Graphs and Search

- other searches in addition to breadth-first/depth-first
  - search in game trees
  - search in optimization problems
  - shortest path from vertex a to vertex b (or all vertices)

- consider Deep Blue (chess-playing computer, beat Kasparov)
  - specialized hardware
  - lots of application-specific knowledge
  - searched/analyzed 200,000,000 moves/sec (compared to Kasparov 3 moves/sec)
backtracking search: Knight’s Tour

- **Knight backtracking**
  - place a knight on an NxN chessboard, have the knight move, try to cover all squares on the board
  - idea: try each possible square
    - if square leads to a solution stop
    - otherwise try another square

- **Key Concepts:**
  - tentatively try a move
  - undo the move if it doesn’t work
Backtracking and Heuristic Search

- **Lots of possible moves, general schema is similar to depth first search**
  - keep going looking for a solution, if a path doesn’t work then back up, try again down another path

- **Heuristic Search**
  - order the moves according to a rule-of-thumb
  - try the moves “best-first”
  - priority queues are useful for storing potential moves
Code Issues

- templated classes functions
  - need access to template source, problem with .h and .cc

- Solutions to template problem
  - use #include “foo.cc” from foo.h (may need -frepo)
  - write all functions inline, part of foo.h

```cpp
template <class Kind> class Foo
{
    void getVal(){return myVal;}
}

use explicit instantiation:
```
```cpp
template class PQueue<string,int>;
```