week18

2024-11-07

Load Balacing(Silver)

소가 있는 목장에 가로 , 세로 울타리를 쳐서
네 구간에 들어가는 소들중 가장 많은 수의 소가 들어있는 구역의 최소값

소는 최대 1000 마리이고 ,
소의 x, y 좌표의 범위는 [0, 1,000,000]

가로 , 세로 울타리가 만나는 점들을 전부 구해서
구간 합을 통해 네 구역에 소가 얼마나 있는지 알 수 있지만
좌표의 범위가 너무 넓어서 힘듦

이중 for 문 돌면 10 의 12 승

Load Balacing(Silver)

소가 최대 1000 마리라는 점을 이용해 좌표 압축으로 풀 수 있다 .
가능한 좌표가 얼마나 많든 결국 1,000,000 x 1,000,000 의 좌표 안에
1000 개의 점만 찍히므로 좌표를 상대적으로 압축하면

1000 x 1000 이라고 생각할 수 있다 .

Load Balacing(Silver)

소의 좌표가
[1, 1], [2222, 2222], [3333, 3333], [5555555, 555555] 라면

[0, 0], [1, 1], [2, 2], [3, 3] 과 같이

x, y 좌표를 정렬해서 상대적으로 배열할 수 있음

소의 좌표가 같은 경우는 없으니 , 압축된 좌표에서도 무조건 서로 다른 좌표를 갖게됨

Load Balacing(Silver)

X 좌표와 Y 좌표를 벡터에 따로 저장

```
int n; cin >> n;
vector<pii> a(n);
vector<int> xpos, ypos;
for(auto &[x, y] : a) {
    cin >> x >> y;
    xpos.push_back(x);
    ypos.push_back(y);
}
```

```cpp
xpos.erase(unique(xpos.begin(), xpos.end()), xpos.end());
sort(ypos.begin(), ypos.end());
ypos.erase(unique(ypos.begin(), ypos.end()), ypos.end());

int r = xpos.size();
int c = ypos.size();
vector<vector<int>> psum(r + 1, vector<int>(c + 1, 0));

for(auto &[x, y] : a) {
    x = lower_bound(xpos.begin(), xpos.end(), x) - xpos.begin();
    y = lower_bound(ypos.begin(), ypos.end(), y) - ypos.begin();
    psum[x + 1][y + 1] = 1;
}
```

x 좌표 기준

x 좌표를 유니크하게 정렬하고 , 정렬된 x 좌표들에서 현재 소의 x 좌표의 위치를 이분탐색으로 찾음

→ 좌표 찾고 바로 누적합 배열에 1 을 더해줌 → 누적합 구하고 r x c 돌면서 최소값 찾기

성냥개비

큰 수 → 그리디
가장 큰 수를 만드려면 일단 자리수가 제일 커야 함
숫자 1 을 만드는데 성냥개비 2 개로 제일 작게 들기때문에
성냥개비가 짝수인 경우 → 성냥개비 / 2 만큼 1
성냥개비가 홀수인 경우 → 7 하나 하고 (성냥개비 – 3) / 2 만큼 1

성냥개비

작은 수 → 디피

dp[x] = y 일 때
성냥개비 x 를 사용했을때 최소값 y 라고 생각
dp[0] = 0 , 성냥개비 1 개인 경우는 없으니깐 잘 처리 해줘야함

그러면 dp[x] = for(min(dp[x], dp[x – 성냥개비 수] * 10 + 만들 숫자));

{ 만들 숫자 , 성냥개비 수 } => {{1, 2}, {2, 5}, {4, 4}, {6, 6},{7, 3}, {8, 7}, {0, 6}};
같은 성냥개비가 든다면 작은수만 남기기
→ 3 을만들려면 5 개 드는데 , 2 만들때 5 개 필요하니까 3 만들꺼면 무조건 2 를 만든다는거라 배열에서 제거해 줌

```cpp
ll dp[101];
void init_min() {
    vector<pair<ll, int>> pos{{1, 2}, {2, 5}, {4, 4}, {6, 6},
                              {7, 3}, {8, 7}, {0, 6}};
    dp[0] = 0;
    dp[1] = 1e17;
    for(int i = 2; i <= 100; i++) {
        dp[i] = 1e17;
        for(auto &[num, cost] : pos) {
            if(i - cost >= 0) {
                if(num == 0 && dp[i - cost] == 0) continue;
                dp[i] = min(dp[i],  dp[i - cost] * 10LL + num);
            }
        }
    }
}
```