

# Admin

- ◊ Today's topics
  - More recursive backtracking examples
  - Pointers, recursive data
- ◊ Reading
  - pointers Ch 2.2-2.3
  - linked lists Ch 9.5 (sort of), handout #21
  - algorithms, big O Ch 7
- ◊ Assign 3 due Wed
- ◊ Tomorrow is SuperTuesday!

Lecture #11

# Backtracking pseudocode

```
bool Solve(configuration conf)
{
    if (no more choices) // BASE CASE
        return (conf is goal state);

    for (all available choices) {
        try one choice c;
        // solve from here, if works out, you're done
        if (Solve(conf with choice c made)) return true;
        unmake choice c;
    }

    return false; // tried all choices, no soln found
}
```

# Sudoku solver

- ◊ Arrange 1 to 9 with no repeats in row, col, or block

- Solve by recursive backtracking
- Not much logic, just brute-force

3	6	5	4	7	
5	2				
8	7			9	1
	9	1		8	
9	8	9	3		6
6	5	3	6	5	
1	3		2	5	
				7	4
	5	2	6	3	

- ◊ Cast as decision problem

- Each call will make one decision and recur on rest
- How many decisions do you have to make?
- What options do you have for each?

# Sudoku code

```
bool SolveSudoku(Grid<int> &grid)
{
    int row, col;

    if (!FindUnassignedLocation(grid, row, col))
        return true; // all locations successfully assigned!

    for (int num = 1; num <= 9; num++) { // options are 1-9
        if (NoConflicts(grid, row, col, num)) { // if # looks ok
            grid(row, col) = num; // try assign #
            if (SolveSudoku(grid)) return true; // recur if succeed stop
            grid(row, col) = UNASSIGNED; // undo & try again
        }
    }

    return false; // this triggers backtracking from early decisions
}
```

# Cryptarithmic

## ◆ Encrypted arithmetic puzzle

- Assign letter to digit (S = 4, E = 7, ...) so math is correct, each digit/letter used once
- Recognize the recursive cone!
  - Assign D E M N O R S Y to digits 0-9 is like building permutations of DEMNORSY-

$$\begin{array}{r} \text{SEND} \\ + \text{MORE} \\ \hline \text{MONEY} \end{array}$$

## ◆ Dumb, exhaustive strategy

- Find unassigned letter, assign digit
- Recur from there and see if solution works out
- If not, unmake assignment and try again

# Dumb solver

```
bool DumbSolve(puzzleT puzzle, string lettersToAssign)
{
    if (lettersToAssign == "")
        return PuzzleSolved(puzzle);

    for (int digit = 0; digit <= 9; digit++) {
        if (AssignLetter(lettersToAssign[0], digit)) {
            if (DumbSolve(puzzle, lettersToAssign.substr(1))) return true;
            UnassignLetter(lettersToAssign[0], digit);
        }
    }
    return false; // nothing worked, need to backtrack
}
```

# Smarter solver

## ◆ Not all permutations plausible!

- Don't waste time on ridiculous choices

## ◆ Use grade-school addition knowledge

- Start with lastmost column (least significant digit)
  - Assign 'D', then assign 'E', now consider 'Y'
  - Assign 'Y' value so math works out (if impossible, fall here)
  - Recur on next column

## ◆ Heuristics

- Avoids niggling around in dead ends
- Choose more likely options to explore first
- Eliminate obvious bad choices

$$\begin{array}{r} \text{SEND} \\ + \text{MORE} \\ \hline \text{MONEY} \end{array}$$

# Looking for patterns

## ◆ Knapsack filling

- Sack holds 50 lbs, which items to select for highest value?

## ◆ Traveling salesman

- Visit 10 cities, how to cover shortest total distance?

## ◆ Dividing into fair teams

- Equal total team IQ? :-)

## ◆ Finding hidden words

- Richard Milhaus Nixon -> "criminal"

# Pointers

- A pointer is an *address*
  - All data is stored in memory
  - Each location in memory is indexed by address
  - Can refer to data by using its address in memory
- Why use pointers?
  - Provide shared access to common data
  - Build flexible, dynamic data structures
  - Precisely control allocation/deallocation
- Why are pointers considered scary?
  - Operations can be error-prone
  - Pointer mistakes have wide variation in symptoms
  - Memory bugs can be hard to understand and fix

# Simple pointer operations

```
int main()
{
    int num;
    int *p, *q;

    p = new int;
    *p = 10;

    q = new int;

    *q = *p;

    q = p;

    delete p;
    delete q; // bad idea, q already deleted!

    q = NULL;
```