Backtracking (images and music)

Nashville Bluegrass Band

Searching with no guarantees

- Search for best move in automated game play
  - Can we explore every move?
  - Are there candidate moves ranked by "goodness"?
  - Can we explore entire "tree" of possible moves?

- Search with all or partial information
  - Predictive texting with T9 or iTap or ...
  - What numbers fit in Sudoku square
  - Finding blobs on a grid

- Try something, if at first you don't succeed ....

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 NxN, this is the N queens problem
  - Place one queen/column
  - Horiz/Vert/Diag attacks
- Backtracking
  - Tentative placement
  - Recurse, if ok done!
  - If fail, undo tentative, retry

Backtracking idea with N queens

- For each column C, tentatively place a queen
  - Try first row in column C, if ok, move on to next column
    - Typically "move on" is recursive
  - If solved, done, otherwise try next row in column C
    - Must unplace queen when failing/unwind recursion
- Each column C "knows" what row R it's on
  - If first time, that's row zero, but might be an attack
  - Unwind recursion/backtrack, try "next" location
- Backtracking: record an attempt go forward
  - Move must be "undoable" on backtracking/unwinding
N queens backtracking: Queens.java

```java
public boolean solve(int col) {
    if (col == mySize) return true;
    // try each row until all are tried
    for (int r = 0; r < mySize; r++) {
        if (myBoard.safeToPlace(r, col)) {
            myBoard.setQueen(r, col, true);
            if (solve(col + 1)) return true;
            myBoard.setQueen(r, col, false);
        }
    }
    return false;
}
```

Run in debug mode, tracing execution

```java
for (int r = 0; r < mySize; r++) {
    if (myBoard.safeToPlace(r, col)) {
        myBoard.setQueen(r, col, true);
        if (solve(col + 1)) return true;
        myBoard.setQueen(r, col, false);
    }
}
return false;
```

Blob Counting, Flood Fill

- Flood a region with color
  - Erase region, make transparent, ...
  - How do we find the region?
- Finding regions, blobs, edges, ...
  - See blob counting code
  - What is a blob?
- Recursion helps, but necessary?
  - Performance, clarity, ...
  - Ease of development

BlobCount or edge detection or ...

- How do we find images? Components? Paths?
  - Create information from data
Details and Idioms in blob code

- **Method blobFill** has four parameters
  - (row,column) of where search starts
  - Character being searched for (initially * or blob)
  - Character to fill with on success (e.g., count '2' or '4')
    - Mark for visualization
    - Mark to ensure we don't search again!
  - If (row,column) is part of blob, count it and ask neighbors for their counts
    - They're part of blob (if never visited before)

- **Return total of yourself and neighbors**
  - Key to recursion: do one thing and ask for help

Blob questions

- **What changes if diagonal cells are adjacent?**
  - Conceptually and in code

- **How do we find blob sizes in a range?**
  - Not bigger than X, but between X and Y

- **How would we number blobs by size rather than by when they're found?**
  - Do we have the tools to do this in existing code?

- **Can we avoid recursion and do this iteratively?**

Search, Backtracking, Heuristics

- **How do you find a needle in a haystack?**
  - How does a computer play chess?
  - Why would you write that program?

- **How does Bing/Googlemap find routes from one place to another?**
  - Shortest path algorithms
  - Longest path algorithms

- **Optimal algorithms and heuristic algorithms**
  - When is close good enough? How do measure "closeness"?
  - When is optimality important, how much does it cost?

Exhaustive Search/Heuristics

- **We use binary search with arrays to organize data, in searching we don't need to examine all the data to find what we're looking for (sorted array)**
  - How do you eliminate half items with one guess?
  - Does it make sense to sort to do one search? Two? N?

- **What do we do when the search space is huge?**
  - How many chess boards are there?
  - Count routes between my house and yours?

- **Exhaustive search: look at everything!**
Basic ideas in backtracking search

- **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
    - Backtracking step, choices must be undoable

- **Inherently recursive, when to stop searching?**
  - When all columns tried in N queens
  - When we have found the exit in a maze
  - When every possible move 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**

Pruning vs. Exhaustive Search

- **If we consider every possible placement of 4 queens on a 4x4 board, how many are there? (N queens)**
  - $4 \times 4 \times 4 \times 4$ if we don’t pay attention to any attacks
  - $4 \times 3 \times 2 \times 1$ if we avoid attacks in same row

- **What about if we avoid diagonal attacks?**
  - Pruning search space makes more search possible, still could be lots of searching to do!

- **Estimate how long to calculate # solutions to the N-queens problem with our Java code…**

Tuomas Sandholm, CMU Professor

- **IJCAI Computers and Thought 2003**
  - Organ Network uses Carnegie Mellon algorithm to match live kidney donors with recipients
  - National Pilot Program facilitates kidney paired-donation transplants

- **http://bit.ly/9No5S5**
  - Using game theory, a group of computer scientists has developed a set of algorithms to help thwart terrorist attacks by randomizing where and when security checkpoints, officers, canine units and other deterrents are located in and around the airport.

Vince Conitzer @ cs.duke.edu

- **Vincent Conitzer will receive the 2011 Computers and Thought Award**
  - The award is presented every two years to the world’s leading AI researchers under the age of 35.
  - Conitzer, an assistant professor of computer science at Duke University, is receiving the award in recognition of his seminal work at the boundary of microeconomic theory and artificial intelligence, in particular for groundbreaking work on computational aspects of game theory, social choice, and mechanism design.