Simulation of Minimax and alpha-beta algorithm for game

Project Group 12

By

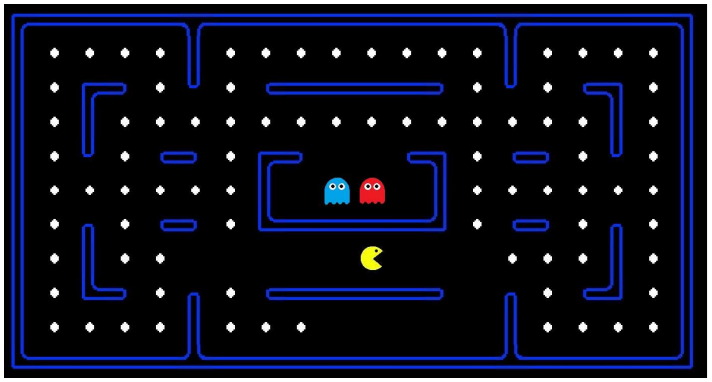
Sanket Chandorkar
sxc127330@utdallas.edu

Overview

- Introduction to Pac-Man
- Problem statement
- Formalization
- Utility function
- Minimax Simulation
- Alpha-Beta Simulation
- Comparing results
- Challenges
- Possible Future Enhancements
- Conclusion

Introduction to Pac-Man

- The all-time classic game: Pac-Man
- Developed by Namco in the 1980's. Hugely popular and largely synonymous with the video-game culture of the times.



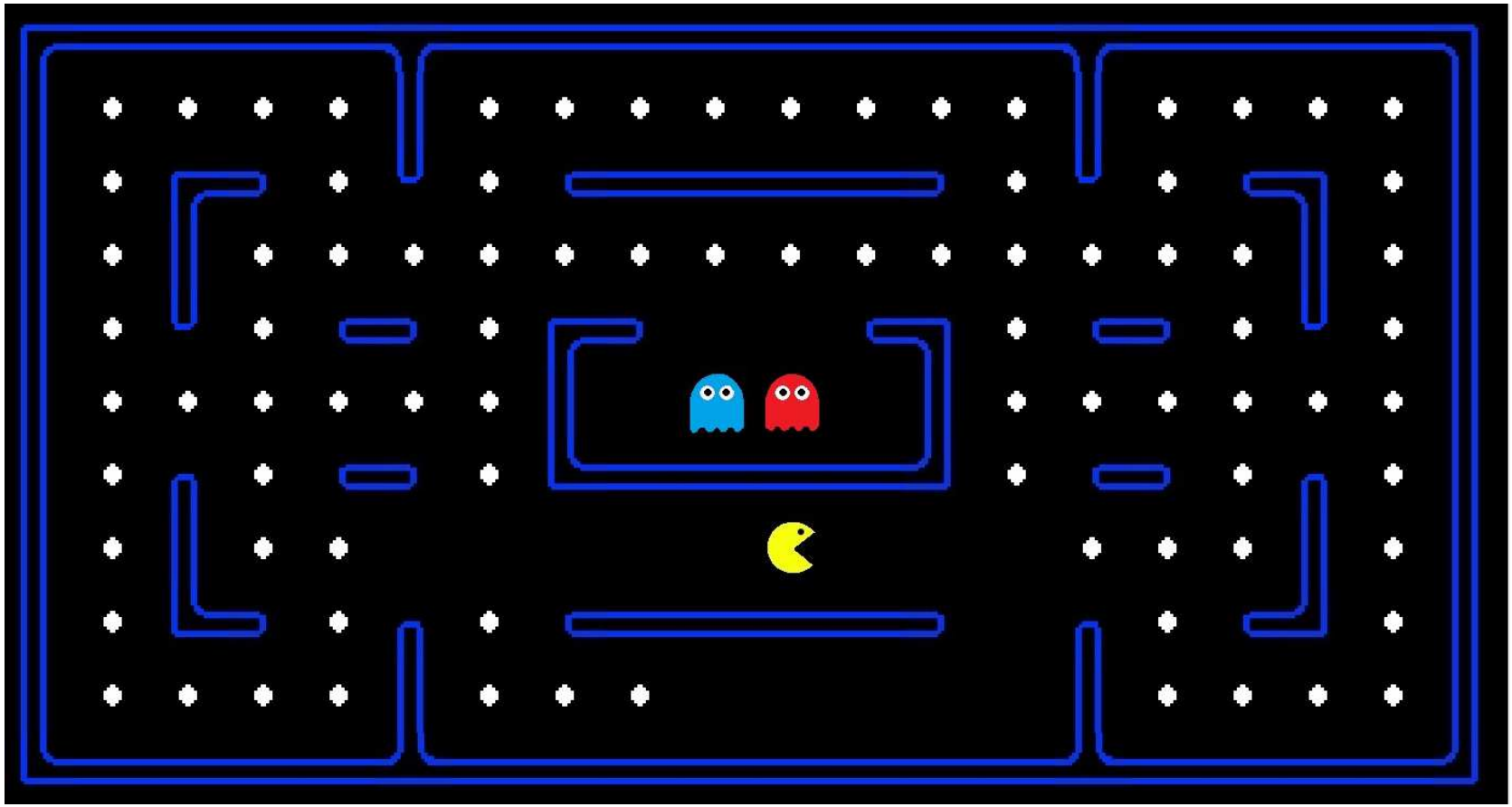
Game type

	deterministic	chance
perfect information	PAC MAN	backgammon monopoly
imperfect information	battleships, blind tictactoe	bridge, poker, scrabble nuclear war

Problem statement

Pac-Man 

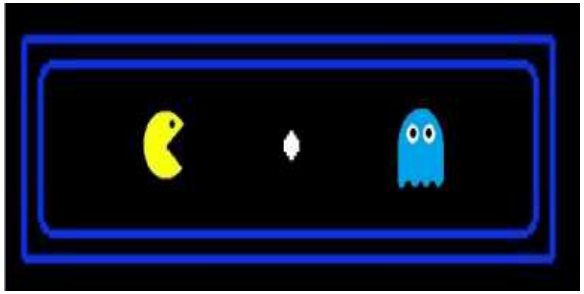
Ghost  



Formalization

State

Position of the PAC Man ,Ghosts and dots



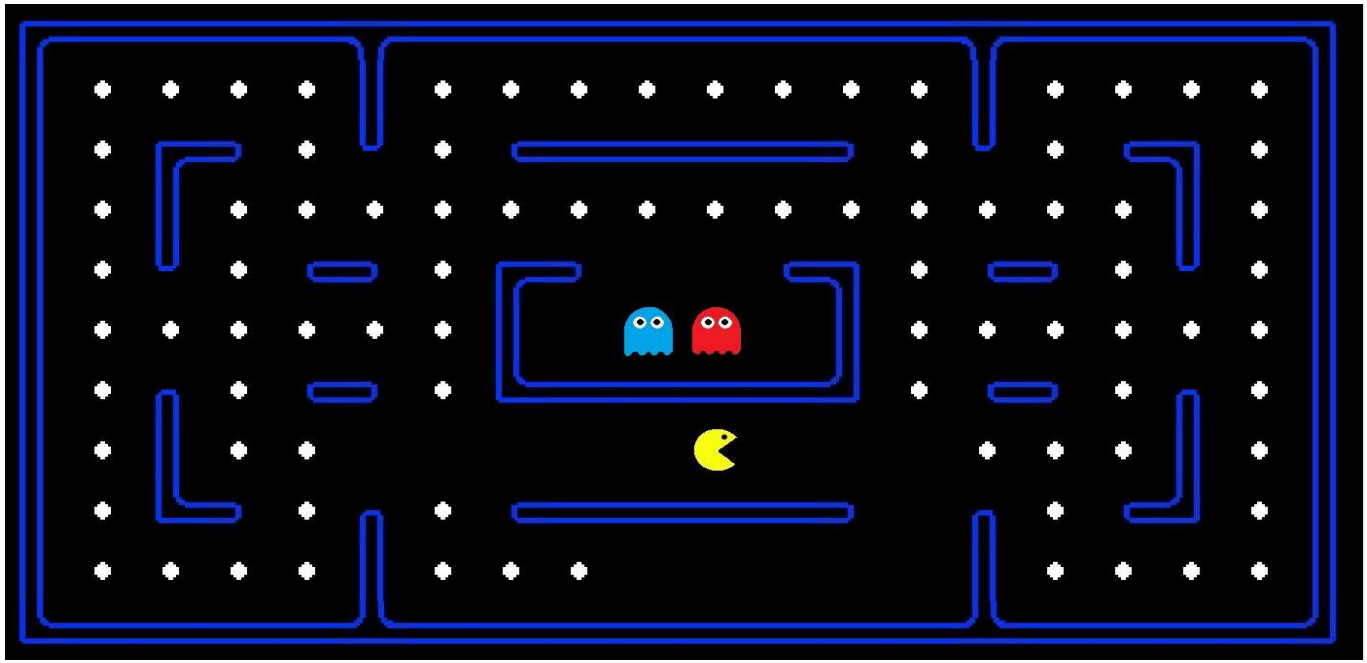
State1

State2

Formalization

Game Maze

The constrained environment where PAC Man and the ghost moves.

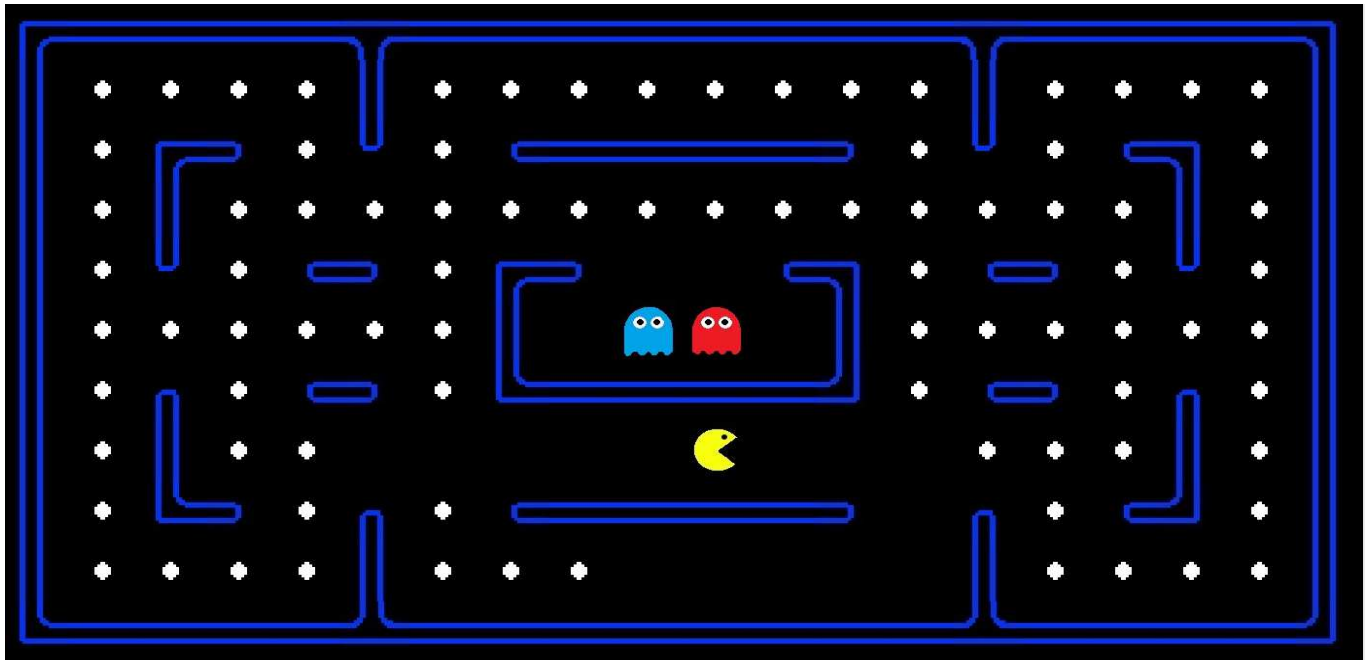


Formalization

Goal

Pac-Man wins by eating all dots.

Looses if ghosts eats him first.



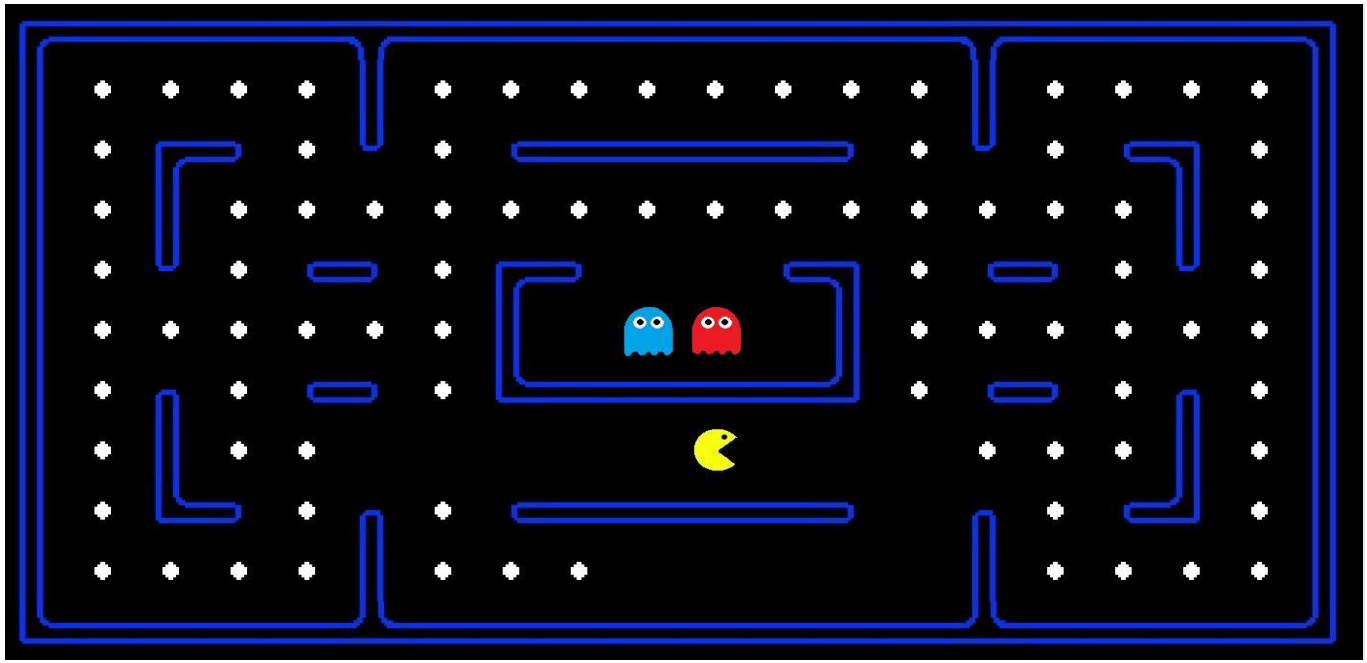
Formalization

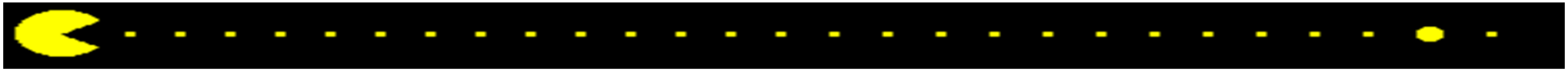
Operation

Simple_Move: A simple move is any direction.

Eating_dot_move: PAC Man eats the dot in adjacent space.

Killing_move: Ghost kills PAC Man in adjacent space.





Utility Function

- Most important component of game design.
- Decides the intelligence of the game.

Utility Function Demo

Utility Function

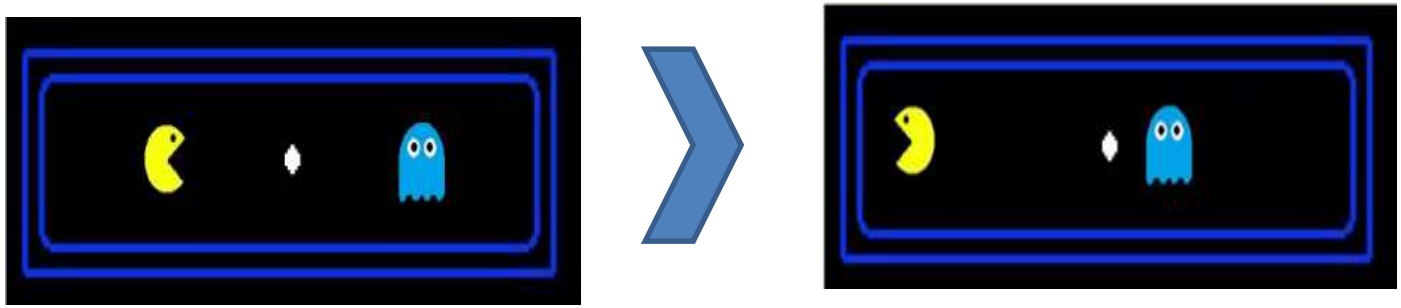
- Utility Function =

  [Step taken by Pac-Man X (-1)] +

  [Dots Eaten X (100)] +

  [Win X 5000/Lose X -5000] -

  [Min actual distance between Pac-Man & Ghost]

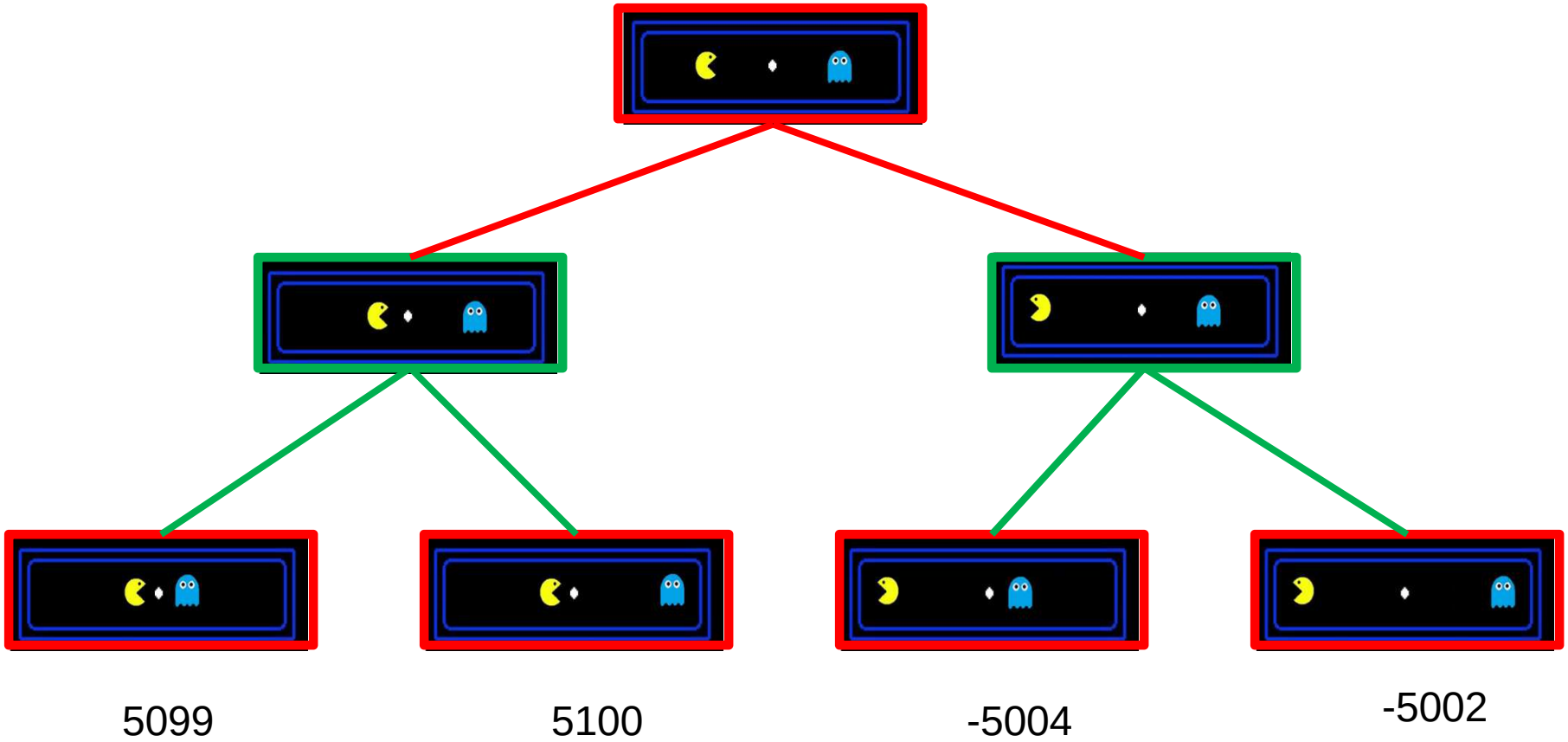


State1 State2

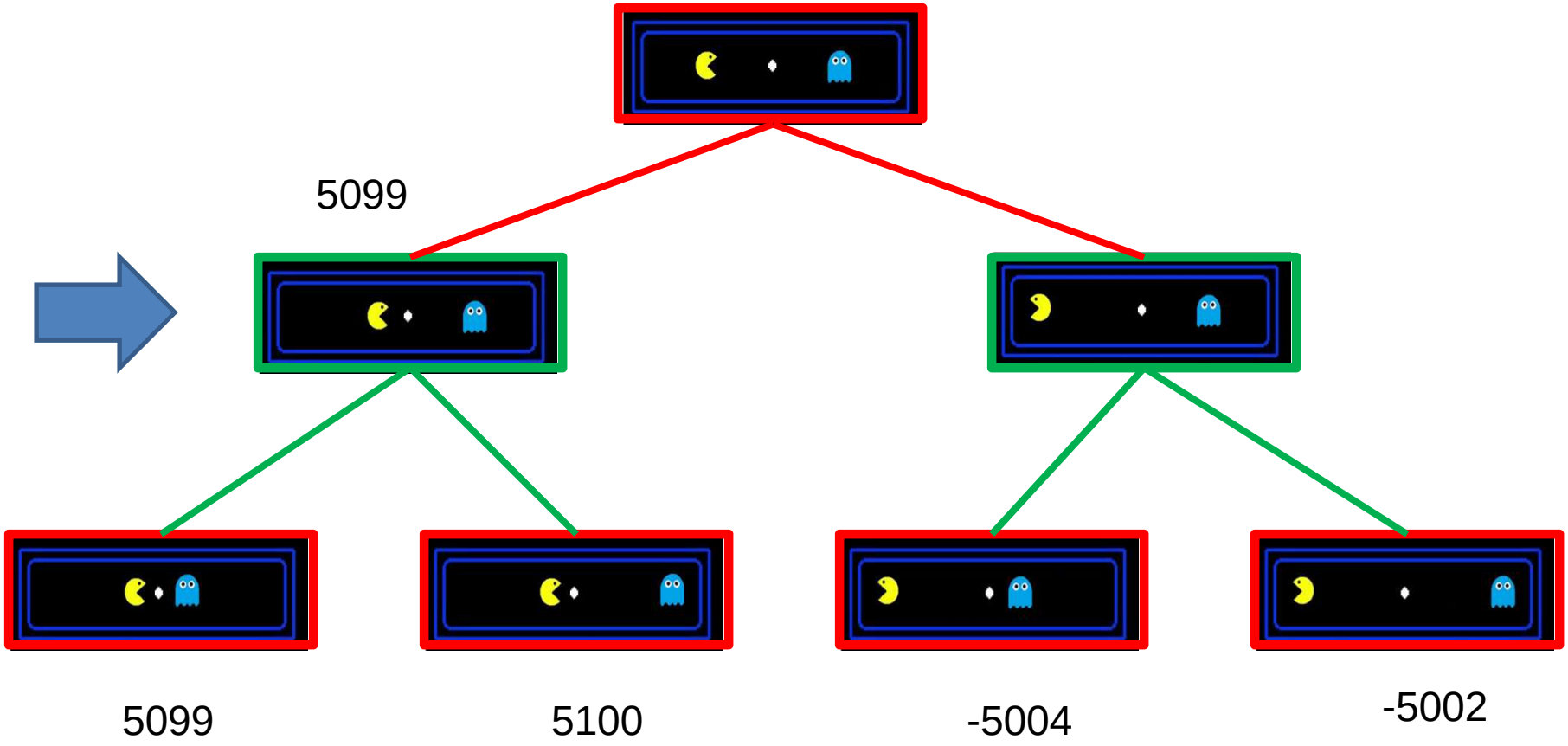
MiniMax

- Pac Man tries to maximize its score.
- Ghost tries to minimizes Pac Man’s Score.
- Each terminal nodes value is computed as per utility function.

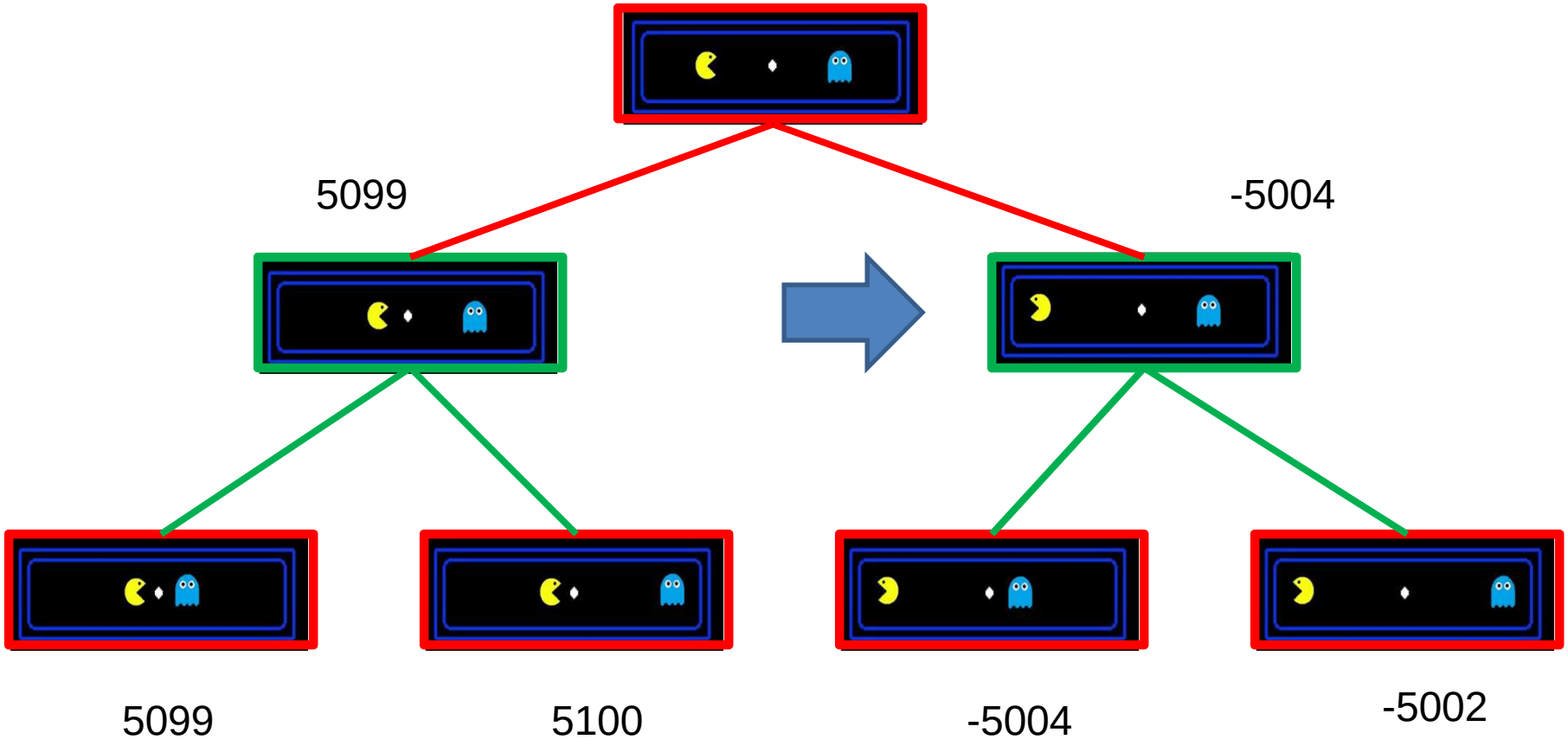
MiniMax Simulation



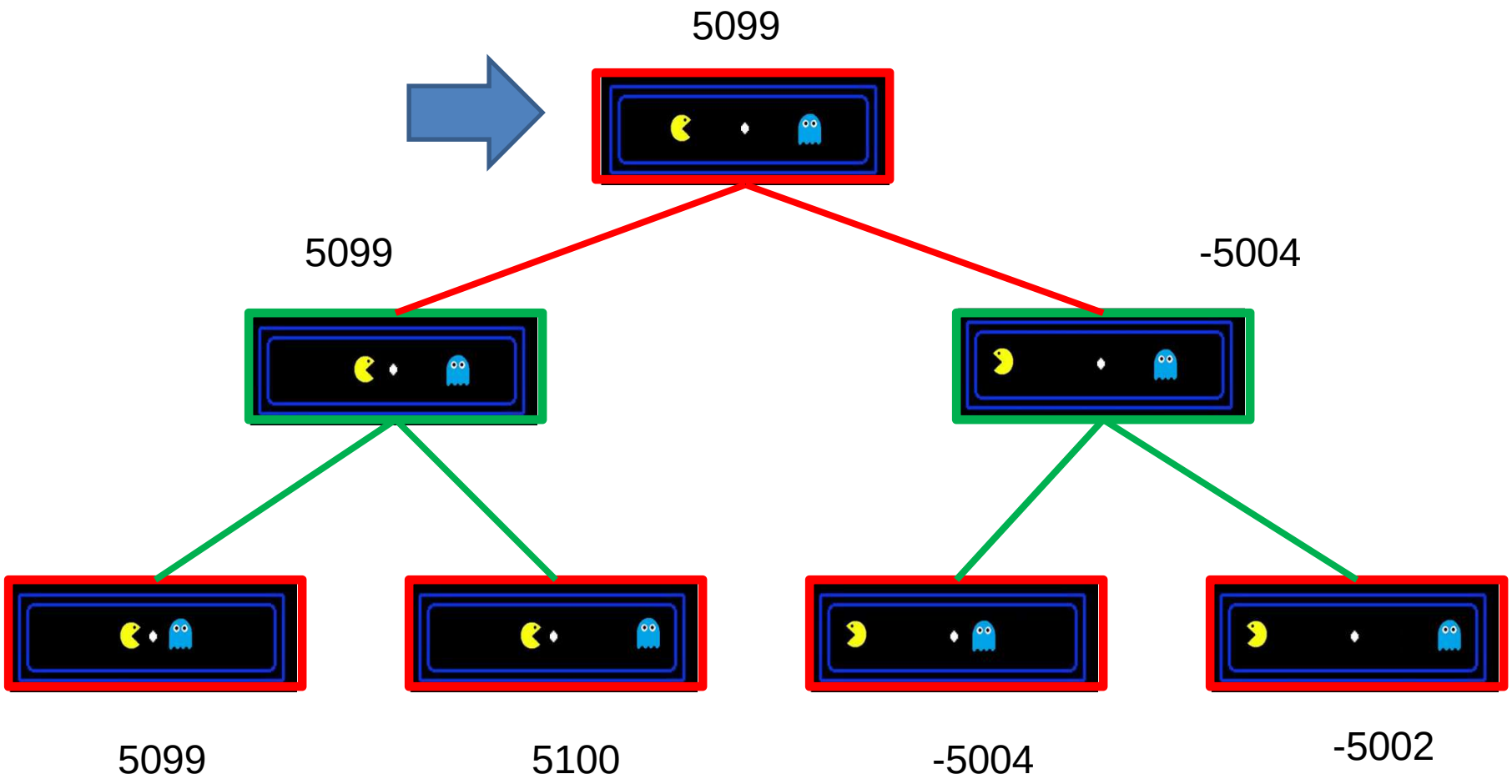
MiniMax Simulation



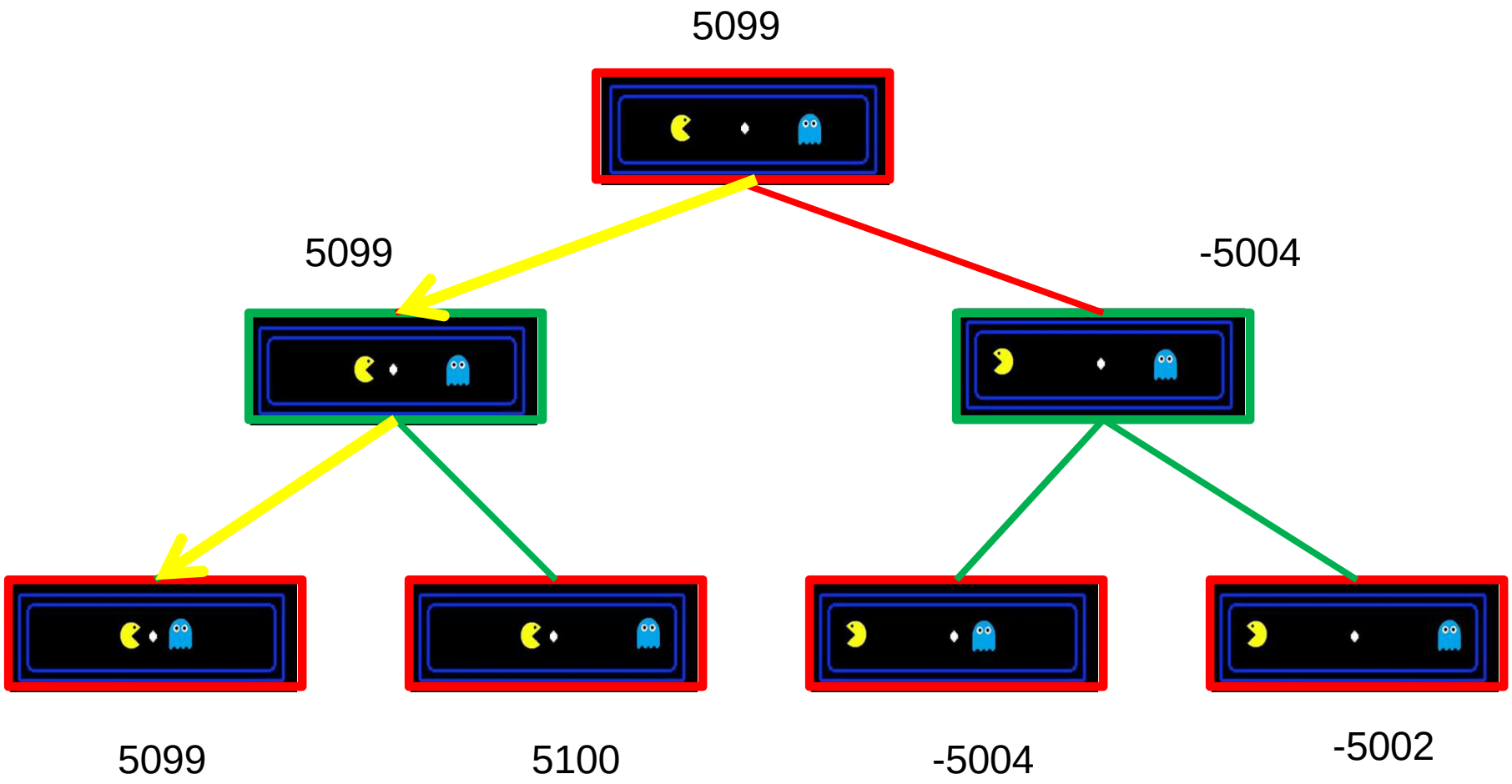
MiniMax Simulation



MiniMax Simulation



MiniMax Simulation



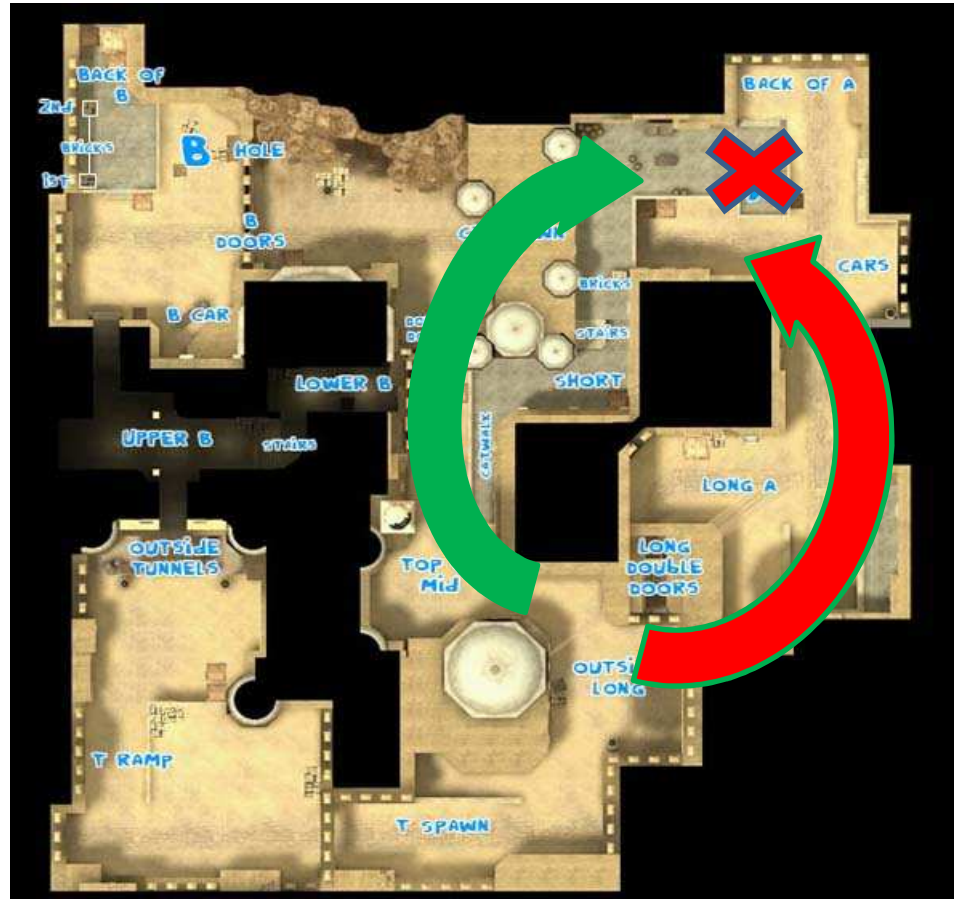
MiniMax Demo

MiniMax Limitations

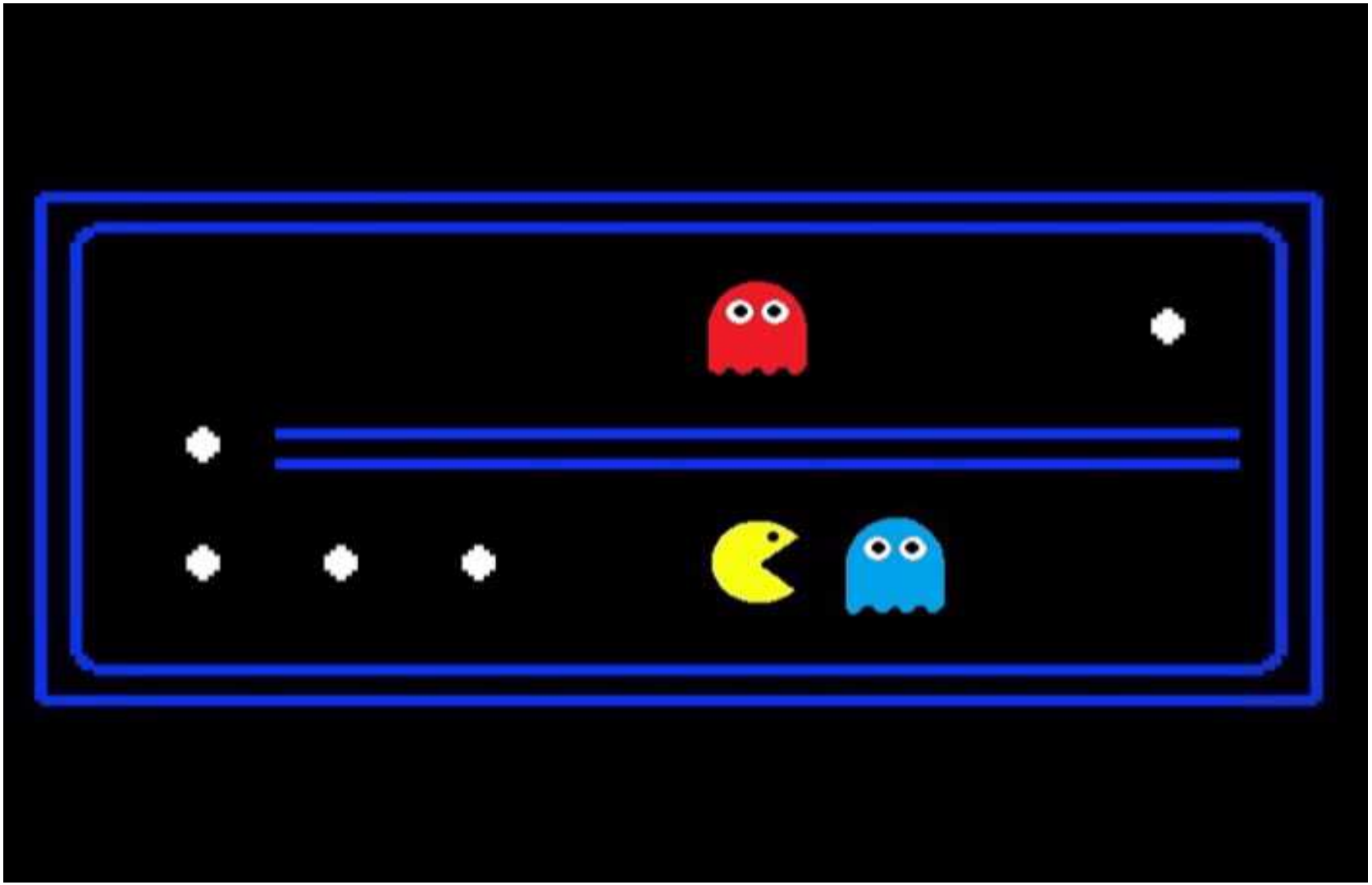
- Traverses the entire game tree.
- Not Suitable for big game trees.

Flanking in PAC MAN

Attack on the sides of an opposing force.



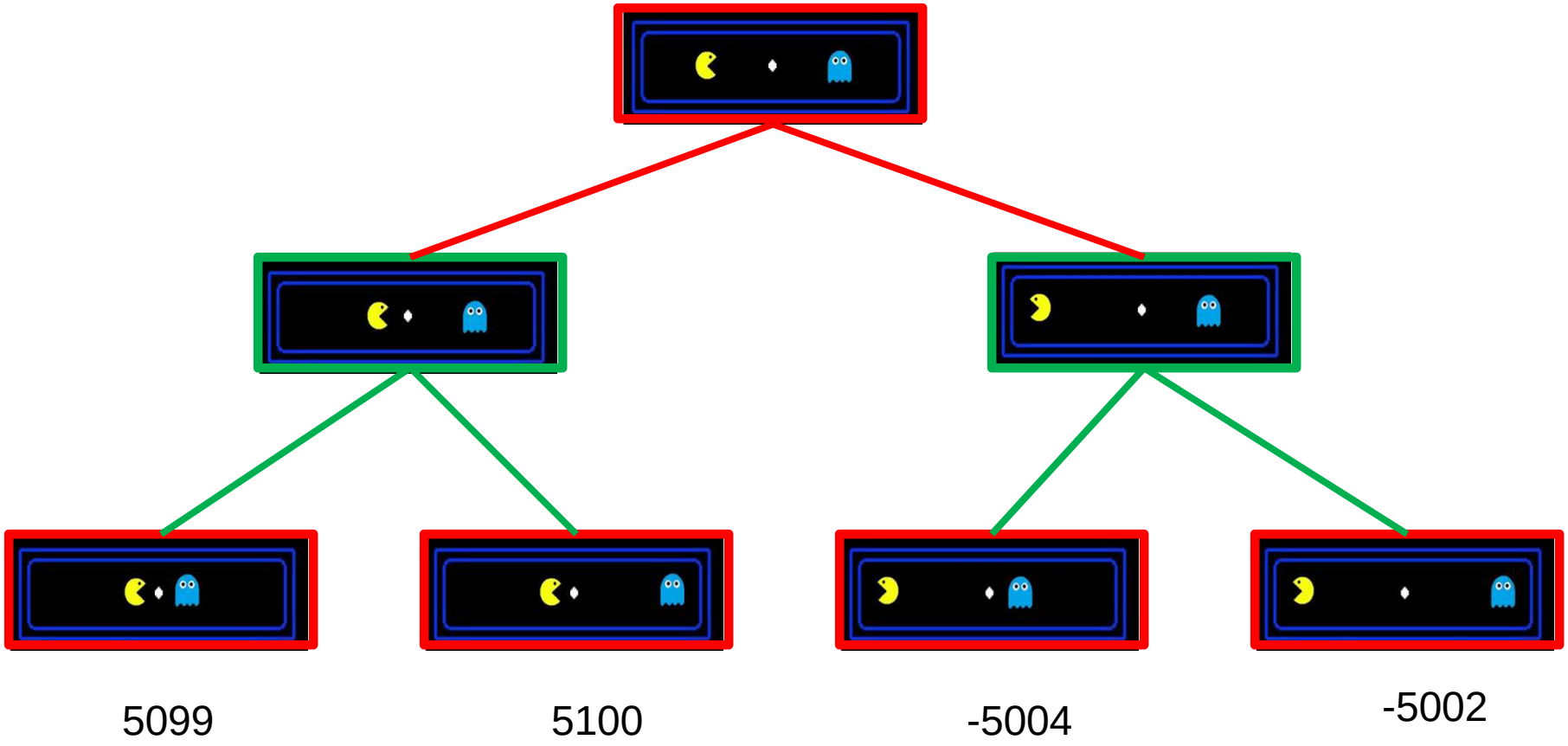
Flanking demo - Pac Man



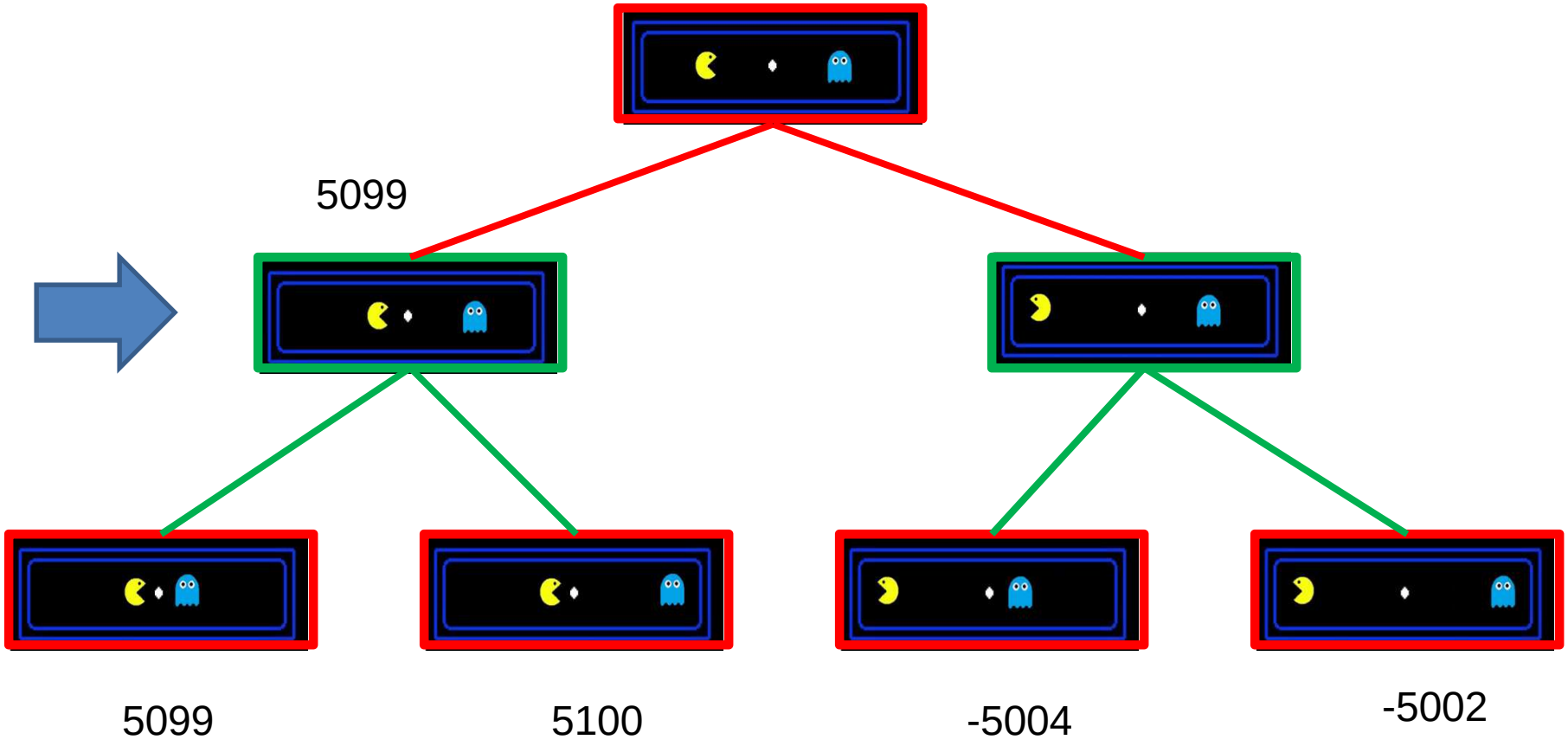
Alpha-Beta Pruning

- Extension of Minimax Algorithm.
- Reduce size of the game tree by pruning it.

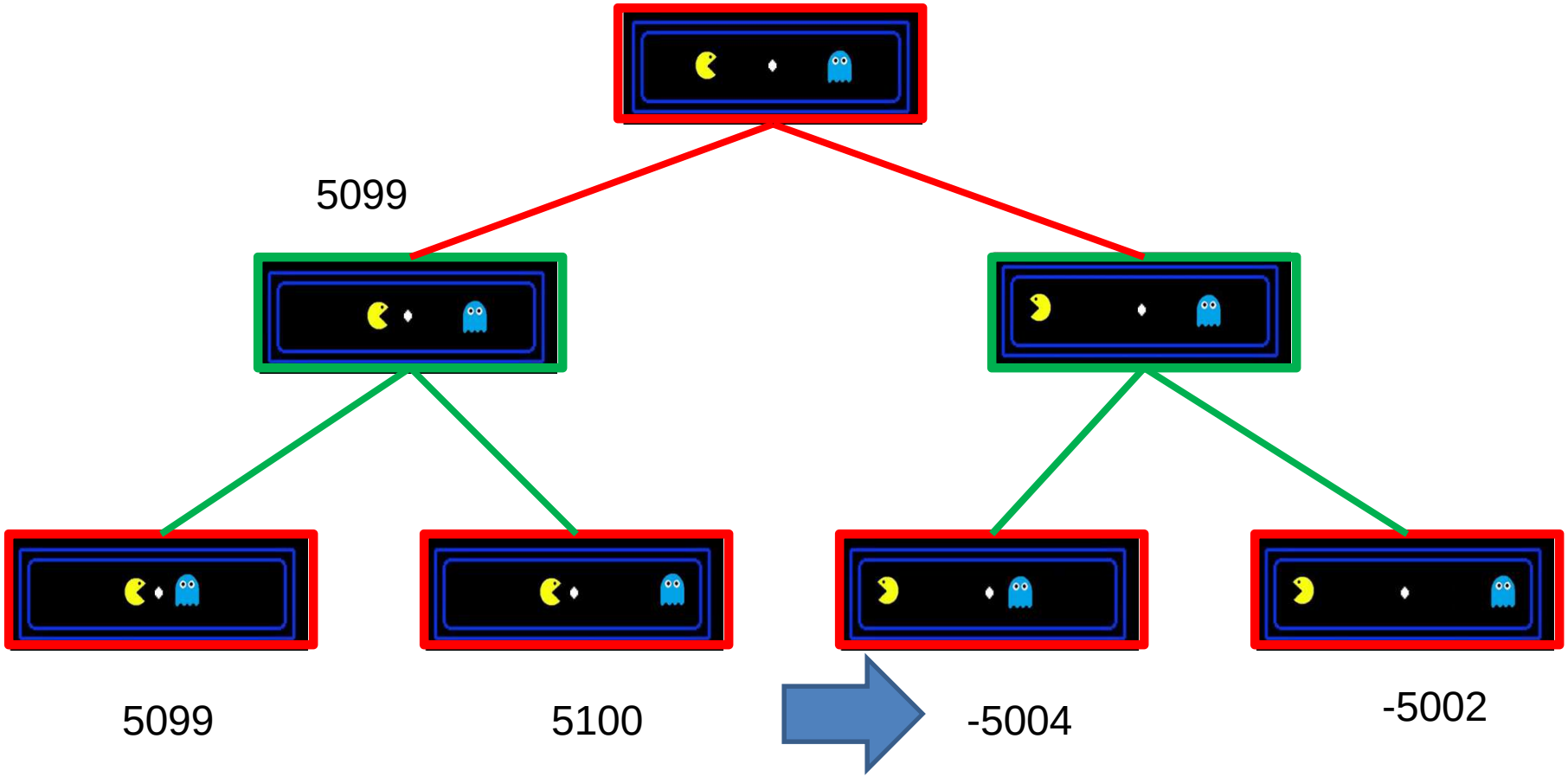
Alpha-Beta Pruning Simulation



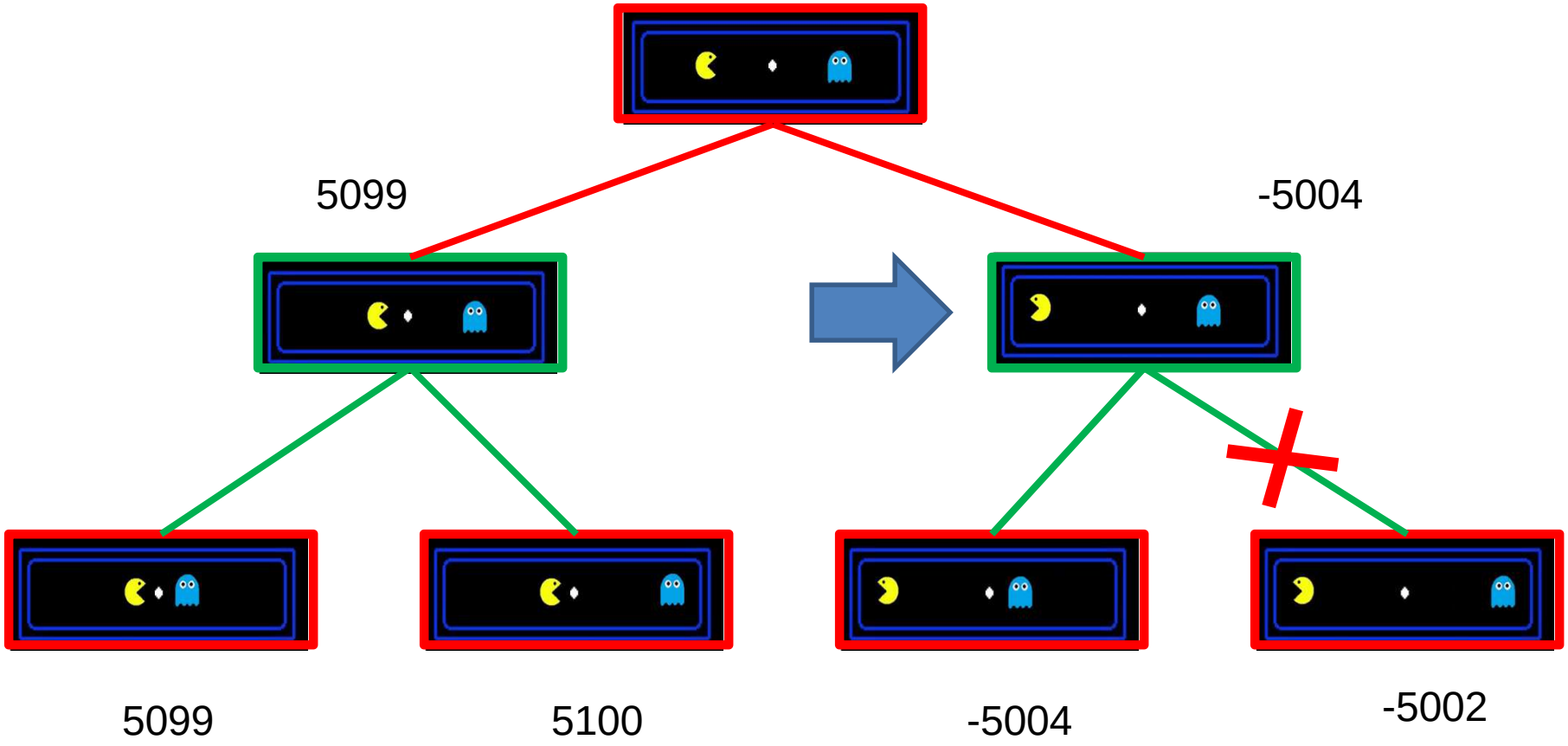
Alpha-Beta Pruning Simulation



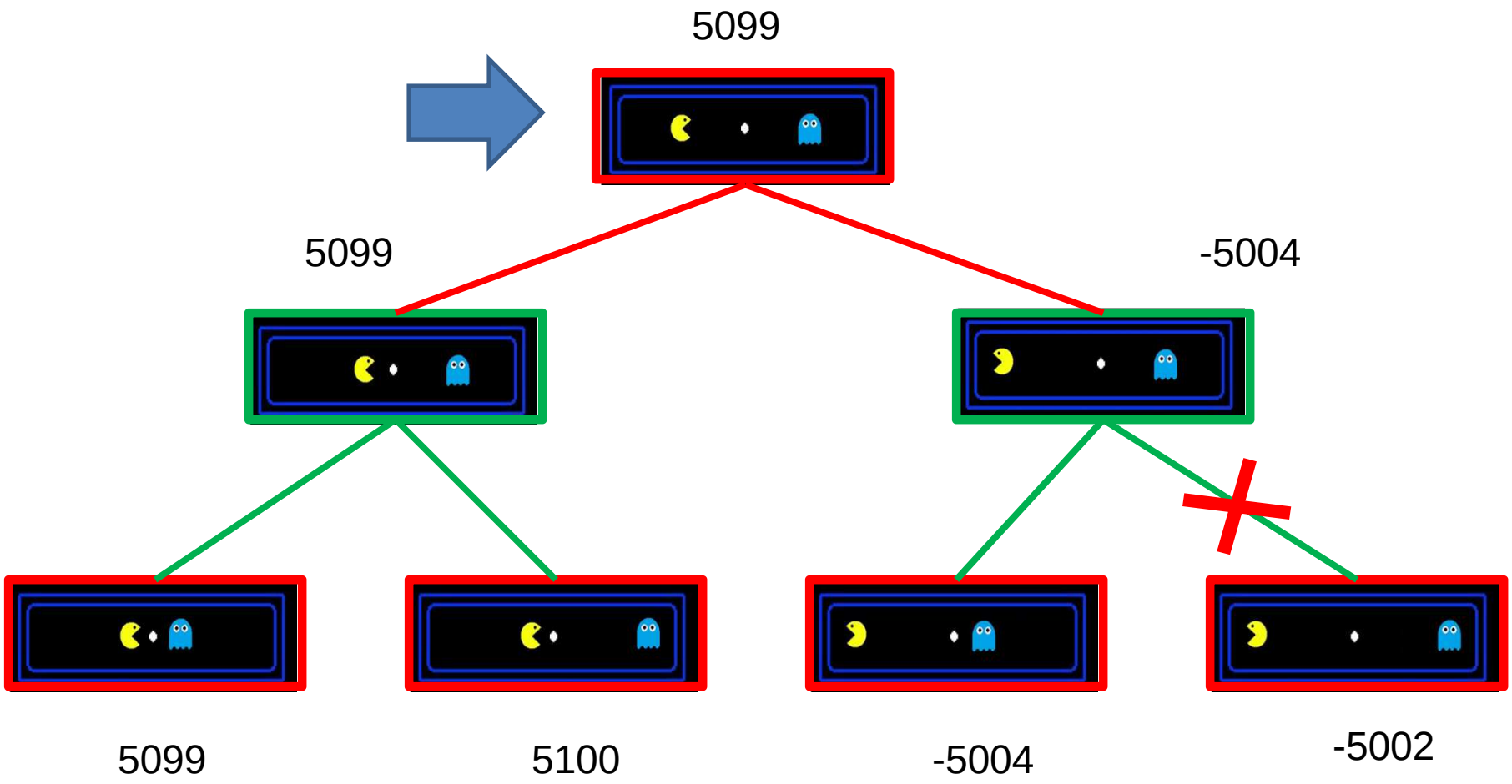
Alpha-Beta Pruning Simulation



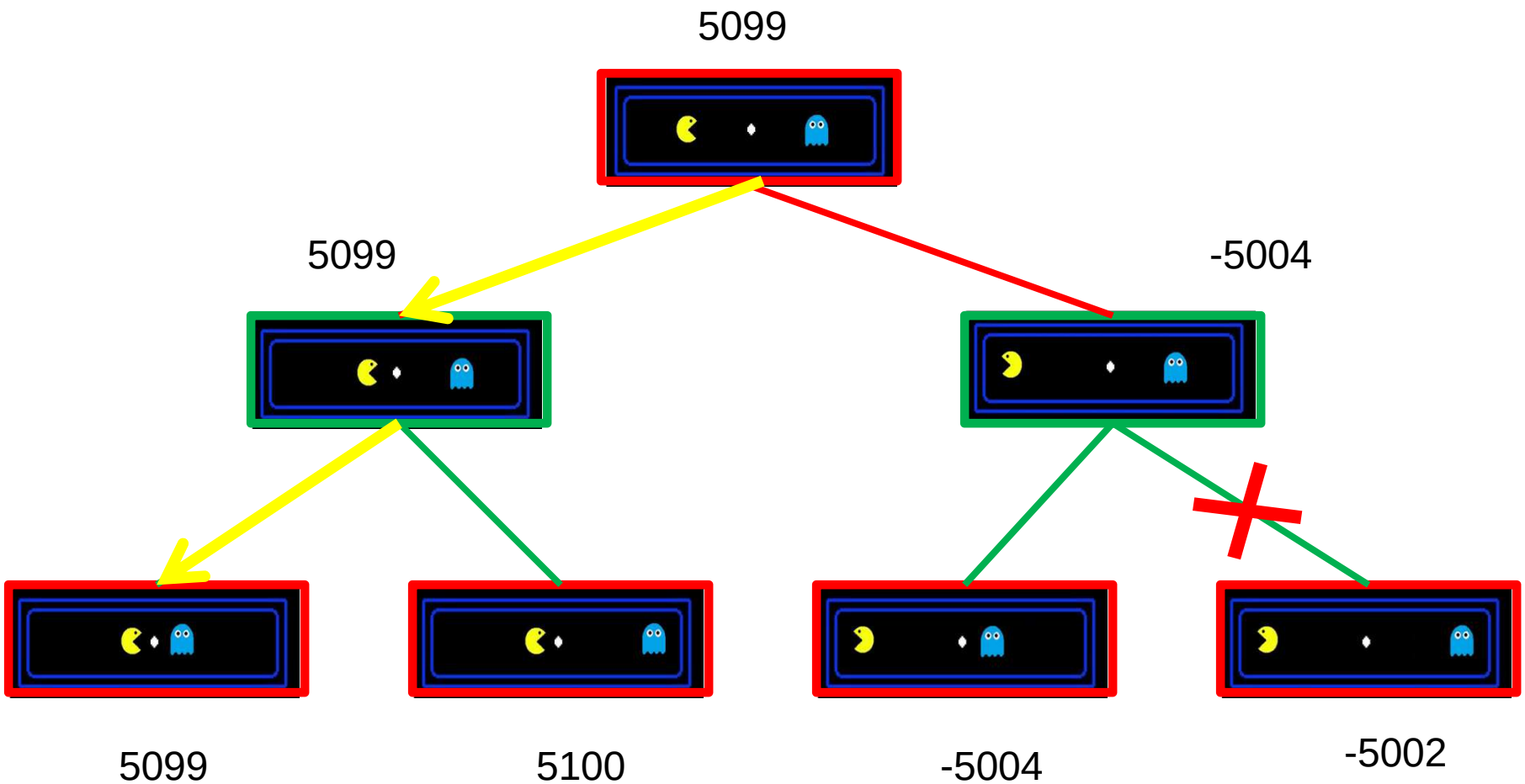
Alpha-Beta Pruning Simulation



Alpha-Beta Pruning Simulation

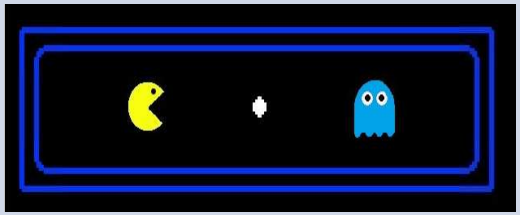
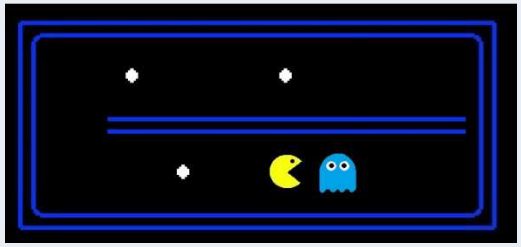
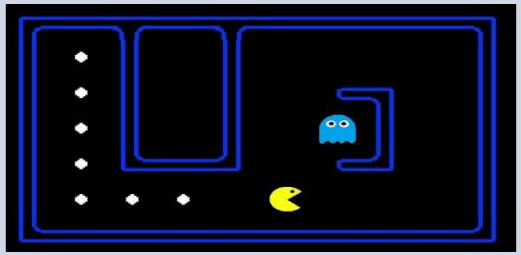


Alpha-Beta Pruning Simulation



Alpha Beta Pruning Demo

Comparing results

Map	Minimax Nodes analyzed	Alpha Beta Nodes analyzed	% nodes analysis reduction
	25	23	8%
	249	230	7.6%
	21074	10266	**51.2%**

Challenges

- Designing and implementing good utility function.

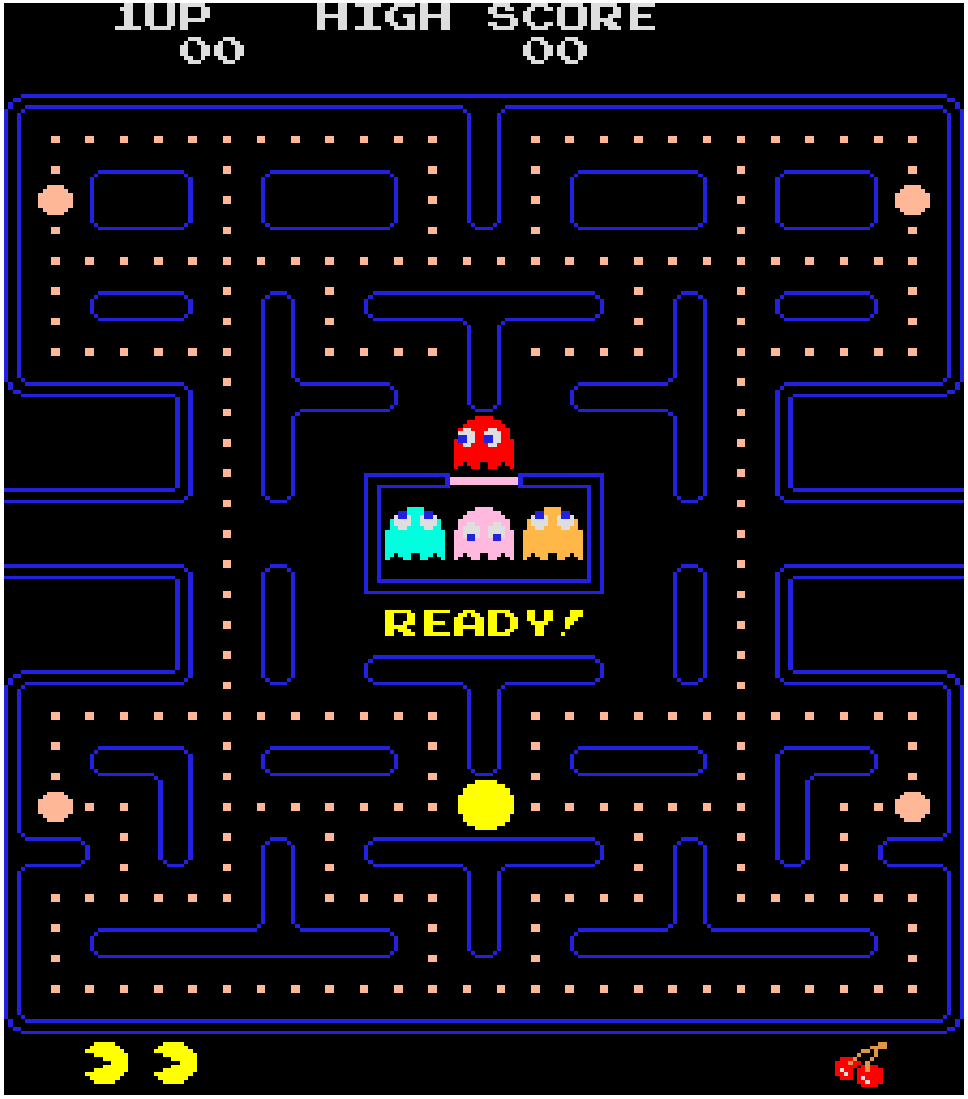
- Handling loop states.

  "These states were skipped to simplify game tree."

- Designing a well structured game.

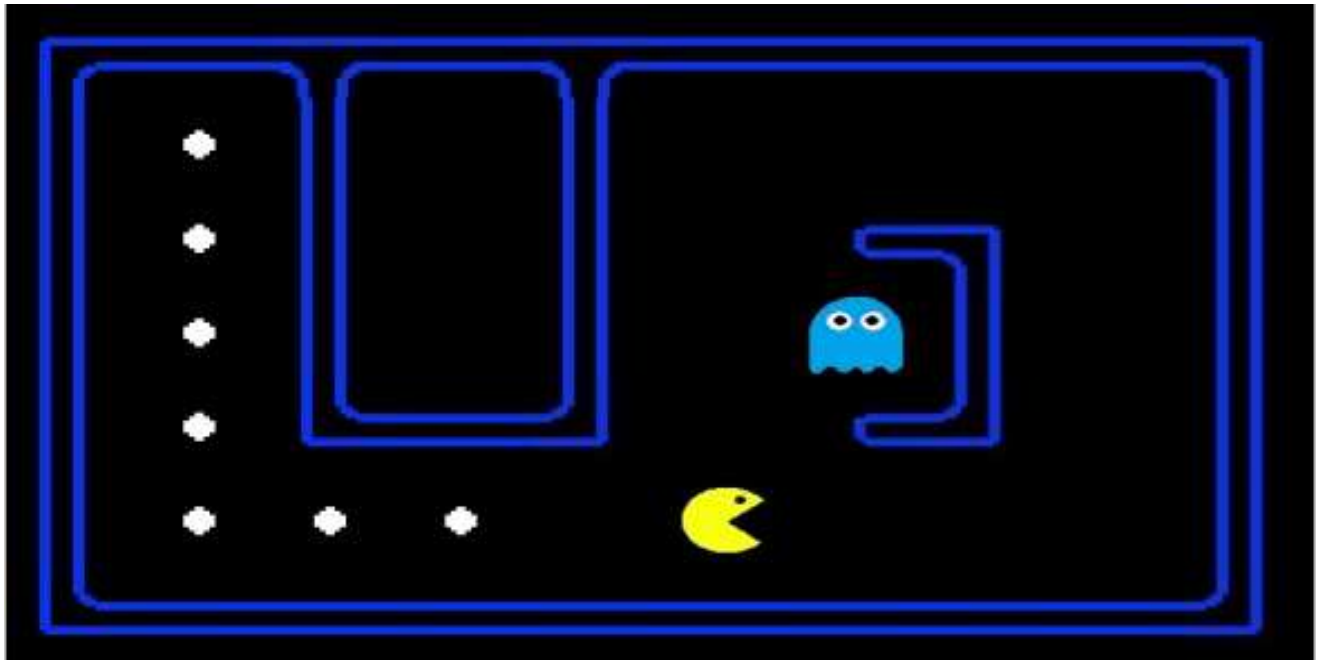
Possible future enhancements

- Full Implementation of the game.



Possible future enhancements

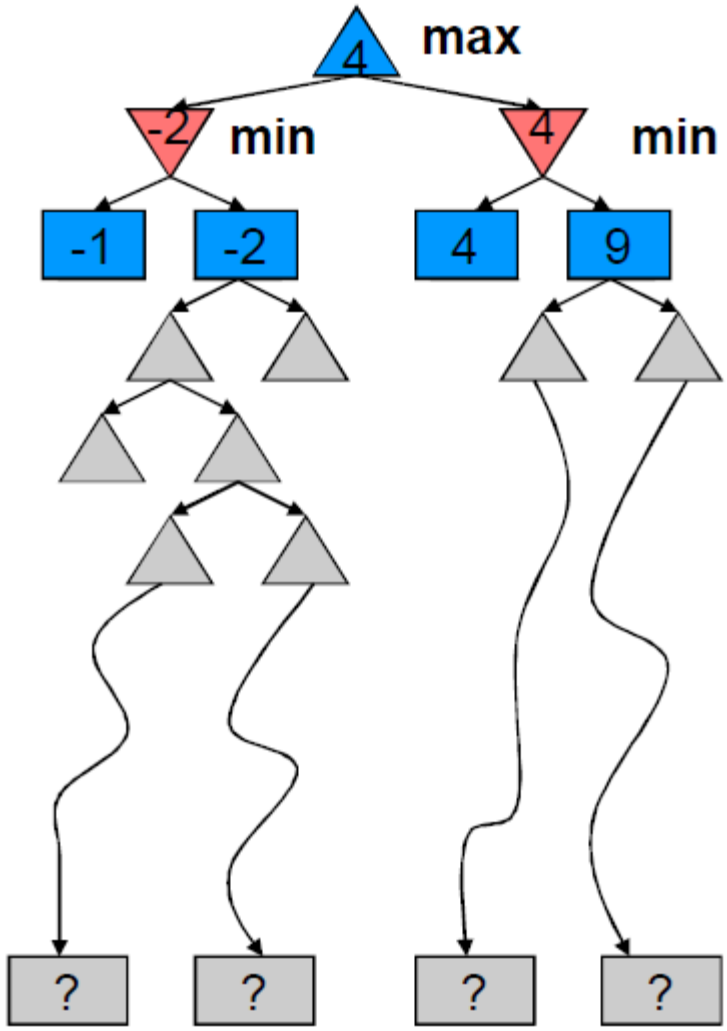
- Supporting **big maze** with **big game tree**.



Possible future enhancements

- Depth limited minimax/Alpha beta implementations.

- Better evaluation function for increased Game- Intelligence.



Conclusion

- Successful implementation of both Algorithms.
- Alpha beta performs better than minimax.

Thanks