Faculty of Computer and Artificial Intelligence
Sadat University

Analysis and Design of Algorithms (CS 302)

Lecture :3

“BackTracking”

Dr.Sara A Shehab

Backtracking

INTRODUCTION TO BACKTRACKING

A short list of categories

Algorithm types we will consider include:

- Simple recursive algorithms
- Backtracking algorithms
- Divide and conquer algorithms
- Dynamic programming algorithms
- Greedy algorithms
- Branch and bound algorithms
- Brute force algorithms
- Randomized algorithms

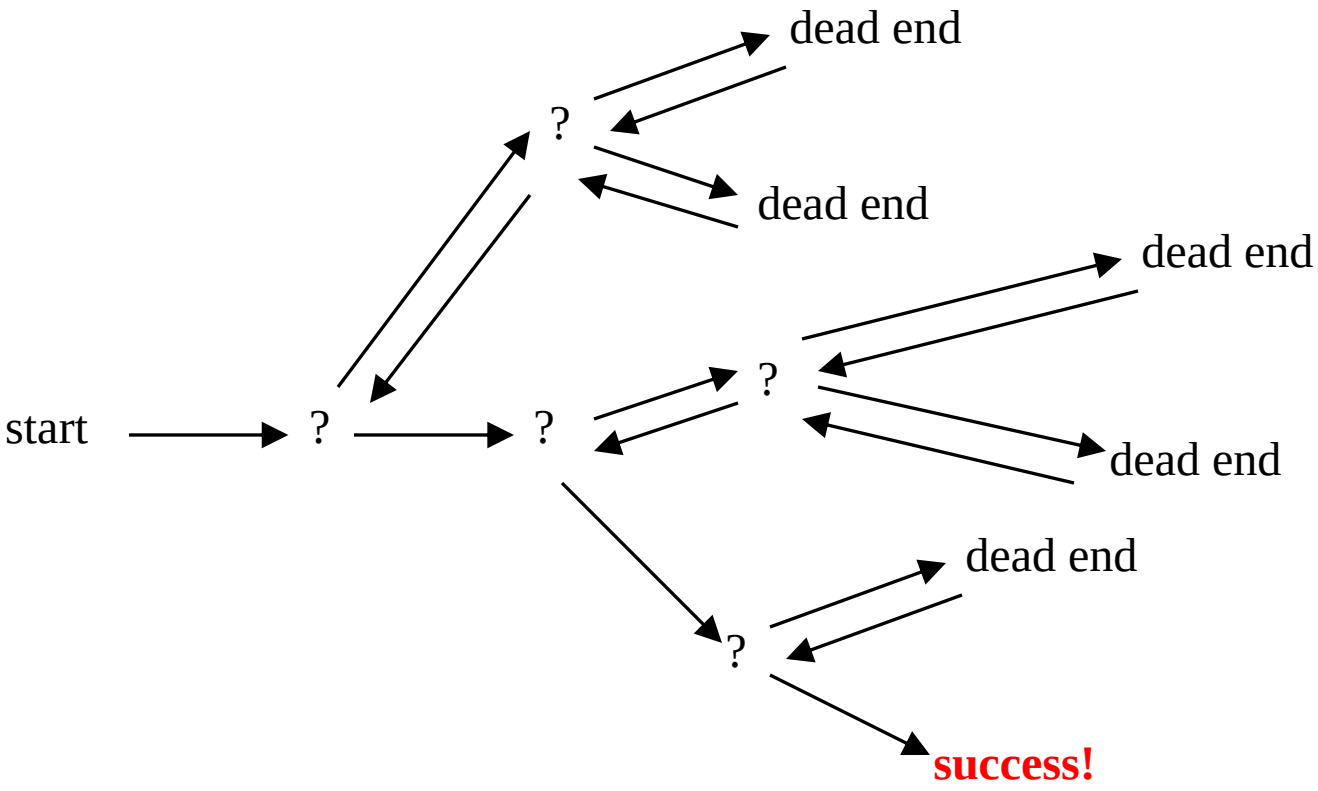
Backtracking

Suppose you have to make a series of *decisions,* among various *choices,* where

- You don't have enough information to know what to choose
- Each decision leads to a new set of choices
- Some sequence of choices (possibly more than one) may be a solution to your problem

Backtracking is a methodical way of trying out various sequences of decisions, until you find one that "works"

Backtracking (animation)



The backtracking algorithm

Backtracking is really quite simple--we “explore” each node, as follows:

To “explore” node N:

1. If N is a goal node, return “success”
2. If N is a leaf node, return “failure”
3. For each child C of N,
   3.1. Explore C
      3.1.1. If C was successful, return “success”
4. Return “failure”

Solving a maze

Given a maze, find a path from start to finish

At each intersection, you have to decide between three or fewer choices:

- Go straight
- Go left
- Go right

You don't have enough information to choose correctly

Each choice leads to another set of choices

One or more sequences of choices may (or may not) lead to a solution

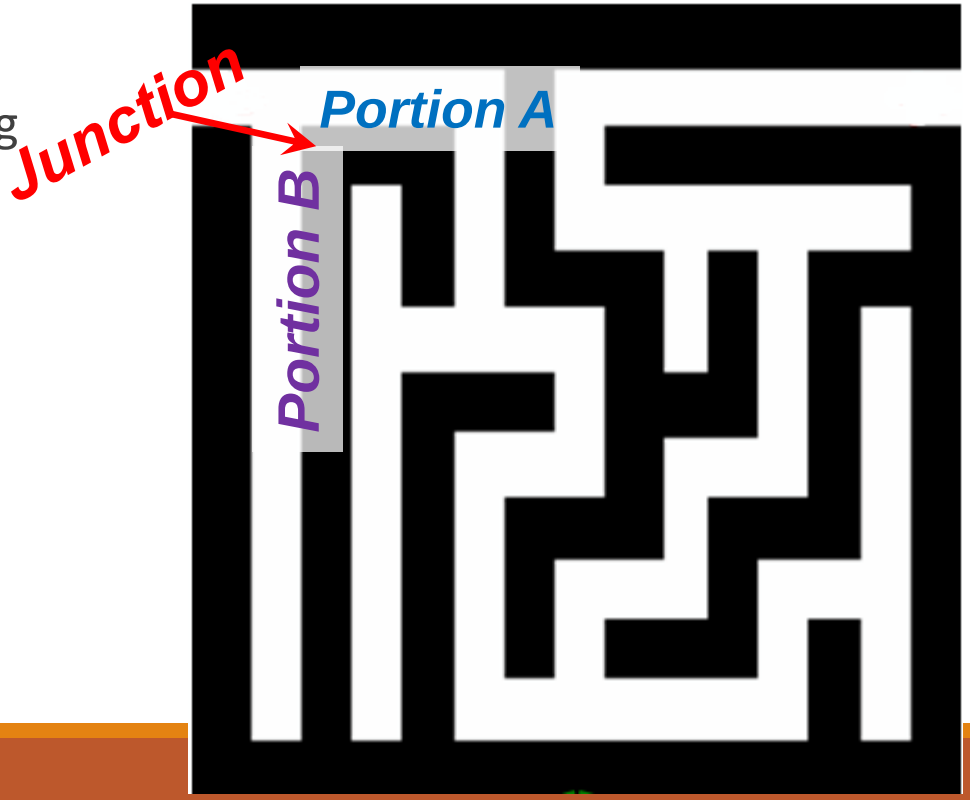
Many types of maze problem can be solved with backtracking

Backtracking

Backtracking is a technique used to solve problems with a large search space, by systematically trying and eliminating possibilities.

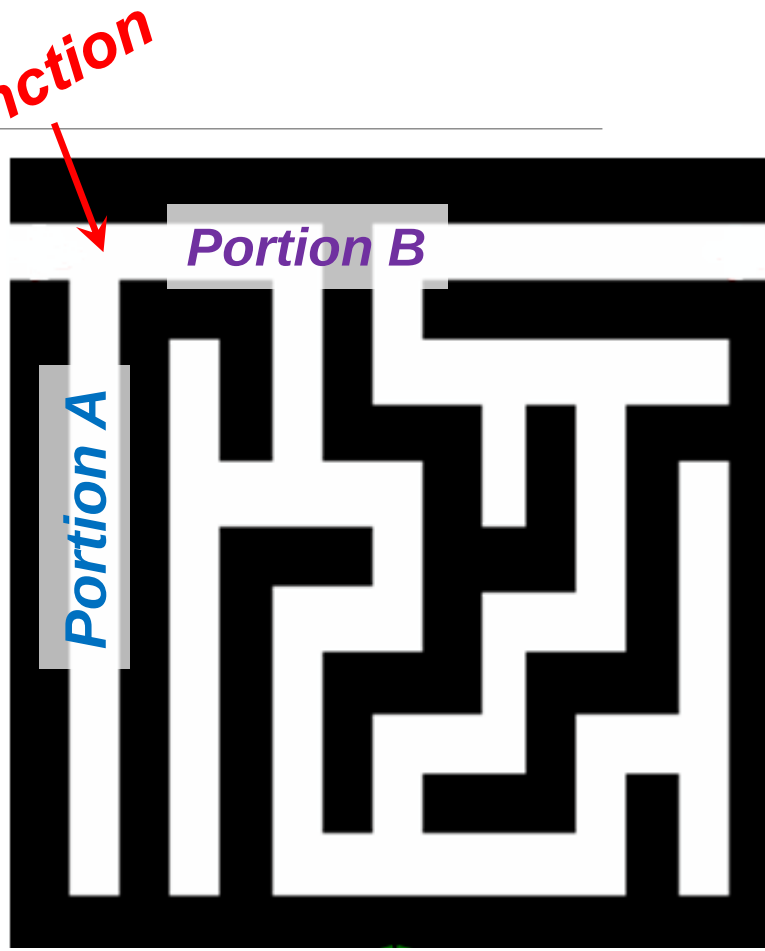
A standard example of backtracking would be going through a maze.

- At some point in a maze,
- you might have two
- options of which direction to go:



Backtracking

- One strategy would be to try going through **Portion A** of the maze.
  - If you get stuck before you find your way out, then you ***"backtrack"*** to the junction.
- At this point in time you know that **Portion A** will ***NOT*** lead you out of the maze,
  - so you then start searching in **Portion B**



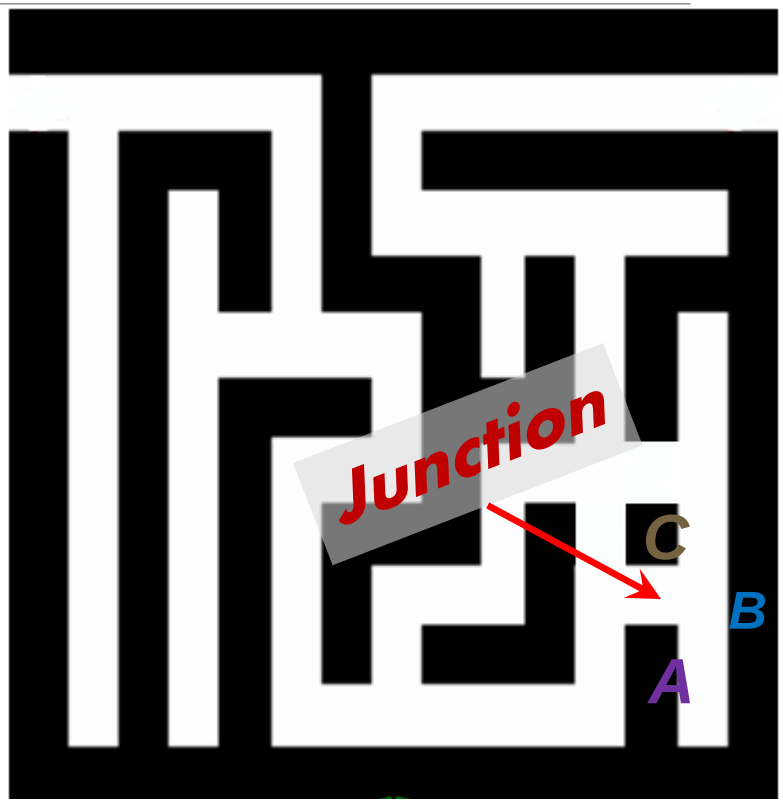
Backtracking

Clearly, at a single junction you could have even more than 2 choices.

The backtracking strategy says to try each choice, one after the other,

- if you ever get stuck, ***"backtrack"*** to the junction and try the next choice.

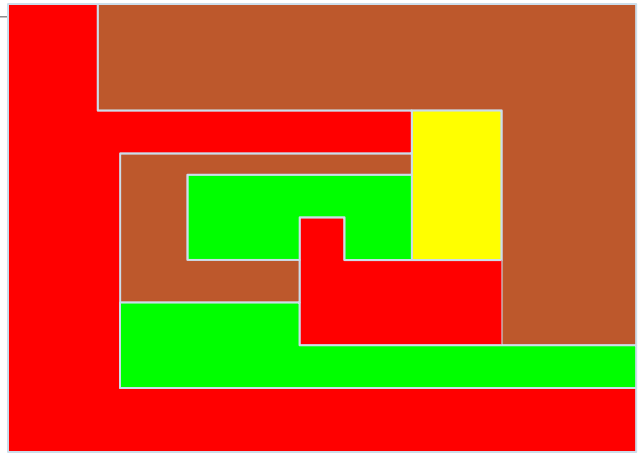
If you try all choices and never found a way out, then there IS no solution to the maze.



Coloring a map

You wish to color a map with not more than four colors

- red, yellow, green, blue



Adjacent countries must be in different colors

You don’t have enough information to choose colors

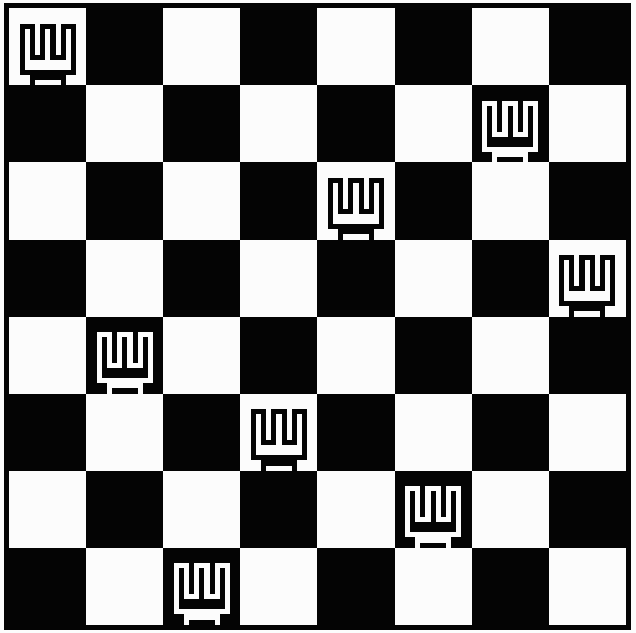
Each choice leads to another set of choices

One or more sequences of choices may (or may not) lead to a solution

Many coloring problems can be solved with backtracking

Backtracking – Eight Queens Problem

Find an arrangement of **8** queens on a single chess board such that no two queens are attacking one another.



In chess, queens can move all the way down any row, column or diagonal (so long as no pieces are in the way).

- Due to the first two restrictions, it's clear that each row and column of the board will have exactly one queen.

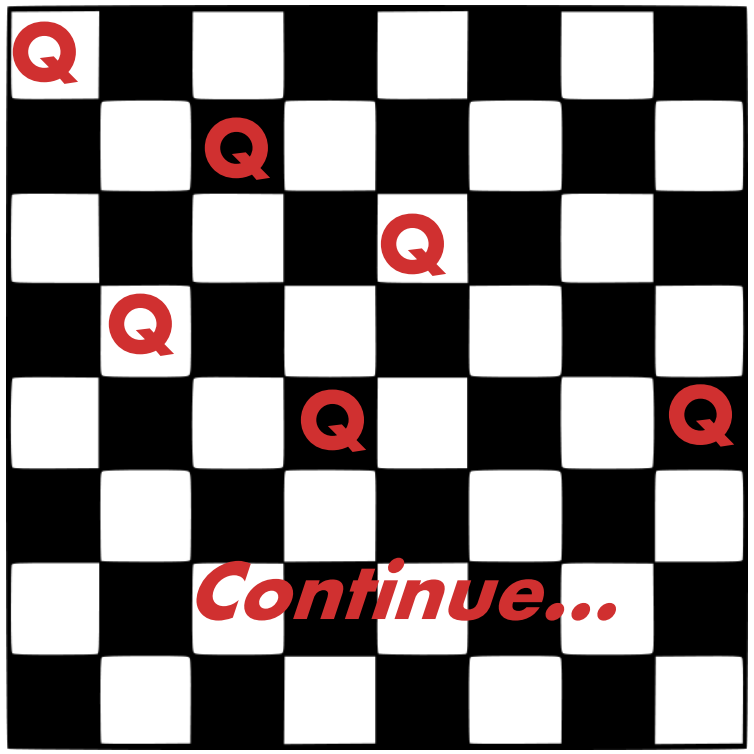
Backtracking – Eight Queens Problem

The backtracking strategy is as follows:

1) Place a queen on the first available square in row 1.

2) Move onto the next row, placing a queen on the first available square there (that doesn't conflict with the previously placed queens).

3) Continue in this fashion until either:
   a) you have solved the problem, or
   b) you get stuck.
      - When you get stuck, remove the queens that got you there, until you get to a row where there is another valid square to try.



Backtracking – Eight Queens Problem

When we carry out backtracking, an easy way to visualize what is going on is a tree that shows all the different possibilities that have been tried.

On the board we will show a visual representation of solving the 4 Queens problem (placing 4 queens on a 4x4 board where no two attack one another).

Backtracking – Eight Queens Problem

The neat thing about coding up backtracking, is that it can be done recursively, without having to do all the bookkeeping at once.

- Instead, the stack or recursive calls does most of the bookkeeping
- (ie, keeping track of which queens we've placed, and which combinations we've tried so far, etc.)

Backtracking – Eight Queens Problem

1) Start in the leftmost column

2) If all queens are placed

   return true

3) Try all rows in the current column.

   Do following for every tried row.

   a) If the queen can be placed safely in this row

      then mark this [row, column] as part of the

      solution and recursively check if placing

      queen here leads to a solution.

Backtracking – Eight Queens Problem

b) If placing the queen in [row, column] leads to
a solution then return true.

c) If placing queen doesn't lead to a solution then
unmark this [row, column] (Backtrack) and go to
step (a) to try other rows.

4) If all rows have been tried and nothing worked,
return false to trigger backtracking.

Backtracking – 8 queens problem - Analysis

Another possible brute-force algorithm is generate the permutations of the numbers 1 through 8 (of which there are 8! = 40,320),

- and uses the elements of each permutation as indices to place a queen on each row.
- Then it rejects those boards with diagonal attacking positions.

The backtracking algorithm, is a slight improvement on the permutation method,

- constructs the search tree by considering one row of the board at a time, eliminating most non-solution board positions at a very early stage in their construction.
- Because it rejects row and diagonal attacks even on incomplete boards, it examines only 15,720 possible queen placements.

Backtracking – 8 queens problem - Analysis

A further improvement which examines only 5,508 possible queen placements is to combine the permutation-based method with the early pruning method:

The permutations are generated depth-first, and the search space is pruned if the partial permutation produces a diagonal attack

Sudoku and Backtracking

Another common puzzle that can be solved by backtracking is a Sudoku puzzle.

The basic idea behind the solution is as follows:

1) Scan the board to look for an empty square that could take on the fewest possible values based on the simple game constraints.
2) If you find a square that can only be one possible value, fill it in with that one value and continue the algorithm.
3) If no such square exists, place one of the possible numbers for that square in the number and repeat the process.
4) If you ever get stuck, erase the last number placed and see if there are other possible choices for that slot and try those next.

Mazes and Backtracking

A final example of something that can be solved using backtracking is a maze.

- From your start point, you will iterate through each possible starting move.
- From there, you recursively move forward.
- If you ever get stuck, the recursion takes you back to where you were, and you try the next possible move.

In dealing with a maze, to make sure you don't try too many possibilities,

- one should mark which locations in the maze have been visited already so that no location in the maze gets visited twice.
- (If a place has already been visited, there is no point in trying to reach the end of the maze from there again.

Backtracking – homework problem

Determine how many solutions there are to the 5 queens problem.

- Demonstrate backtracking for at least 2 solutions to the 5 queens problem, by tracing through the decision tree as shown in class.


The End