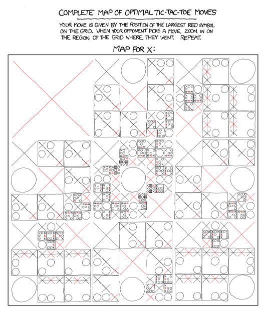
Adam Blank | Lecture 17 | Autumn 2016

CSE 143

Computer Programming II

Recursive Backtracking
Recursive Backtracking


COMPLETE MAP OF OPTIMAL TIC-TAC-TOE MOVES
YOUR MOVE IS GIVEN BY THE POSITION OF THE LARGEST RED SYMBOL ON THE GRID. WHEN YOUR OPPONENT PICKS A MOVE, ZOOM IN ON THE REGION OF THE GRID WHERE THEY WENT. REPEAT.
MAP FOR X:

Outline

1 NQueens

2 Sentence Splitter

Recursive Backtracking

Definition (Recursive Backtracking)

Recursive Backtracking is an attempt to find solution(s) by building up partial solutions and abandoning them if they don't work.

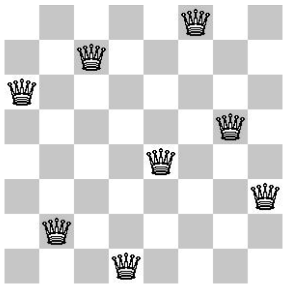
Recursive Backtracking Strategy

- If we found a solution, stop looking (e.g. return)
- Otherwise for each possible choice c...
  - Make the choice c
  - Recursively continue to make choices
  - Un-make the choice c (if we got back here, it means we need to continue looking)

NQueens Problem

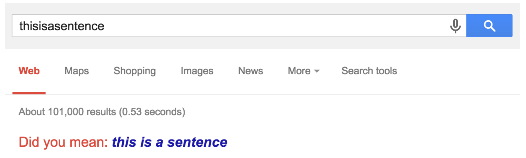
The **NQueens** problem is the challenge to place n queens on a chess board so that none of them are attacking each other.

We will begin by solving this problem using for loops, and then we will solve it much more elegantly using recursive backtracking.



Implementing a Tiny Piece of Google

When you enter a query with no spaces like **thisisasentence** into Google:



thisisasentence

Web | Maps | Shopping | Images | News | More | Search tools

About 101,000 results (0.53 seconds)

Did you mean: ***this is a sentence***

It fixes it into **this is a sentence** using recursive backtracking.

Sentence Splitting

Given an input string, sentence, containing **no spaces**, write a method:

```
public static String splitSentence(String sentence)
```

that returns sentence split up into words.

Sentence Splitting

Sentence Splitting

Given an input string, `sentence`, containing **no spaces**, write a method:

```
public static String splitSentence(String sentence)
```

that returns `sentence` split up into words.

To do recursive backtracking, we need to answer these questions:

- What are the choices we're making incrementally?
  ...which character to split at
- How do we "undo" a choice?
  ...re-combine a string by the char we split at
- What are the base case(s)?
  ...our left choice isn't a word **and** our right choice IS a word

It helps to answer these questions for a particular input. So, pretend we're working with:

thisisasentence

One More Important Choice

When doing recursive backtracking, we need to differentiate between:

- finding a result
- failing to find a result (e.g., backtracking)

Generally, we do this by treating `null` as a failure. For example:

- On the input, "**thisisasentence**", none of the recursive calls should return "**thisis**", because it isn't a word.
- If we get down to an empty string, that would indicate a failure; so, we'd return `null`

Sentence Splitter Solution

```java
public String splitSentence(String sentence) {
   // The entire sentence is a dictionary word!
   if (words.contains(sentence)) {
      return sentence;
   }

   // Try splitting at every character until we find one that works...
   for (int i = sentence.length() - 1; i > 0; i--){
      String left = sentence.substring(0, i);
      String right = sentence.substring(i, sentence.length());

      // If the left isn't a word, don't bother recursing.
      // If it is, split the remainder of the sentence recursively.
      if (words.contains(left)) {
         right = splitSentence(right);
         // Since the left was a word, if the right is also an answer,
         // then we found an answer to the whole thing!
         if (right != null) {
            return left + " " + right;
         }

         // Undo our choice by going back to sentence
      }
   }
   return null;
}
```