1. 크로스 컨트리

문제 파악

1. 각 팀별 선수 숫자는 6명 혹은 그 미만
2. 팀에 선수가 6명 미만일 경우 점수 산출에서 제외
3. 각 선수의 점수 = 등수
4. 팀별 상위 4명의 등수의 합이 팀별 점수
5. 점수가 가장 적은 팀이 우승
6. 동점인 팀 발생시 5번째 주자의 점수가 제일 작은 팀 우승

1. 크로스 컨트리

문제 파악

등수	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
팀	A	B	C	C	A	C	B	D	A	A	C	A	C	C	A
점수	1	n/a	2	3	4	5	n/a	n/a	6	7	8	9	10	11	12

2번째로 들어온 B팀의 선수는 B팀이 6명을 채우지 못했으므로
그 다음으로 들어온 C팀의 선수가 2등이 되어 2점을 챙겨간다.

1. 크로스 컨트리

문제 풀이

1. 들어온 선수의 팀을 순서대로 저장
2. 선수가 6명인 팀을 저장
3. 다시 1을 순회하면서 선수가 6명인 팀에 대해서만 점수 저장
4. 3에서 점수 저장할 때 상위 4명의 팀 점수와 동점 비교용 5번째 주자의 점수를 따로 저장

1. 크로스 컨트리

자료구조

Tuple: pair의 일반화, 3개 이상의 자료형을 묶어서 사용
get으로 각각의 자료형에 접근 가능

<팀 번호, 팀 점수, 다섯 번째 주자 점수>로 저장하여
팀 점수->다섯 번째 주자 점수 순의 기준으로 sort

1. 크로스 컨트리

핵심 코드

```
for (int j = 1; j <= n; j++) {
    int idx = find(allowed.begin(), allowed.end(), temp[j]) - allowed.begin();
    if (idx == allowed.size())  // 주자 인정 안됨
        continue;
    if (team[idx][1] < 4)  // 팀별 상위 4명의 주자의 점수가 반영
        get<1>(final[idx]) += score++;
    else if (team[idx][1] == 4)  // 동점 비교용 5번째 주자 점수 기록
        get<2>(final[idx]) = score++;
    else score++;  // 6번째 주자는 기록은 안하지만 점수 변동은 필요
    team[idx][1]++;  // 팀별로 몇 번째 주자까지 세었는지 확인
}
```

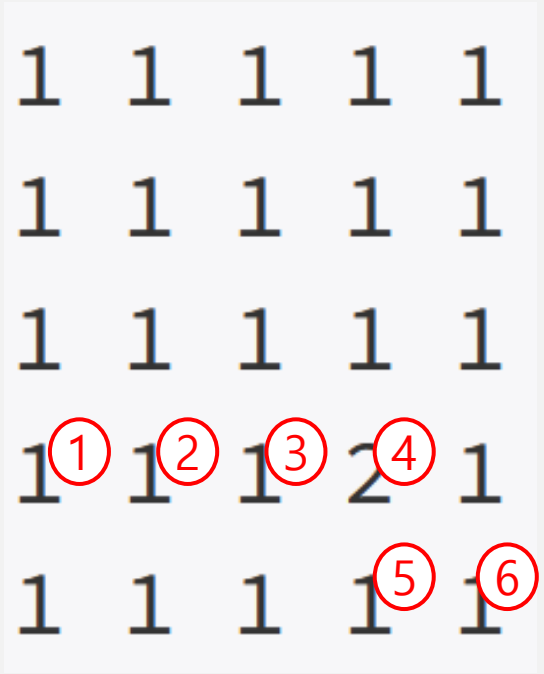
2. 숫자판 점프

문제 파악

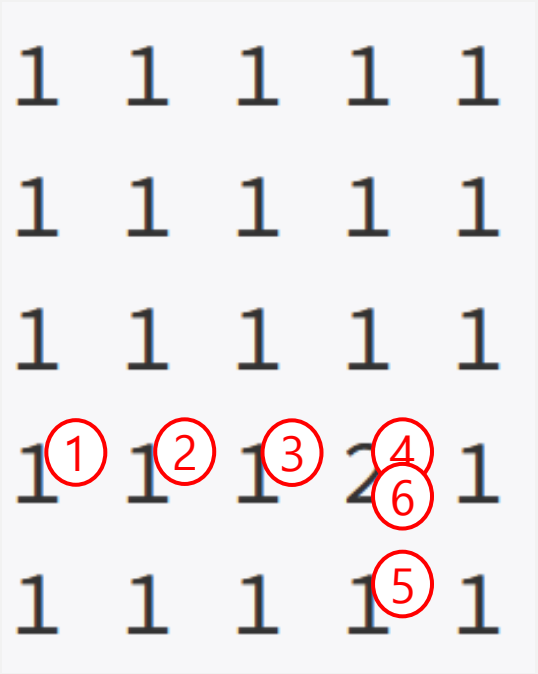
1. 5 X 5 숫자판에서 임의의 점에서 출발
2. 6번 이동 가능, 이전에 이동했던 칸으로 다시 이동할 수 있음
3. 출발 칸부터 이동한 순서대로 숫자를 기록해서 6자리 수를 만듦
4. 이렇게 나온 서로 다른 6자리 수의 총 개수

2. 숫자판 점프

문제 파악



111211



111212

2. 숫자판 점프

문제 풀이

DFS로 노드의 방문 여부를 따지지 않는 tree search를 사용

임의의 출발 지점을 가정하므로 모든 위치에 대해 DFS를 적용

시간은 2초, 격자 크기와 search 깊이는 작게 고정되어 문제 없다 판단

2. 숫자판 점프

자료구조

Set: 중복을 허락하지 않는 정렬된 자료구조
그래서 일단 가능한 모든 경우의 수를 따져서 set에 삽입하면
중복되는 경우는 전부 걸러진다.

2. 숫자판 점프

핵심 코드

```
int sq[5][5];
int xdir[5] = { 0, 1, 0, -1 };  // x 이동 방향
int ydir[5] = { 1, 0, -1, 0 };  // y 이동 방향
set<string> num;  // 편의상 string 사용

void six(int x, int y, string s) {
    if (s.size() == 6) {  // 길이가 6인 숫자가 목표
        num.insert(s);  // set에 삽입, 중복은 자동 제거
        return;
    }
    for (int i = 0; i < 4; i++) {  // DFS, 4방향 이동
        int xx = x + xdir[i];  // 다음 x좌표
        int yy = y + ydir[i];  // 다음 y좌표
        if (xx < 0 || xx > 4 || yy < 0 || yy > 4)  // 5x5 사각형 범위 내에서만 이동
            continue;
        six(xx, yy, s + to_string(sq[x][y]));  // 다음 위치로 이동, 지금까지 만든 숫자 전달
    }
}
```