Kick Start 2020 - Round F

Analysis: Painters' Duel

Test Set 1

There are only 4 rooms, (1, 1), (2, 1), (2, 2) and (2, 3). Both the players, will try to paint the room (2, 2), painting which would block the other player to paint more rooms.

Thus there are five cases,

- If **C** = 2, none of the players can move, therefore the answer is 0.
- If any of the players starts on room (2, 2) and **C** != 2, they will paint two rooms and the other player will paint one room.
- If room (2, 2) is blocked, none of the players can move, therefore the answer is 0.
- If any room other than (2, 2) is blocked, and no player start on (2, 2), then Alma will paint two rooms and Berthe will paint one room.
- If no room is blocked, and no player start on (2, 2), then Alma will paint three rooms and Berthe will paint one room.

Test Set 2

At any point during the game, few rooms of the museum might be under construction, few might have been painted by Alma, few might have been painted by Berthe and a few might be unpainted.

The rooms which are either under construction, painted by Alma or painted by Berthe are not paintable any further. Lets represent these rooms as blocked rooms, and the remaining rooms as free rooms. Thus, without losing any information we can uniquely represent the state of the museum with the following parameters,

- The free rooms.
- Position of Alma.
- Position of Berthe.
- Whose turn is it?
- Present score of the game

We can represent blocked/free rooms of the museum as a binary string, 0 denoting the blocked rooms, and 1 representing the free rooms.

Lets define a function, `CalcScore()`, which takes the binary string, position of Alma and Berthe, and whose turn it is as input parameters and returns the optimal score as output, that is, the minimal possible score if it is Berthe's turn and maximum possible score if it is Alma's turn.

`CalcScore()` recursively calls itself with every reachable state, that is, the player tries to paint all the three neighbouring rooms, thus, recursively call `CalcScore()` for all the valid states.

Lets denote T(N) as time complexity when N rooms are free. In every nested call of `CalcScore()`, the number of free rooms decreases by 1, thus the recursion depth of `CalcScore()` can be at max $\mathbf{S}^2$-2. A room has at most three neighboring rooms, but if it is not the starting position at least one of them will already be painted because that's where the player came from. Thus, when a player reaches a room, he has only two neighboring rooms to move to. So, every call to `CalcScore()` can generate at max 2 recursive calls to `CalcScore()`.

Thus,

- $T(N) = 2 \times T(N-1)$
- $T(0) = O(1)$

Therefore, the time complexity of the initial configuration is $O(2^{\mathbf{S}^2})$. Which although seems to be not good enough for **S** = 6, but in practice, the solution passes well within the time limits as most of the recursion depth will be way less than the presumed maximum. Depends on the implemetation details, but the number of calls to `CalcScore()` for the worst case is of the order of 10^5.

Although not needed for Test Set 2, but, we can use techniques like Memoization and Alphaâ€“Beta pruning to further speed up the solution.