1. 세모왕 김세모

시간 제한 : 1초 | 메모리 제한 : 128MB

해설

직각 삼각자를 만들 수 있는 조건은 피타고라스의 법칙을 성립하는 것이다.

따라서 주어진 세 변의 길이가 피타고라스의 법칙을 만족하는지 안하는지만 판단하면 된다.

주의할 점은 **세 변의 길이가 반드시 오름차순으로 주어진다는 보장이 없다**. 따라서 가장 긴 변의 길이를 찾아낼 수 있어야 한다.

게다가 세 변의 길이가 10^9미만, 즉 int형으로 저장할 수 있는 값이지만, 피타고라스의 법칙을 만족하는지 확인하는 과정에서 int 값의 범위를 넘어갈 수 있다. 예를 들어 세 변의 길이를 담아둘 변수를 int형으로 선언하고 $30000, 40000, 50000$로 입력을 받았을 때, $30000^2 + 40000^2, 50000^2$를 계산하는 과정 중 오버플로우가 발생한다. 그러므로 자료형 long long을 사용해야 한다.

2. 소공이의 씨꾸릿-★ 뽀올더

시간 제한 : 1초 | 메모리 제한 : 256MB

해설

각 "파일의 비밀번호"는 "폴더 이름"과 그 "파일 이름"의 최소공배수다.

먼저 "폴더 이름"과 "파일 이름"의 최대공약수를 구한 다음, 이를 이용해 최소공배수를 구할 수 있다.

최대공약수를 구하는 알고리즘으로 유클리드 호제법을 사용할 수 있다.

주의할 점은 **"폴더 이름"과 "파일 이름"이 int 범위에 들어간다 해도 최소공배수가 이를 초과할 수 있다**는 것이다. 이러한 연산 도중 오버플로우 발생을 방지하기 위해 long long 형으로 변수를 선언해주자.

step1. 유클리드 호제법을 통해 두 수 A와 B의 최대 공약수 $GCD(A, B)$를 구함

step2. $LCM(A, B) = A \times B / GCD(A, B)$을 통해 정답을 구하면 된다. ($LCM(A, B)$: A, B의 최소 공배수)

+추가설명

두 수 A 와 B 의 최대공약수를 G 라 하자.

$$G \overline{)\, A \quad B} \\ \quad\; a \quad b$$

$A = G \times a$

$B = G \times b$ (a 와 b는 서로소이다.)

두 수 A, B의 최소 공배수 $L = a \times b \times G$ 이다.

$A \times B = a \times b \times G \times G = L \times G$ 이다.

$$\therefore L = A \times B / G$$

유클리드 호제법의 시간 복잡도는 대략 $O(\log(\min(a, b)))$이고, 파일이 N개이므로

총 시간 복잡도는$O(N\log(\min(a, b)))$정도이다.

3. 강박증에 걸린 박팰린

시간 제한 : 2초 | 메모리 제한 : 128MB

해설

주어진 문자열에서 서로 다른 위치의 두 문자를 골라서 서로 바꿨을 때 팰린드롬 판별을 해본다.

팰린드롬이 맞다면 두 문자의 위치를 출력한다.

팰린드롬으로 바꾸는 게 불가능하다면 -1를 출력한다.

브루트 포스(Brute Force)를 통해 이중 반복문으로 모든 경우의 수를 확인할 수 있다.

주의해야 할 점은 반드시 한 번은 바꿔야함을 잊지 말아야한다. 그리고 애초에 팰린드롬 문장이 입력으로 주어진다면 가장 앞에 있는 단어와 가장 뒤에 있는 단어를 바꾸면 되므로 답은 {1, (문장의 길이)}가 된다.

```
1.  cin >> input; // 문자열 입력.
2.    int len = input.length();
3.    for(int i=0;i<len;i++){
4.      for(int j=i+1;j<len;j++){ // 2중 반복문으로 브루트 포스.
5.        string cmp = input;
6.        cmp.replace(i, 1, input.substr(j, 1));
7.        cmp.replace(j, 1, input.substr(i, 1)); // 서로 바꿈.
8.        if(is_palindrome(cmp)==1){ // 팰린드롬이라면
9.          cout << i+1 << " " << j+1; // 서로 바꾼 위치 출력. 위치는 1부터 시작.
10.         return 0; // 프로그램 종료.
11.       }
12.     }
13.   }
14. cout << -1;
15. // 위 반복문을 탈출했다면 두 문자를 바꿨을 때 팰린드롬임을 만족하는 경우가 없으므로 불가능임을 -
    1로 나타낸다.
```

시간 복잡도 $O(N^2)$로 해결 가능하다.

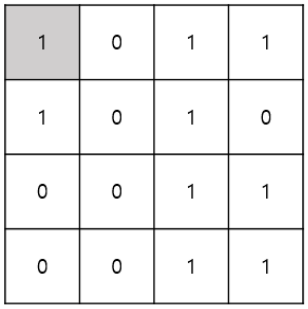
4. 판치기왕 김판판

시간 제한 : 2초 | 메모리 제한 : 128MB

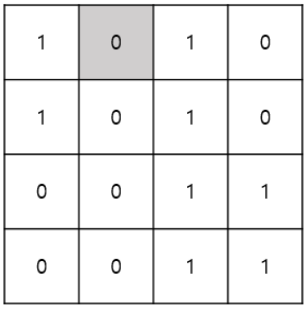
해설

greedy 기법을 사용해서 문제를 해결할 수 있다. $(0,0)$ 부터 $(M-N, M-N)$ 까지 차례로 동전의 상태를 확인한다. 만약 (i,j) $(0 \le i,j \le M-N)$ 동전의 상태가 뒷면이라면, (i,j) 부터 $(i+N, j+N)$ 까지의 모든 동전을 뒤집는다. $(M-N, M-N)$ 까지 탐색을 완료하고 모든 동전이 모두 앞면인지 확인해준다. 앞면이 아닌 동전이 있다면 -1을 출력해주고 모두 앞면이라면 동전을 뒤집어준 횟수를 출력한다.

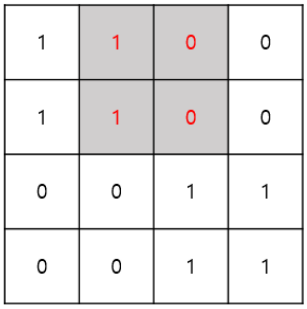
M=4
N=2



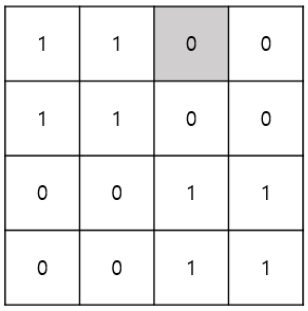
1. (0,0)==1 이므로 넘어간다



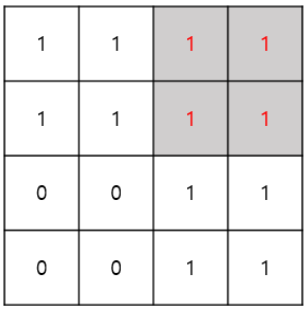
2. (0,1)==0 이므로 N*N만큼 동전을 뒤집어야 한다



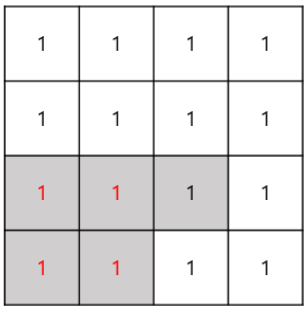
3. (0,1)을 기준으로 2*2만큼 뒤집어주고 마저 탐색한다



4. (0,2)==0 이므로 이전과 같이 동전을 뒤집어준다



5. 동전을 뒤집고 (1,0)을 탐색하러 간다



6. 이 과정을 (M-N, M-N)지점인 (2,2)까지 반복하고 뒤집어준 횟수를 출력해준다

5. 잔치기 잔치기!

시간 제한 : 2초 | 메모리 제한 : 128MB

해설

자료구조 Double-ended queue를 활용한 문제이다. C++의 STL인 deque(양방향 큐) 자료형을 사용하면 쉽게 해결이 가능하다.(deque의 자세한 사용 방법은 따로 설명하지 않음)

1. 잔을 한 번 치면, deque의 front를 pop하여 뒤쪽에 넣어준다.
2. 잔을 두 번 치면, deque의 back을 pop하여 앞쪽에 넣어준다.
3. 잔을 세 번 치면, 1번 과정 2번 반복.
4. 잔을 안치면, deque의 front를 pop하면 된다.

입력이 마무리 되면 남아있는 데이터 값(남은 사람의 번호)을 출력해주면 된다.

6. Trick or Treat !!

시간 제한 : 1초 | 메모리 제한 : 256MB

해설

그래프 완전 탐색을 할 수 있는 지에 대해서 물어보는 문제.

BFS 또는 DFS를 통해서 문제를 해결할 수 있다. 시간 복잡도는 $O(N + M)$.

그래프 탐색을 한 후 방문하지 못한 집의 번호를 정렬하여 오름차순으로 출력하면 된다.

7. NA LCS

시간 제한 : 1초 | 메모리 제한 : 256MB

해설

문제를 해결하기 위해선 크게 두 가지를 구해야 한다.

1. 소수

주어지는 두 수 $A, B(1 \leq A < B10{,}000)$ 중 큰 수를 M이라 하자. 소수는 가장 간단한 방법인 이중 반복문으로 시간 복잡도 $O(M^2)$ 만에 구할 수 있다. 하지만 문제에서 M의 제한이 최대 $10{,}000$이므로 주어진 시간 내에 해결할 수 없다. 하지만 에라토스테네스의 체 를 사용하면 시간 복잡도 $O(Mlog(logM))$만에 구할 수 있다.

2. 최장 공통 부분 수열(문자열)

두 문자열의 최장 공통 부분 문자열(Longest Common Subsequence)을 구해야 한다. 이 때 문자열의 길이는 최대 $4{,}719$가 된다($M = 10{,}000$일 때, 1부터 모든 소수를 구해서 이어서 붙였을 때, 소수의 개수 총 $1{,}299$개, 문자열의 길이는 $4{,}719$가 된다). 이는 다이나믹 프로그래밍(DP)를 통해서 구할 수 있다. 두 문자열의 길이를 L_1, L_2라 하면, 시간 복잡도 $O(L_1 L_2)$만에 구할 수 있다.