Programming bot for Hypersonic coding game. Status update 2019-05-02.

Performance:

- EugeneYakNN is 17th/245 at Silver League (top result, average is twentyish) with ab9541c at GitHub.
- EugeneYakNN is in Top 3% in global ranking (34,985 th of 1,440,862).
- ~ 0.5 ms think time (limit is 100ms) with conservative memory footprint without frequent reallocations.

Features / Architecture:

- Most of the battles bot survives till the end, collects items and blows multiple boxes when possible.
- Code correctly predicts next round grid (but not entities) to ensure it has right understanding of game rules. This includes chain reaction of bomb explosions, correct detection of explosion obstacles. Each turn it verifies the predicted grid with input from game engine and will exit with error once founds mismatch.
- Input is validated
- Debug techniques: always dump inputs to stderr and accept input from file so it is very easy to reproduce any issue from web IDE and possible to implement automated tests in future.
- Bot prints internal states and estimations to simplify debugging with web IDE.
- To avoid possible bugs the code always operates with “Position” instead of individual coordinates those could easily be mixed up

Limitations:

- Bot has limited logic to predict consequences of placing bomb: it only counts boxes at neighbor cells and sometimes traps itself by bomb. Also it is forbidden to place bomb at explosion range of other bomb to reduce the chance it is unable to run away in case of chain reaction.
- Bot places bombs if the current cell bomb placement value is not worse that the distant target cell, which leads to bug when the distant target is item without boxes around. On the other hand sometimes it helps to blow opponents at final rounds by active bombs placement.
- Bot has poor algorithm of choosing next distant target. It scans cells for a certain depths from its location with Depth-first search, in addition to unreliable scoring system sometimes it starts to randomly switch between targets every turn which makes it stuck between couple of cells (*Asinus Buridani inter duo prata* bug).
- The search algorithm has poor scalability and high maintain cost. This complicates adding complex features e.g. treat cell as safe if it will explode earlier than the player reaches it.
- Very rare bug when a bot cannot find the escape path in case of lot of bombs - maybe because of limited depth of search – need to debug.

Next steps:

- In short term tune the algorithm to better estimate bomb placement value (count all boxes in explosion range), it worth trying to prioritize boxes over items, fix obvious bugs (*Asinus Buridani inter duo prata*, unsafe walking, not seeing safe escape path, useless bomb placing, etc).
- In long term: instead of tuning existing algorithm, do brute-force Alpha-Beta MiniMax. Challenges here: ensure it will fit into 100ms (need prototyping and progressive deepening), predicting other players will dramatically increase branching factor so it may help to assume they staying in the cell and only decide whether to place a bomb, need implementation for prediction of entities and prediction of score for destroyed boxes, need efficient resource management to limit footprint and latency.