**Don’t Connect 5**

“The essence of strategy is choosing what not to do”

― Michael Porter

You and your friend Eric are bored playing the classic [*“Five in a Row”*](https://en.wikipedia.org/wiki/Gomoku) game, so both of you decided to make the game more interesting by adding the following rule changes:

* The game now happens on the vertices of a [hexagonal grid](https://mathworld.wolfram.com/HexagonalGrid.html) instead of a board.
* Unlike the classic game, connections do not have to go in a straight line(and there is no straight line to go…) - any path is a connection.
* And you win when you con - ???

“Wait!”, your other friend Kevin interrupted. Kevin has always been bad at the connect 5 game, so he demanded that you and Eric make it a **Don’t Connect 5 game.**

*The following are the (actual) rules*

* *The board is a hexagonal grid with radius 4*
* *Three players take turns to place stones of their color on the vertices. Each player may choose to pass and not place a stone*
* *The game ends when no stone is placed for three consecutive turns, either because there is no space or no player chooses to make a move.*
* *Score is calculated by the following method for each player:*
  + *For each* [*connected component*](https://en.wikipedia.org/wiki/Component_(graph_theory)) *formed by the stones placed down by the player, the diameter(*[*length of the longest path*](https://en.wikipedia.org/wiki/Distance_(graph_theory)#:~:text=The%20diameter%20d%20of%20a%20graph%20is%20the%20maximum%20eccentricity%20of%20any%20vertex%20in%20the%20graph.%20That%20is%2C%20d%20is%20the%20greatest%20distance%20between%20any%20pair%20of%20vertices%20or%2C%20alternatively%2C)*) in the component is calculated.*
  + *Depending on how many stones are there in the longest path, a score is added to the player’s final score*

| *#stones* | *<3* | *3* | *4* | *5* | *>5* |
| --- | --- | --- | --- | --- | --- |
| *score* | *0* | *1* | *3* | *0* | *0* |

* *The scores from all of a player’s connect components are summed, giving the total score.*

You are to implement one function, bot\_move

bot\_move takes parameters

* The current board: a dictionary of {(x, y, z):id}, where x, y, z is the hexagonal coordinate and id is the player id(0, 1, or 2)
* Your player number(0, 1, or 2)

Bot\_move should output a 3-tuple of hexagonal coordinates, the position of your next move; or None, if no move is intended to be taken.

**Time/Memory**

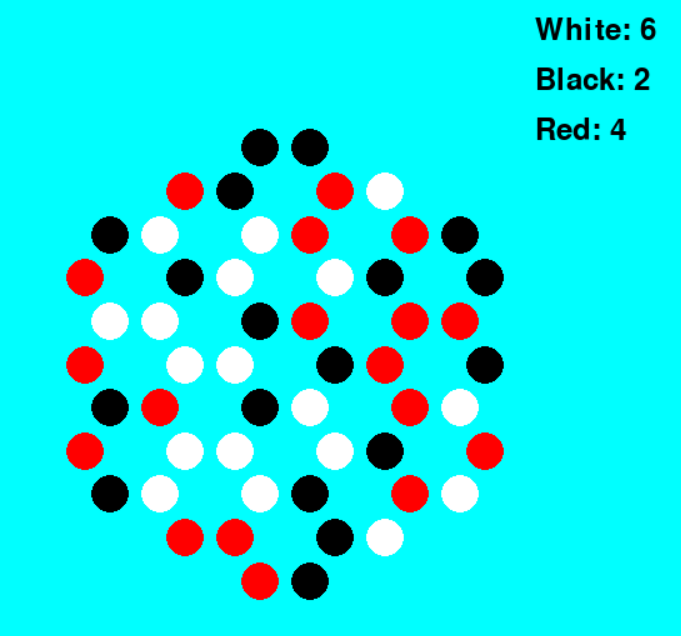
* Memory limit: 256 MB
* Time limit: 30 seconds **for each game**(not each move)
* If there is a timeout, your bot will **not move** for the rest of the game.
* In case of an illegal output (occupied or out-of-bound), **no move** will be taken.

**Grading:** Your ranking in each game affect your ELO rating, with changes proportional to

| Rank | Score | Rank | Score | Rank | Score | Rank | Score |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | .5 | 1 | 1 | 1 | 0 |
| 2 | 0 | 1 | .5 | 2 | -.5 | 1 | 0 |
| 3 | -1 | 3 | -1 | 2 | -.5 | 1 | 0 |

**Example of longest path calculation**

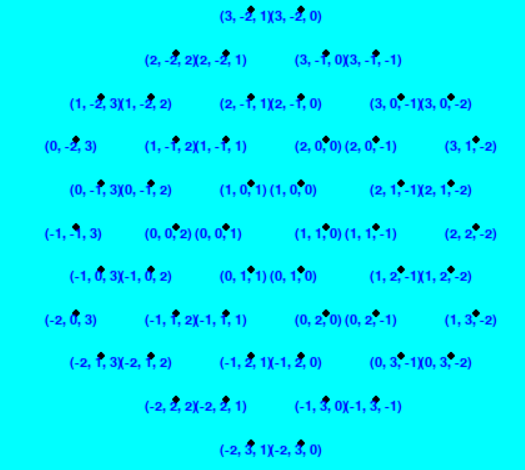
|  |  |
| --- | --- |
| This connected component gives red 0 point because it has diameter 5. | This connected component gives red 3 points because it has diameter 4. |

**Example of score calculation**

In the above game,

* White has 3 + 3 = 6 points since it has two connected components each of diameter 4.
* Black has 1 + 1 = 2 points since it has two connected components each of diameter 3.
* Red has 3 + 1 = 4 points since it has one connected component of diameter 4 and one of diameter 3.

**Explanation of Hexagonal Coordinates**



Each position is encoded by a coordinate (x, y, z). The ‘center’(which is not a valid position) has coordinates (0, 0, 0). An increase in each entry of the coordinate corresponds to moving in a certain direction.

* x: 60 deg, top-right
* y: -60 deg, bottom-right
* z: 180 deg, left