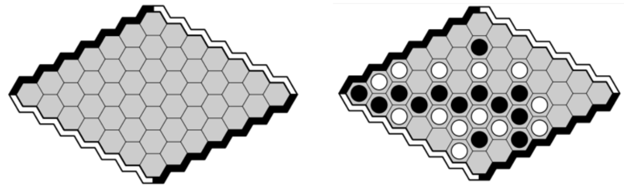
CMPUT 396 – Hex Game



Hex is played with 2 player, alternating moves. The goal is to connect your two sides together.

Properties:

- On any board, draws are not possible
- On an NxN board, there exists a winning strategy for the first player
- On an Nx(N+k) board (a non-square board), for the player with the shorter path across (whose sides are closer) there exists a winning strategy (whether they are the first or second player)
- Solving arbitrary Hex states (i.e. who wins) is P-space complete

History

Piet Hein invented the game in 1942 (then called Polygon).

It was independently re-invented in 1948 by mathematician John Nash at Princeton University. Nash explained the idea to David Gale, who introduced it to the Princeton game theory group together. Gale made a 14x14 board and left it in the Princeton common room. This was post-WWII when game theory was really popular (people thought it could help diffuse the Cold War).

In 1950 Claude Shannon (Bell Labs) built analog machines, called Shannon Machines, to play Hex and birdcage (played a reasonably good game of Hex). Shannon's machine was good for the beginning of the game, but not so much for tactical positions.

In 1952 the Parker Brothers marketed a version, which they called "Hex" and the name stuck.

Martin Gardner wrote about Hex in his Mathematical Games column in Scientific American in July 1957.

In 2013, Pawlewicz, Hayward et al. solved 2 openings for 10x10 Hex.

MoHex (Arneson, Hayward, Henderson, Huang, Pawlewicz et al) is the current strongest HexBot, stronger than most humans on sizes up to 13x13.

Solving Hex

How is this different from solving Tic-Tac-Toe? Do we need a new algorithm? Will alphabeta search still work?

Tic-Tac-Toe States:

- Number of states at most number of nodes in search tree: $\leq 9! = 362\,880$
  - $\leq$ because some terminal nodes not at max depth and transpositions
- Can we do better?
  Number of states at most number of different positions: $\leq 3^9 = 19683$
  - 9 empty board positions with 3 possibilities in each board position (empty, X, O)
- Can we do better?
  Many of these states are not reachable
  Many of these states are isomorphic
  - From root state only 3 non-iso moves, so 6500 nodes
  - Alphabeta search from the root examines 3025 nodes

Hex States:

- 3x3: $\leq 3^9 = 19683$
- 6x6: $\leq 3^{36} = 1.6 \times 10^{17}$
- 10x10: $\leq 3^{100} = 5 \times 10^{47}$
  - 10x10 number of states at most half full: 1.2×10^{43}
- In comparison: to solve checkers there are about 10^{20} positions
- Clearly, 10x10 Hex has too many states to search exhaustively
- So far there have been two 10x10 opening Hex moves solved:
  - Center
  - B9 (2nd position on obtuse-corner-to-obtuse-corner diagonal)

John Nash was excited when he came up with this game because Hex is a simple game and he could easily prove that whoever goes first can win.

The first player always has at least many stones as the opponent and they decide the start of the game.

If I don't lose, I will win. (Someone has to lose and someone has to win.)

John Nash's First Player Can Win Proof:

On an NxN board, there exists a winning strategy for the first player

Proof by contradiction:

I am P1. Assume that P2 has a winning strategy. For my first move, I will play anywhere and then forget about it. Then P2 plays. I will follow P2's winning strategy. My extra stone from the beginning doesn't hurt me, worst case I've already played there when I am following P2's strategy and then I play anywhere and forget about it again. Therefore, P1 has a winning strategy, which contracts our original assumption.

Another explanation from Stack Exchange:

The point is that in the game Hex, it never hurts to have an extra piece on the board.

So, suppose there is a strategy for the second player, but you are stuck with being the first player. What should you do?

Well, you can place a stone on the board, and then pretend in your own mind that it isn't there! In other words, you are imagining that the other player will now make the first move. In your mind, you are imagining that you are the second player now, and you can follow the winning strategy for the second player.

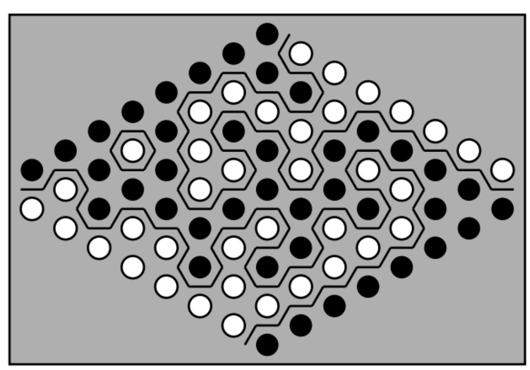
The only time you could have trouble is if your winning strategy tells you to place a move at the position you are pretending is empty. Since it's not really empty, you can't really make a move there. But luckily, you have already moved there, so you can imagine that you are making a move there right now -- the stone is already there, so you can imagine that you are just putting it there now. But in reality you still need to make a move, so you can do the same thing you did at the beginning -- just place a stone at some random position, and then imagine that you didn't.

The real state of the game is always just like your imagined state, except that there is an extra stone of yours on the board, which can't make things any worse. It limits the opponent's options, but if you have a winning strategy, it will work for any moves the opponent makes, so this isn't a problem either.

The conclusion is that, if there were a strategy for the second player to win, then you could "steal" that strategy as outlined above to win even when you are the first player. This is a contradiction, because if there were really a winning strategy for the second player, then the first player would not be able to guarantee a win. Therefore, there is not in fact any strategy for the second player to win.

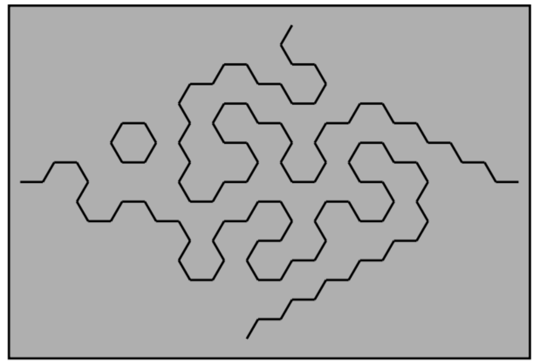
(link:https://math.stackexchange.com/questions/856436/john-nashs-hex-proof?fbclid=IwAR2CpGDDV-g9FefSaVSbTWwmnFNmk7khepUCXwwXfgTzH-Iy1sXyDFx2oGI)

Why Hex Cannot End In a Draw
On any board, draws are not possible



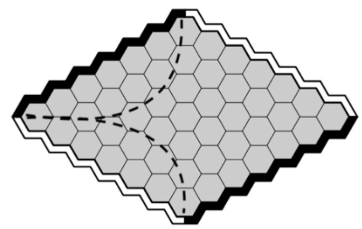
Take a board and put black on black's two sides and white on white's two sides. Put the rest of the stones on the board somewhere. Between any two cells that touch with different colours, draw a line segment between them.

Observation about line segments:
- We will always have black on the left of the line and white on the right
- Path cannot end in the middle of the board
- Path must end in one of the four corners (somewhere on the outside of the board)



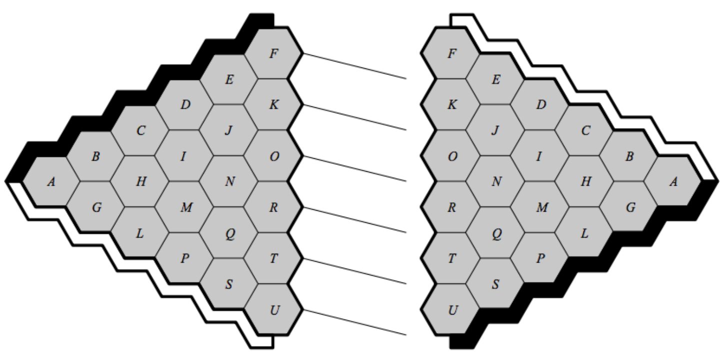
There are two places for the path to end up (starting from the top left corner):
- Surrounding black (upper right), or
- Surrounding white (lower left)

If the path goes from top left to lower left, black must have a path through the game board. If the path goes from top left to bottom right, white must have a path through the game board.



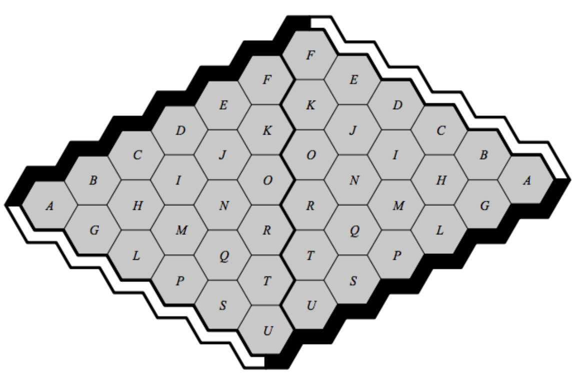
Nx(N+1) Hex: Longer-Side Win

On an Nx(N+k) board (a non-square board), for the player with the shorter path across (whose sides are closer) there exists a winning strategy (whether they are the first or second player)



For example, in this 7x6 board, break it into 2 triangles and label it so that the coordinates mirror each other.

White has the shorter path, so there exists a winning strategy on this board for white, whether White is the first or second player.



Say that white is P2. Black can move anywhere and then White should use a “pairing” strategy to “mirror” all of Black’s moves. In all cases, Black will lose.

In 2013, the strongest computer Hex player used alphabeta search.

What is a good algorithm? Correct, efficient, works in all cases

A good algorithm grows at most polynomially with the input. For example, $O(n^3)$ is a good algorithm, but $O(2^n)$ is not.

NP-Complete Problem

Solving arbitrary Hex states (i.e. who wins) is P-space complete

You can prove things about Hex fairly easily. You can prove that it's really hard to solve.

NP-Complete: non-deterministic polynomial. Problems are yes or no questions. If the answer turns out to be yes there is a proof that the answer is yes that can't be verified in polynomial time. You can show that solving your problem is at least as hard as every problem in NP-space.

P-problems: can be solved in polynomial time

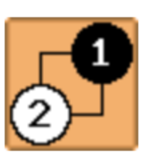
PSPACE: the set of all decision problems that can be solved by a Turing machine using a polynomial amount of space.

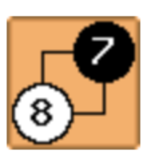
PSPACE-complete: if it can be solved using an amount of memory that is polynomial in the input length (polynomial space) and if every other problem that can be solved in polynomial space can be transformed to it in polynomial time. The problems that are PSPACE-complete can be thought of as the hardest problems in PSPACE because a solution to any one such problem could easily be used to solve any other problem in PSPACE.

If you can solve a PSPACE problem in polynomial time, you can solve any PSPACE problem in polynomial time. (This is a million dollar question.)

2x2 Go Board

What is the minimax score for a board like this? White can't move at some point because it will recreate a previous board position.





Really, move 8 should not be allowed because it recreates the first board position, so at this point, black will win by one point (with stone 7).

For Go, pass should always be the first move that you consider. This is because the opponent will also consider pass, which gives you the score for the game if it ends right then. After you have considered that, choose the best move.

Solving 2x2 Go: tromp.github.io/java/go/twoxtwo.html

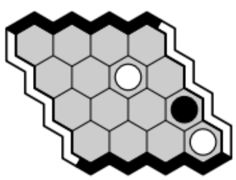
- *"The program below (download) solves the game of Go played on a 2x2 board using area rules and positional superko. It demonstrates the enormous importance of good move ordering in exhaustive alpha beta search. With the given ordering of passing first, only 1446 nodes are searched, to a depth of no more than 22. But trying passes after moves requires the search of as many as 19397529 nodes, to a depth of 58. Minimax, which doesn't depend on move ordering, takes over a week while searching a few trillion nodes."*

Considering pass first searches far less nodes (and in contrast, not passing searches far more nodes).

John Tromp says that the depth should be 22, but it's 11 here because the tree is not complete (but it's best for getting +1 score).

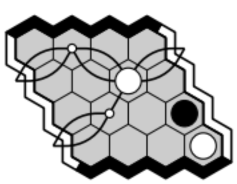
Back to Hex…

The search space for Hex is a lot smaller than for a lot of other games. Why? We can make massive amounts of pruning through **virtual connections** and **inferior cells**.



From this diagram alone we know that white will win. How?

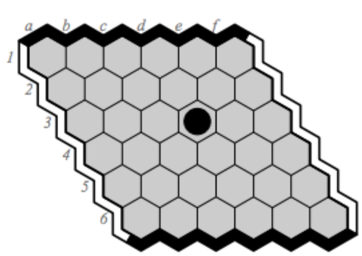
We can see that white has a lot of potential paths to take, lots of options. We could carve out a board subsection and it still represents a game.



The white stone is **virtually connected** to the white sides using various cells (look at the black paths from the white stone at the top).

This shows a winning strategy from this position.

What about on this 6x6 board?



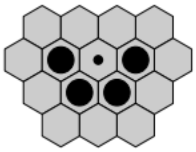
Black's move in the centermost cell is a winning move.

Where should white play to prevent black from immediately connecting to the top?

- D2 or E2? Both lead to black still being able to win two different ways
- E1? This is the intersection of the two moves above and it prevents black from making two winning paths, so it's better. But, black can respond at C2 and still gain two connections to the top edge after that.

Must-Play Analysis: Consider all your virtual connections. If you don't play in these connections, you are allowing your opponent to block you there.

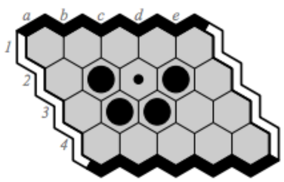
Inferior cells: cells that can be easily ruled out or ignored in the search for a single winning move. The black dot in the middle of this black group represents an inferior cell.



Imagine that there is a side-to-side white path that includes the black dot. This means that it must include the two cells above it, but those cells are already adjacent so even if it didn't have the black dot, it still has a path. Therefore, it can be ignored.

Why would black ever play there? It wouldn't, that's an awful move. We can use the same reasoning as before, you can already reach any other cell the black pieces could already reach.

Conclusion: this cell is useless for both players. It's a dead cell.



Is A3 a dead cell? It doesn't help black. If it's not dead, there has to be a path from side-to-side where if you don't have the cell, you no longer have the path. This also follows from the explanation above (look at a board edge as stones, and not a wall). If you place a stone in the dead zone, it doesn't change the winner.

Joinset: for a Hex position *P* and a player *X*, a joinset is a set of empty cells which, when *X*-coloured, joins *X*'s two sides (so for black, it's a set of empty cells which when coloured black, join black's two sides for the win). More simply, the set of cells that can connect two sides of the same colour.

A joinset is minimal if it is not a proper subset of some other joinset.

Algorithm to find dead cells:

- Take all shortest paths across the board.
    - The path must be disconnected if you remove any stones from this path.
- But wait, finding dead cells is NP-Complete.
- You could also use pattern matching.

So what should we do? Come up with ideas that are useful to solve Hex positions.

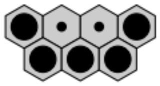
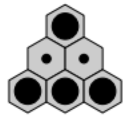
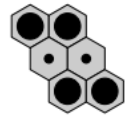
Solving 6x6 Go is an open problem (a 5x5 solution gave someone a PhD). We can solve 6x6 with what we know so far.

How Hayward Solved 7x7 Hex

This is in the *Puzzling Hex Primer* paper.

Captured cells were important. A cell is captured if for every opponent move in the set, the player has a response to prevent any winning strategy from occurring. This allows us to prune these cells.

Three common captured set patterns:

Is there any point in white playing in the small black dots of the earlier diagrams? Not really… unless black is not smart enough to respond. In other words, if white had a winning move, it wouldn't be there. So, we can assume that there are black cells to prevent searching/moving there.

This is all we need to create a Hex solver!

Permanently Inferior Cells: whenever you see the pattern on the left (black cells lining up in the left and bottom making an L-shape), you can add black stones as on the right. You can play in the center as the black player, but it doesn't really matter because it will become dead if someone plays at the top right corner.