Game playing

- **Types of games**
  - Deterministic vs. chance
  - Perfect vs. imperfect information

- **Active area of research**
  - Why?
    - Clear criteria for success
    - Interesting, hard problems
    - Fun

- **Typical game**
  - 2-player, *zero sum* game
  - Players alternate moves
  - Perfect information, no chance
  - Examples?

Rules:
1. Red goes first
2. On their turn, a player must move their piece
3. They must move to a neighboring square, or if their opponent is adjacent to them, with a blank on the far side, they can hop over them
4. The player that makes it to the far side first wins.
Classic problem: N queens

- Can queens be placed on a chess board so that no queens attack each other?
  - Easily place two queens
  - What about 8 queens?
- Make the board N×N, this is the N queens problem
  - Place one queen/column
  - # different tries/column?
- Backtracking
  - Use “current” row in a col
  - If ok, try next col
  - If fail, back-up, next row
Backtracking idea with N queens

- **Try to place a queen in each column in turn**
  - Try first row in column $C$, if ok, move onto next column
  - If solved, great, otherwise try next row in column $C$, place queen, move onto the next column
    - Must unplace the placed queen to keep going

- **What happens when we start in a column, where to start?**
  - If we fail, move back to previous column (which remembers where it is/failed)
  - When starting in a column anew, start at beginning
    - When backing up, try next location, not beginning

- **Backtracking in general, record an attempt go forward**
  - If going forward fails, undo the record and backup
Basic ideas in backtracking search

- **We need to be able to enumerate all possible choices/moves**
  - We try these choices in order, committing to a choice
  - If the choice doesn’t pan out we must undo the choice
    - This is the backtracking step, choices must be undoable

- **Process is inherently recursive, so we need to know when the search finishes**
  - When all columns tried in N queens
  - When we have found the exit in a maze
  - When every possible moved tried in Tic-tac-toe or chess?
    - Is there a difference between these games?

- **Summary: enumerate choices, try a choice, undo a choice, this is brute force search: try everything**
N queens backtracking: nqueens.cpp

bool Queens::SolveAtCol(int col)
    // pre: queens placed at columns 0,1,...,col-1
    // post: returns true if queen can be placed in column col
    //       and N queen problem solved (N is square board size)
{
    int k; int rows = myBoard.numrows();
    if (col == rows) return true;

    for(k=0; k < rows; k++) {
        if (NoQueensAttackingAt(k,col)) {
            myBoard[k][col] = true;   // place a queen
            if (SolveAtCol(col+1)) {
                return true;
            }
            myBoard[k][col] = false;  // unplace the queen
        }
    }

    return false;
}
Computer v. Human in Games

- Computers can explore a large search space of moves quickly
  - How many moves possible in chess, for example?
- Computers cannot explore every move (why) so must use heuristics
  - Rules of thumb about position, strategy, board evaluation
  - Try a move, undo it and try another, track the best move
- What do humans do well in these games? What about computers?
  - What about at Duke?
Backtracking, minimax, game search

- We’ll use tic-tac-toe to illustrate the idea, but it’s a silly game to show the power of the method
  - What games might be better? Problems?

- Minimax idea: two players, one maximizes score, the other minimizes score, search complete/partial game tree for best possible move
  - In tic-tac-toe we can search until the end-of-the game, but this isn’t possible in general, why not?
  - Use static board evaluation functions instead of searching all the way until the game ends

- Minimax leads to alpha-beta search, then to other rules and heuristics
Minimax for tic-tac-toe (see ttt.cpp)

- Players alternate, one might be computer, one human (or two computer players)

- Simple rules: win scores +10, loss scores -10, tie is zero
  - X maximizes, O minimizes

- Assume opponent plays smart
  - What happens otherwise?

- As game tree is explored is there redundant search?
  - What can we do about this?
```cpp
int Game::bestMove(Board::Player p, int & move)
{
    // check for game over or too deep in search first
    int best = (p == Board::X ? COMPUTER_WIN : HUMAN_WIN);
    int score;
    int dontCareMove;
    for(k=0; k < myBoard.size(); k++) {
        if (myBoard.isClear(k)) {      // can we move here?
            myBoard.place(k,p);
            score = bestMove(opposite(p),dontCareMove);
            myBoard.unplace(k);
            if (scoreIsBetter(score, best,p)) {
                best = score;
                move = k;
            }
        }
    }
    return best;
}
```
Caching or Memoization

- In Tic-Tac-Toe do we see the same board more than once?
  
  \[
  \begin{array}{ccc}
  X & O & . \\
  X & ? & . \\
  \cdot & \cdot & \cdot \\
  \end{array}
  \quad \begin{array}{ccc}
  X & ? & . \\
  X & O & . \\
  \cdot & \cdot & \cdot \\
  \end{array}
  \]

- Repercussions in terms of search tree?
  - Does avoiding search result in significant savings?
  - How can we easily do this? Hint: maps!

- Lessons applied more widely
  - More storage results in lower runtime, general tradeoff
  - Can we have too much of a good thing?
Heuristics

- Can do pruning - see alpha-beta
- World will still be too big
  - Checkers: ~$10^{40}$ states
  - Chess: ~$10^{120}$ states

- A heuristic is a rule of thumb, doesn’t always work, isn’t guaranteed to work, but useful in many/most cases
  - Search problems that are “big” often can be approximated or solved with the right heuristics

- Checkers: Chinook, Chess: Deep Blue, Othello: TD-gammon
Anita Borg 1949-2003

- “Dr. Anita Borg tenaciously envisioned and set about to change the world for women and for technology. … she fought tirelessly for the development technology with positive social and human impact.”

- “Anita Borg sought to revolutionize the world and the way we think about technology and its impact on our lives.”