Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)

Newspapers and magazines often have crypt-arithmetic puzzles of the form:

```
  SEND
+ MORE
--------
 MONEY
--------
```

The goal here is to assign each letter a digit from 0 to 9 so that the arithmetic works out correctly. The rules are that all occurrences of a letter must be assigned the same digit, and no digit can be assigned to more than one letter.

- First, create a list of all the characters that need assigning to pass to Solve
- If all characters are assigned, return true if puzzle is solved, false otherwise
- Otherwise, consider the first unassigned character
- for (every possible choice among the digits not in use)

  make that choice and then recursively try to assign the rest of the characters
  if recursion sucessful, return true
  if !successful, unmake assignment and try another digit

- If all digits have been tried and nothing worked, return false to trigger backtracking

```
/* ExhaustiveSolve
* ---------------
* This is the "not-very-smart" version of cryptarithmetic solver. It takes
* the puzzle itself (with the 3 strings for the two addends and sum) and a
* string of letters as yet unassigned. If no more letters to assign
* then we've hit a base-case, if the current letter-to-digit mapping solves
* the puzzle, we're done, otherwise we return false to trigger backtracking
* If we have letters to assign, we take the first letter from that list, and
* try assigning it the digits from 0 to 9 and then recursively working
* through solving puzzle from here. If we manage to make a good assignment
* that works, we've succeeded, else we need to unassign that choice and try
* another digit. This version is easy to write, since it uses a simple
* approach (quite similar to permutations if you think about it) but it is
* not so smart because it doesn't take into account the structure of the
* puzzle constraints (for example, once the two digits for the addends have
* been assigned, there is no reason to try anything other than the correct
* digit for the sum) yet it tries a lot of useless combos regardless
*/
bool ExhaustiveSolve(puzzleT puzzle, string lettersToAssign)
{
    if (lettersToAssign.empty()) // no more choices to make
        return PuzzleSolved(puzzle); // checks arithmetic to see if works
    for (int digit = 0; digit <= 9; digit++)   // try all digits
    {
        if (AssignLetterToDigit(lettersToAssign[0], digit))
        {
            if (ExhaustiveSolve(puzzle, lettersToAssign.substr(1)))
                return true;
            UnassignLetterFromDigit(lettersToAssign[0], digit);
        }
    }
    return false;  // nothing worked, need to backtrack
}
```

The algorithm above actually has a lot in common with the permutations algorithm, it pretty much just creates all arrangements of the mapping from characters to digits and tries each until one works or all have been successfully tried. For a large puzzle, this could take a while.
A smarter algorithm could take into account the structure of the puzzle and avoid going down dead-end paths. For example, if we assign the characters starting from the ones place and moving to the left, at each stage, we can verify the correctness of what we have so far before we continue onwards. This definitely complicates the code but leads to a tremendous improvement in efficiency, making it much more feasible to solve large puzzles.

Below pseudocode in this case has more special cases, but the same general design

- Start by examining the rightmost digit of the topmost row, with a carry of 0

- If we are beyond the leftmost digit of the puzzle, return true if no carry, false otherwise
- If we are currently trying to assign a char in one of the addends
  If char already assigned, just recur on row beneath this one, adding value into sum
  If not assigned, then
    - for (every possible choice among the digits not in use)
      make that choice and then on row beneath this one, if successful, return true
      if !successful, unmake assignment and try another digit
    - return false if no assignment worked to trigger backtracking
- Else if trying to assign a char in the sum
- If char assigned & matches correct,
  recur on next column to the left with carry, if success return true,
- If char assigned & doesn't match, return false
- If char unassigned & correct digit already used, return false
- If char unassigned & correct digit unused,
  assign it and recur on next column to left with carry, if success return true
- return false to trigger backtracking

Source:
http://see.stanford.edu/materials/icspacs106b/H19-RecBacktrackExamples.pdf

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Backtracking

Writing code in comment? Please use **ideone.com** and share the link here.

Mayank • 8 months ago
Disclaimer: The code contains some redundancies.
http://ideone.com/jvZq3i

Pavi • a year ago

I couldn't find out where to raise this issue, hence landing here. Sorry! for that.
I want to suggest a small change in your site in the way the ad page opens in same window when clicked on a Ad url.
Request you to change your site's code so that URLs when clicked (especially Ad urls) must open in a seperate tab or window and not on the same page. Thanks! if you consider this.

2 ▲ | ▼ • Reply • Share ›

rajeev • a year ago

great, could you provide complete implementation?

▲ | ▼ • Reply • Share ›

sateeshkkm ➚ rajeev • 3 months ago

Here come complete solution in C#

```
private static bool SolveCrypto(Puzzle puzzle, string letters, int[] arr)

{

if (string.IsNullOrEmpty(letters))

return IsPuzzelSolved(puzzle, arr);

for (int i = 0; i <= 9; i++)

{

if(AssignLetterToDigit(letters[0], i, arr))

{

if (SolveCrypto(puzzle, letters.Substring(1), arr))

return true;
```

see more

1 ▲ | ▼ • Reply • Share ›

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Recent Comments

- Ashish Aggarwal

  Try Data Structures and Algorithms Made Easy -...

  Algorithms · 17 minutes ago

- Vlad

  Thanks. Very interesting lectures.

  Expected Number of Trials until Success · 1 hour ago

- cfh

  My implementation which prints the index of the...

  Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

- Gaurav pruthi

  forgot to see that part ;)

  Bloomberg Interview | Set 1 (Phone Interview) · 1 hour ago

- saeid aslami

  thanks

  Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 1 hour ago

- Cracker

  Implementation:...

  Implement Stack using Queues · 2 hours ago