈아가며 보드에 돌을 놓아서 최종적으로 보드에 자신의 돌이 많은 사람이 이기는 게임이다.  
보드는 4x4, 6x6, 8x8(가로, 세로 길이) 크기를 사용한다. 6x6 보드에서 게임을 할 때, 처음에 플레이어는 다음과 같이 돌을 놓고 시작한다(B : 흑돌, W : 백돌).  
4x4, 8x8 보드에서도 동일하게 정가운데에 아래와 같이 배치하고 시작한다.



그리고 흑, 백이 번갈아가며 돌을 놓는다. 처음엔 흑부터 시작하는데 이 때 흑이 돌을 놓을 수 있는 곳은 다음과 같이 4군데이다.



🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 서예진, 삼성 Expert Academy

✔ 풀이과정

<queue> 라이브러리의 우선순위 큐를 이용하여 scoville의 값을 기준으로 오름차순 정렬하여 해결했다.

|  |
| --- |
| // 서예진님 코드  #include<iostream>  #include <vector>  **using** **namespace** std**;**  int main**(**int argc**,** char**\*\*** argv**)**  **{**  ios\_base**::**sync\_with\_stdio**(false);**  cin**.**tie**(NULL);** cout**.**tie**(NULL);**  int test\_case**;**  int T**;**  cin**>>**T**;**  /\*  여러 개의 테스트 케이스가 주어지므로, 각각을 처리합니다.  \*/  **for(**test\_case **=** 1**;** test\_case **<=** T**;** **++**test\_case**)**  **{**  int num**;**  cin **>>** num**;** //사각형길이  int len**;** //입력할 돌 갯수  cin **>>** len**;**  vector**<**vector**<**int**>>** ans**(**num**,**vector**<**int**>(**num**,**0**));**// (0,0)  //초기화 과정  ans**[**num**/**2**][**num**/**2**-**1**]** **=** 1**;** //검정 1  ans**[**num**/**2**-**1**][**num**/**2**]** **=** 1**;**  ans**[**num**/**2**-**1**][**num**/**2**-**1**]** **=** 2**;** //흰색 2  ans**[**num**/**2**][**num**/**2**]** **=** 2**;**    //돌 주변으로 확인할 곳 8칸  int dx**[]** **=** **{-**1**,**0**,**1**,-**1**,**1**,-**1**,**0**,**1**};**  int dy**[]** **=** **{-**1**,-**1**,-**1**,**0**,**0**,**1**,**1**,**1**};**    **for(**int i **=** 0**;** i **<** len**;** i**++){**  int tmp1**,** tmp2**,** tmp3**;**  cin **>>** tmp2 **>>** tmp1 **>>** tmp3**;**  tmp1 **-=** 1**;**  tmp2 **-=** 1**;**    **if(**tmp3 **==** 1**)** **{** //검정 1  //일단 그자리에 검정돌을 놓음.  ans**[**tmp1**][**tmp2**]** **=** 1**;**  //주변 자리의 경우의 수를 따져야 함.    **for(**int i **=** 0**;** i **<** 8**;**i**++){**  **if((**tmp1**+**dx**[**i**]>=**0**)&&(**tmp1**+**dx**[**i**]<**num**)&&(**tmp2**+**dy**[**i**]>=**0**)&&(**tmp2**+**dy**[**i**]<**num**)){**  //정사각형 범위안에 들어와야 함.  vector**<**pair**<**int**,**int**>>** tmpvec**;**  **if(**ans**[**tmp1**+**dx**[**i**]][**tmp2**+**dy**[**i**]]** **==** 2**){** //해당 칸에 놓여진 돌이 흰색이면, (검정을 만나야 흰->검 가능)  tmpvec**.**push\_back**(**make\_pair**(**tmp1**+**dx**[**i**],** tmp2**+**dy**[**i**]));**  int k **=** 1**;**  **while(**1**){**  //옆으로 계속 가면서 같은 색 돌인지 확인하면서, tmpvec에 저장.  k**++;**  **if((**tmp1**+**dx**[**i**]\***k**>=**0**)&&(**tmp1**+**dx**[**i**]\***k**<**num**)&&(**tmp2**+**dy**[**i**]\***k**>=**0**)&&(**tmp2**+**dy**[**i**]\***k**<**num**)){**  **if(**ans**[**tmp1**+**dx**[**i**]\***k**][**tmp2**+**dy**[**i**]\***k**]** **==** 2**)** tmpvec**.**push\_back**(**make\_pair**(**tmp1**+**dx**[**i**]\***k**,** tmp2**+**dy**[**i**]\***k**));** //흰색만나면 임시벡터에 저장  //빈칸이면  **else** **if(**ans**[**tmp1**+**dx**[**i**]\***k**][**tmp2**+**dy**[**i**]\***k**]** **==** 1**)** **{** //검정을 만나면  **for(**int j **=** 1**;** j **<=** k**;** j**++){**  ans**[**tmp1**+**dx**[**i**]\***j**][**tmp2**+**dy**[**i**]\***j**]** **=** 1**;** //저장한데까지 검정으로 바꿈  **}**  **break;** //검정 만났으면 탈출해라.  **}**  **else** **if** **(**ans**[**tmp1**+**dx**[**i**]\***k**][**tmp2**+**dy**[**i**]\***k**]** **==** 0**)** **break;**  **}**  **else{**  //범위 벗어나면 break 해라.  **break;**  **}**  **}**  **}**  **}**  **}**  **}**    **else** **if(**tmp3 **==** 2**)** **{** //흰색 2  ans**[**tmp1**][**tmp2**]** **=** 2**;**  //주변 자리의 경우의 수를 따져야 함.    **for(**int i **=** 0**;** i **<** 8**;**i**++){**  **if((**tmp1**+**dx**[**i**]>=**0**)&&(**tmp1**+**dx**[**i**]<**num**)&&(**tmp2**+**dy**[**i**]>=**0**)&&(**tmp2**+**dy**[**i**]<**num**)){**  //정사각형 범위안에 들어와야 함.  vector**<**pair**<**int**,**int**>>** tmpvec**;**  **if(**ans**[**tmp1**+**dx**[**i**]][**tmp2**+**dy**[**i**]]** **==** 1**){** //해당 칸에 놓여진 돌이 검정이면, (흰색을 만나야 검->흰 가능)  tmpvec**.**push\_back**(**make\_pair**(**tmp1**+**dx**[**i**],** tmp2**+**dy**[**i**]));**  int k **=** 1**;**  **while(**1**){**  //옆으로 계속 가면서 같은 색 돌인지 확인하면서, tmpvec에 저장.  k**++;**  **if((**tmp1**+**dx**[**i**]\***k**>=**0**)&&(**tmp1**+**dx**[**i**]\***k**<**num**)&&(**tmp2**+**dy**[**i**]\***k**>=**0**)&&(**tmp2**+**dy**[**i**]\***k**<**num**)){**  **if(**ans**[**tmp1**+**dx**[**i**]\***k**][**tmp2**+**dy**[**i**]\***k**]** **==** 1**)** tmpvec**.**push\_back**(**make\_pair**(**tmp1**+**dx**[**i**]\***k**,** tmp2**+**dy**[**i**]\***k**));** //검정만나면 임시벡터에 저장  **else** **if(**ans**[**tmp1**+**dx**[**i**]\***k**][**tmp2**+**dy**[**i**]\***k**]** **==** 2**)** **{** //흰색을 만나면  **for(**int j **=** 1**;** j **<=** k**;** j**++){**  ans**[**tmp1**+**dx**[**i**]\***j**][**tmp2**+**dy**[**i**]\***j**]** **=** 2**;** //저장한데까지 검정으로 바꿈  **}**  **break;** //흰색 만났으면 탈출해라.    **}**  **else** **if** **(**ans**[**tmp1**+**dx**[**i**]\***k**][**tmp2**+**dy**[**i**]\***k**]** **==** 0**)** **break;**  **}**  **else{**  //범위 벗어나면 break 해라.  **break;**  **}**  **}**  **}**  **}**  **}**  **}**  **}**  int b**=**0**,** w**=**0**;**  //이제 전체에서 검정이랑 흰색의 개수를 세면 됨.  **for(**int i **=** 0**;** i **<** num**;** i**++){**  **for(**int j **=** 0**;** j **<** num**;** j**++){**  **if(**ans**[**i**][**j**]** **==** 1**)** b**+=**1**;**  **if(**ans**[**i**][**j**]** **==** 2**)** w**+=**1**;**  **}**  **}**  cout **<<** "#" **<<** test\_case **<<**" "**<<**b **<<** " " **<<** w **<<** endl**;**  **}**  **return** 0**;**//정상종료시 반드시 0을 리턴해야합니다.  **}** |

1. 백준 – 1080. 행렬

0과 1로만 이루어진 행렬 A와 행렬 B가 있다. 이때, 행렬 A를 행렬 B로 바꾸는데 필요한 연산의 횟수의 최솟값을 구하는 프로그램을 작성하시오.

행렬을 변환하는 연산은 어떤 3\*3크기의 부분 행렬에 있는 모든 원소를 뒤집는 것이다. (0 -> 1, 1 -> 0)

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 정보라, 백준

✔ 풀이과정

|  |
| --- |
| //정보라님의 코드  #include <iostream>  #include <algorithm>  #include <vector>  #include <queue>  #include <string>  **using** **namespace** std**;**  int main**()** **{**  // 행렬 입력받기  int n**,** m**,** i**,** j**,** p**,** q**;**  scanf\_s**(**"%d %d"**,** **&**n**,** **&**m**);**  string temp**;**  int before**[**51**][**51**];**  int after**[**51**][**51**];**  **for** **(**i **=** 0**;** i **<** n**;** i**++)** **{**  cin **>>** temp**;**  **for** **(**j **=** 0**;** j **<** m**;** j**++)**  before**[**i**][**j**]** **=** temp**[**j**]** **-** '0'**;**  **}**  **for** **(**i **=** 0**;** i **<** n**;** i**++)** **{**  cin **>>** temp**;**  **for** **(**j **=** 0**;** j **<** m**;** j**++)**  after**[**i**][**j**]** **=** temp**[**j**]** **-** '0'**;**  **}**  // 연산 -> 가지고 있는 행렬과 바꿔야하는 행렬을 비교했을 때  int res **=** 0**;**  **for** **(**i **=** 0**;** i **<** n**-**2**;** i**++)** **{**  **for** **(**j **=** 0**;** j **<** m**-**2**;** j**++)** **{**  // 중심점을 기준으로 좌상단 값이 다르다면 행렬 뒤집기  **if** **(**before**[**i**][**j**]** **!=** after**[**i**][**j**])** **{**  res**++;**  **for** **(**p **=** 0**;** p **<** 3**;** p**++)** **{**  **for** **(**q **=** 0**;** q **<** 3**;** q**++)** **{**  before**[**i **+** p**][**j **+** q**]** **=** 1 **-** before**[**i **+** p**][**j **+** q**];**  **}**  **}**  **}**  **}**  **}**  // 모든 연산이 끝났을 때 before과 after이 일치하지 않으면 -1 출력  **for** **(**i **=** 0**;** i **<** n**;** i**++)** **{**  **for** **(**j **=** 0**;** j **<** m**;** j**++)** **{**  **if** **(**before**[**i**][**j**]** **!=** after**[**i**][**j**])** **{**  printf**(**"-1"**);**  **return** 0**;**  **}**  **}**  **}**  printf**(**"%d"**,** res**);**  **return** 0**;**  **}** |

1. 백준 - 2206, 벽 부수고 이동하기

N×M의 행렬로 표현되는 맵이 있다. 맵에서 0은 이동할 수 있는 곳을 나타내고, 1은 이동할 수 없는 벽이 있는 곳을 나타낸다. 당신은 (1, 1)에서 (N, M)의 위치까지 이동하려 하는데, 이때 최단 경로로 이동하려 한다. 최단경로는 맵에서 가장 적은 개수의 칸을 지나는 경로를 말하는데, 이때 시작하는 칸과 끝나는 칸도 포함해서 센다. 만약에 이동하는 도중에 한 개의 벽을 부수고 이동하는 것이 좀 더 경로가 짧아진다면, 벽을 한 개 까지 부수고 이동하여도 된다.

한 칸에서 이동할 수 있는 칸은 상하좌우로 인접한 칸이다. 맵이 주어졌을 때, 최단 경로를 구해 내는 프로그램을 작성하시오.

🧑 문제풀이자 : 전원

👨‍💻 문제선정자 및 출처 : 차준영, 백준

✔ 풀이과정

|  |
| --- |
| //차준영님의 코드  #include <iostream>  #include <algorithm>  #include <vector>  #include <queue>  #include <string>  **using** **namespace** std**;**  int dx**[**4**]** **=** **{** **-**1**,** 0**,** 1**,** 0 **};**  int dy**[**4**]** **=** **{** 0**,** 1**,** 0**,** **-**1 **};**  int main**()** **{**  int n**,** m**,** i**,** j**;**  scanf\_s**(**"%d %d"**,** **&**n**,** **&**m**);**  string temp**;**  int map**[**1001**][**1001**];**  **for** **(**i **=** 0**;** i **<** n**;** i**++)** **{**  cin **>>** temp**;**  **for** **(**j **=** 0**;** j **<** m**;** j**++)**  map**[**i**][**j**]** **=** temp**[**j**]** **-** '0'**;**  **}**  // 방문여부 배열 초기화 ([x][y][0]은 벽 안 부순 상태에서 방문, [x][y][1]은 벽을 부순 상태에서 방문)  int check**[**1001**][**1001**][**2**]** **=** **{** 0 **};**  check**[**0**][**0**][**0**]** **=** 1**;**  // ((x, y), (res, bw)) 상태로 전달 (★ 현재 상태에서 벽을 부쉈는지 여부를 전역변수로 초기화 한다면 관리가 어려워짐)  queue**<**pair**<**pair**<**int**,** int**>,** pair**<**int**,** int**>>>** loc**;**  loc**.**push**(**make\_pair**(**make\_pair**(**0**,** 0**),** make\_pair**(**1**,** 0**)));**    **while** **(!**loc**.**empty**())** **{**  int x **=** loc**.**front**().**first**.**first**;**  int y **=** loc**.**front**().**first**.**second**;**  int res **=** loc**.**front**().**second**.**first**;**  int bw **=** loc**.**front**().**second**.**second**;**  loc**.**pop**();**  // 도착지점에 도달  **if** **(**x **==** n **-** 1 **&&** y **==** m **-** 1**)** **{**  printf**(**"%d"**,** res**);**  **return** 0**;**  **}**  // 사방진을 돌면서 너비우선탐색  **for** **(**i **=** 0**;** i **<** 4**;** i**++)** **{**  int next\_x **=** x **+** dx**[**i**];**  int next\_y **=** y **+** dy**[**i**];**  // 배열의 범위를 벗어나면 pass  **if** **(**next\_x **<** 0 **||** next\_y **<** 0 **||** next\_x **>=** n **||** next\_y **>=** m**)** **continue;**  **else** **{**  // 벽이 가로막고 있지만 한 번도 부순 적이 없는 경우, 그리고 한 번도 방문하지 않았을 때  // 벽이 가로막고 있을 때 한 번 부순 적이 있으면 pass  **if** **(**map**[**next\_x**][**next\_y**]** **==** 1 **&&** bw **==** 0 **&&** check**[**next\_x**][**next\_y**][**bw**]** **==** 0**)** **{**  check**[**next\_x**][**next\_y**][**1**]** **=** 1**;** // 방문 체크  loc**.**push**(**make\_pair**(**make\_pair**(**next\_x**,** next\_y**),** make\_pair**(**res **+** 1**,** bw **+** 1**)));**  **}**  // 길인 경우, ★그리고 한 번도 방문하지 않은 경우  **if** **(**map**[**next\_x**][**next\_y**]** **==** 0 **&&** check**[**next\_x**][**next\_y**][**bw**]** **==** 0**)** **{**  check**[**next\_x**][**next\_y**][**bw**]** **=** 1**;** // 방문 체크(0과 1 구분이 중요함)  loc**.**push**(**make\_pair**(**make\_pair**(**next\_x**,** next\_y**),** make\_pair**(**res **+** 1**,** bw**)));**  **}**  **}**  **}**  **}**  // 큐가 완전히 비었는데도 목표지점에 도달하지 못했으면 -1  printf**(**"-1"**);**  **return** 0**;**  **}** |