**Team Name:** RedVelvet
**Team Number:** 7

# Search Algorithms

We plan to use one of the following algorithms:

- Monte Carlo Tree Search (MCTS):
  - MCTS with an Upper Bound Confidence (UCB) function which keeps balance between *exploitation* and *exploration*.
  - For each move, we run some iterations of the algo (around 500) ensuring that a valid move is returned within the time limit.
  - $UCB = w/n + c * \sqrt{lnN/n}$

    $w$ : number of wins with that node as an intermediate

    $n$ : number of simulations with that node as an intermediate

    $N$ : total simulations for current move

    $c$ : a balancing constant

- MiniMax with Alpha-Beta Pruning:
  - Due to time limit of 24 secs, the depth of our search tree will be limited to three.

# Heuristics

1. Considering the evaluation criteria (section 5.1) following are the scores assigned to each smallboard type:

- Winning a center small board adds 30 points to the heuristic.
- Winning a corner small board adds 60 points to the heuristic.
- Winning a side small board adds 100 points to the heuristic.

**2.** Considering the *small_boards_status* of the board, we check for all of the eight possible winning combinations**:**

- ***Scale_Factor*** indicates the importance of winning that small board. Higher the *scale_factor* of a small board, higher are the chances of winning the big board by playing / winning that small board.
- [ - - - ]
    - In this case, we will try to win any of the small board and initialise the *scale_factor* of each of the small boards to **1**.
- [ x - - ] adds 250 points to the heuristics.
    - Here, we assign a *scale_factor* of **2** to the empty small boards of this combo. Hence, this will increase the chances of a move being played in one of the empty small boards and increase the chances of winning.
- [ x x - ] adds 500 points to the heuristics.
    - Here, we assign a *scale_factor* of **3** to the empty small board of this combo. Hence, this will increase the chances of a move being played in the empty small board and increase the chances of winning.
- [ x x x ] adds 1000 points to the heuristics.

**3.** Considering the cells of a small board:

- [ x - - ] adds ***scale_factor * 5*** points to the heuristics.
- [ x x - ] adds ***scale_factor * 10*** points to the heuristics.
- [ x x x ] adds ***scale_factor * 20*** points to the heuristics.

**4.** The above scoring criteria will be followed for the opponent but with negative score (basic minimax approach).

# Competency

We think our heuristic is competent because of following reasons:

- Introduction of a scale factor to each small board.
- Our heuristic gives prefers moves which lead to winning the big board and not just a small board.
- In case of a draw the scores are being assigned according to the small boards won by each player. Our heuristic keeps this into account in point **1** of the above section.

## Advantages of Minimax over MCTS

- MCTS works on random simulations, so it needs a lot of iterations to find a good move. And since the time limit is 24 seconds, we may not find the best possible move using MCTS.
- Also, with the increase in the number of iterations, the memory requirement increases.