1. 퇴사 2

문제 파악

1. 매일 1개의 상담 가능, 걸리는 시간과 이익 주어짐
2. 마지막 날(N)까지 일했을 때 최대 이익 구하기
3. 단, N + 1일 이후에 끝나는 상담은 받지 못함

1. 퇴사 2

문제 풀이

1. DP를 이용한 풀이 가능
2. 시작일 A일에 상담에 1일이 걸린다는 것은 정산을 (A + 1)일에 받는 것으로 계산할 수 있음
3. 1일부터 (N + 1)일까지 각 날짜별로 구해지는 최대 이익 계산

1. 퇴사 2

핵심 코드

```
for (int i = 1; i <= n + 1; i++) {
    dp[i] = dp[i - 1];
    for (int j = 0; j < job[i].size(); j++) {
        dp[i] = max(dp[i], dp[i - job[i][j].first] + job[i][j].second);
    }
}
```

job[i]는 i일에 끝나는 일의 {걸리는 시간, 이익} pair를 담은 vector

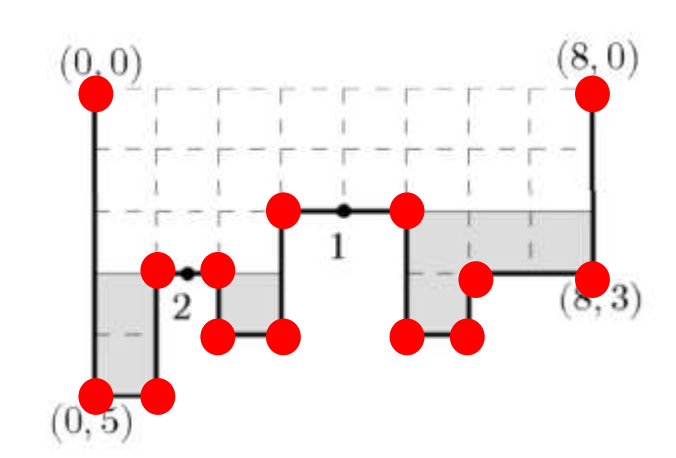
2. 수족관 1

문제 파악

1. 수족관 바닥에 구멍이 있음
2. 구멍을 통해 모든 물이 빠져나가고 남은 물의 총합을 계산
3. 주어지는 처음 입력은 1차원 수조의 삼면을 이루는 꼭짓점의 좌표

2. 수족관 1

문제 풀이



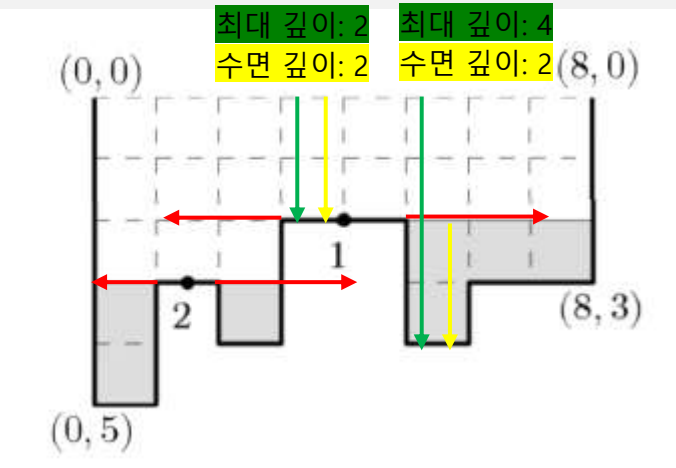
문제 조건:
수족관의 경계를 이루는 변은 꼭짓점 (0, 0)부터 시작하는 데, 수직선분으로 시작하여 수평선분과 수직선분이 번갈아가며 반복되다 수직선분으로 끝난다.

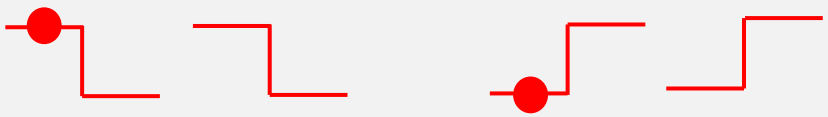
주어지는 꼭짓점을 그림으로 나타내면 왼쪽과 같다.
각 칸마다 "최대 깊이"를 계산하려면 문제 조건을 이용하여, 순서대로 주어진 꼭짓점의 (x, y) 좌표 중 y좌표가 동일한, 연속된 쌍을 찾아서 계산할 수 있다.

```
int n, k, t1[2], t2[2];
cin >> n >> t1[0] >> t2[0];
for (int i = 1; i < n; i++) {
    cin >> t1[1] >> t2[1];
    if (t2[1] == t2[0] && t1[1] != t1[0]) {  // depth[i]는 x좌표 i의 깊이 ([i, i + 1))
        for (int j = t1[0]; j < t1[1]; j++)
            depth[j] = t2[1];
    }
    t1[0] = t1[1], t2[0] = t2[1];
}
```

2. 수족관 1

문제 풀이





각 깊이의 구멍을 기준으로, 양방향으로 한 칸씩 나아가면서 "수면 깊이"를 갱신, 구멍이 있는 깊이는 "최대 깊이" = "수면 깊이"

각 칸의 깊이를 기준으로, 양쪽의 높이가 더 깊을 수도,
얕을 수도 있음

I) 옆 칸의 깊이가 더 깊은 경우:
  i) 현재 칸에 구멍이 있으면 옆 칸 수면 깊이 = 현재 칸 수면 깊이
  ii) 현재 칸에 구멍이 없어도 옆 칸 수면 깊이 = 현재 칸 수면 깊이

II) 옆 칸의 깊이가 더 얕은 경우:
  i) 현재 칸에 구멍이 있으면 옆 칸 수면 깊이 = 옆 칸 최대 깊이
  ii) 현재 칸에 구멍이 없어도 옆 칸 수면 깊이 = 옆 칸 최대 깊이

→ 옆 칸 수면 깊이 = min(현재 칸 수면 깊이, 옆 칸 최대 깊이)

2. 수족관 1

핵심 코드

```
for (int i = 0; i < k; i++) {
    int a, b, c;
    cin >> a >> b >> c >> b;
    // 구멍있는 구간은 누수 최대 깊이 = 현재 깊이
    for (int j = a; j < c; j++)
        hole[j] = b;
    // 구멍있는 구간 양 옆으로 누수 최대 깊이 갱신
    int cur_hole = b;
    for (int j = a - 1; j >= 0; j--) {
        cur_hole = min(cur_hole, depth[j]);  // 깊이가 더 얕은 곳에서는 최대 수면 높이가 그 깊이로 고정
        hole[j] = max(hole[j], cur_hole);
    }
    cur_hole = b;
    for (int j = c; j < t1[0]; j++) {
        cur_hole = min(cur_hole, depth[j]);
        hole[j] = max(hole[j], cur_hole);
    }
}
```

이후에 모든 칸마다
최대 깊이와 수면 깊이의
차를 구해서 합하면 된다.