



DESIGN AND ANALYSIS OF ALGORITHMS

Backtracking

Reetinder Sidhu

Department of Computer Science and Engineering

DESIGN AND ANALYSIS OF ALGORITHMS

Backtracking

Reetinder Sidhu

Department of Computer Science and
Engineering

- Dynamic Programming
 - ▶ Computing a Binomial Coefficient
 - ▶ The Knapsack Problem
 - ▶ Memory Functions
 - ▶ Warshall's and Floyd's Algorithms
 - ▶ Optimal Binary Search Trees
- Limitations of Algorithmic Power
 - ▶ Lower-Bound Arguments
 - ▶ Decision Trees
 - ▶ P, NP, and NP-Complete, NP-Hard Problems
- Coping with the Limitations
 - ▶ **Backtracking**
 - ▶ Branch-and-Bound

Concepts covered

- Backtracking
 - ▶ Introduction
 - ▶ *N* Queens
 - ▶ Hamiltonian Circuit
 - ▶ Subset Sum
 - ▶ Algorithm

- There are two principal approaches to tackling difficult combinatorial problems (NP-hard problems):
 - ▶ Use a strategy that guarantees solving the problem exactly but doesn't guarantee to find a solution in polynomial time
 - ▶ Use an approximation algorithm that can find an approximate (sub-optimal) solution in polynomial time

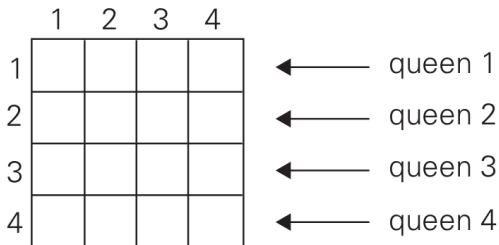
- Exhaustive search (brute force)
 - ▶ useful only for small instances
- Dynamic programming
 - ▶ applicable to some problems (e.g., the knapsack problem)
- Backtracking
 - ▶ eliminates some unnecessary cases from consideration
 - ▶ yields solutions in reasonable time for many instances but worst case is still exponential
- Branch-and-bound
 - ▶ further refines the backtracking idea for optimization problems

- Construct the *state-space tree*
 - ▶ nodes: partial solutions
 - ▶ edges: choices in extending partial solutions
- Explore the state space tree using depth-first search
- “Prune” *nonpromising nodes*
 - ▶ DFS stops exploring subtrees rooted at nodes that cannot lead to a solution and backtracks to such a node’s parent to continue the search

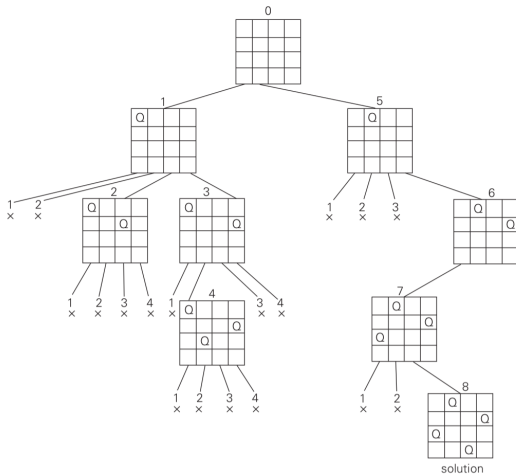
BACKTRACKING

Example: N -Queens Problem

- Place N queens on an $N \times N$ chess board so that no two of them are in the same row, column, or diagonal



State-Space Tree of the 4-Queens Problem

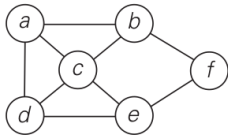


Example: Hamiltonian Circuit Problem

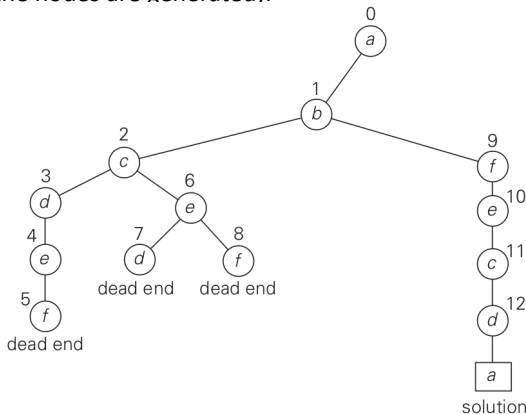
Hamiltonian Circuit

A Hamiltonian circuit is defined as a cycle that passes through all the vertices of the graph exactly once.

- Example graph:



- State-space tree for finding a Hamiltonian circuit (numbers above the nodes indicate the order in which the nodes are generated):

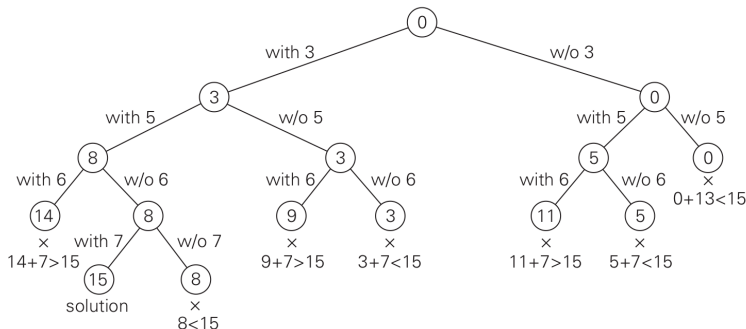


Example: Subset Sum Problem

Subset Sum Problem

Given set $A = \{a_1, \dots, a_n\}$ of n positive integers, find a subset whose sum is equal to a given positive integer d

- State space tree for $A = \{3, 5, 6, 7\}$ and $d = 15$ (number in each node is the sum so far):



BACKTRACKING

Algorithm

Backtrack Algorithm

```
1: procedure BACKTRACK( $X[1 \dots i]$ )
2:   ▷ Input:  $X[1 \dots i]$  specifies first  $i$  promising components of a solution
3:   ▷ Output: All the tuples representing the problem's solutions
4:   if  $X[1 \dots i]$  is a solution then
5:     write  $X[1 \dots i]$ 
6:   else
7:     for each element  $x \in S_{i+1}$  consistent with  $X[1 \dots i]$  and the constraints do
8:        $X[i + 1] \leftarrow x$ 
9:       Backtrack ( $X[1 \dots i + 1]$ )
```

- Output: n -tuples (x_1, x_2, \dots, x_n)
- Each $x_i \in S_i$, some finite linearly ordered set

- Continue the backtracking search for a solution to the four-queens problem, to find the second solution to the problem
- Explain how the board's symmetry can be used to find the second solution to the four-queens problem