Backtracking

General method

- Useful technique for optimizing search under some constraints
- Express the desired solution as an n-tuple $(x_1, \ldots, x_n)$ where each $x_i \in S_i$, S_i being a finite set
- The solution is based on finding one or more vectors that maximize, minimize, or satisfy a *criterion function* $P(x_1, \ldots, x_n)$
- Sorting an array $a[n]$
  - Find an n-tuple where the element x_i is the index of ith smallest element in a
  - Criterion function is given by $a[x_i] \leq a[x_{i+1}]$ for $1 \leq i < n$
  - Set S_i is a finite set of integers in the range [1,n]
- Brute force approach
  - Let the size of set S_i be m_i
  - There are $m = m_1 m_2 \cdots m_n$ n-tuples that satisfy the criterion function P
  - In brute force algorithm, you have to form all the m n-tuples to determine the optimal solutions
- Backtrack approach
  - Requires less than m trials to determine the solution
  - Form a solution (partial vector) and check at every step if this has any chance of success
  - If the solution at any point seems not-promising, ignore it
  - If the partial vector $(x_1, x_2, \ldots, x_i)$ does not yield an optimal solution, ignore $m_{i+1} \cdots m_n$ possible test vectors even without looking at them
- All the solutions require a set of constraints divided into two categories: explicit and implicit constraints

  Definition 1 ***Explicit constraints*** *are rules that restrict each x_i to take on values only from a given set.*

  - Explicit constraints depend on the particular instance I of problem being solved
  - All tuples that satisfy the explicit constraints define a possible *solution space* for I
  - Examples of explicit constraints
    * $x_i \geq 0$, or all nonnegative real numbers
    * $x_i = \{0, 1\}$
    * $l_i \leq x_i \leq u_i$

  Definition 2 ***Implicit constraints*** *are rules that determine which of the tuples in the solution space of I satisfy the criterion function.*

  - Implicit constraints describe the way in which the x_is must relate to each other.
- Determine problem solution by systematically searching the solution space for the given problem instance
  - Use a tree organization for solution space
- 8-queens problem
  - Place eight queens on an 8×8 chessboard so that no queen attacks another queen

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

- Identify data structures to solve the problem
  - First pass: Define the chessboard to be an 8×8 array
  - Second pass: Since each queen is in a different row, define the chessboard solution to be an 8-tuple $(x_1, \ldots, x_8)$, where x_i is the column for ith queen
- Identify explicit constraints
  - Explicit constraints using 8-tuple formulation are $S_i = \{1, 2, 3, 4, 5, 6, 7, 8\}, 1 \leq i \leq 8$
  - Solution space of 8^8 8-tuples
- Identify implicit constraints
  - No two x_i can be the same, or all the queens must be in different columns
    - All solutions are permutations of the 8-tuple $(1, 2, 3, 4, 5, 6, 7, 8)$
    - Reduces the size of solution space from 8^8 to 8! tuples
  - No two queens can be on the same diagonal
- The solution above is expressed as an 8-tuple as $4, 6, 8, 2, 7, 1, 3, 5$

- Sum of subsets
  - Given n positive numbers w_i, $1 \leq i \leq n$, and m, find all subsets of w_i whose sums are m
  - For example, $n = 4$, $w = (11, 13, 24, 7)$, and $m = 31$, the desired subsets are $(11, 13, 7)$ and $(24, 7)$
  - The solution vectors can also be represented by the indices of the numbers as $(1, 2, 4)$ and $(3, 4)$
    - All solutions are k-tuples, $1 \leq k \leq n$
  - Explicit constraints
    - $x_i \in \{j \mid j$ is an integer and $1 \leq j \leq n\}$
  - Implicit constraints
    - No two x_i can be the same
    - $\sum w_{x_i} = m$
    - $x_i < x_{i+1}, 1 \leq i < k$ (total order in indices)
      - Helps in avoiding the generation of multiple instances of same subset; (1, 2, 4) and (1, 4, 2) are the same subset
  - A better formulation of the problem is where the solution subset is represented by an n-tuple $(x_1, \ldots, x_n)$ such that $x_i \in \{0, 1\}$
    - The above solutions are then represented by $(1, 1, 0, 1)$ and $(0, 0, 1, 1)$
  - For both the above formulations, the solution space is 2^n distinct tuples

- n-queen problem
  - A generalization of the 8-queen problem
  - Place n queens on an $n \times n$ chessboard so that no queen attacks another queen
  - Solution space consists of all n! permutations of the n-tuple $(1, 2, \ldots, n)$
  - Permutation tree with 4-queen problem

    * Represents the entire solution space
    * $n!$ permutations for the n-tuple solution space
    * Edges are labeled by possible values of x_i
    * Solution space is defined by all paths from root to leaf nodes
    * For 4-queen problem, there are $4! = 24$ leaf nodes in permutation tree

- Sum of subsets problem
  - Possible tree organizations for the two different formulations of the problem
  - Variable tuple size formulation
    * Edges labeled such that an edge from a level i node to a level $i + 1$ node represents a value for x_i
    * Each node partitions the solution space into subsolution spaces
    * Solution space is defined by the path from root node to any node in the tree
  - Fixed tuple size formulation
    * Edges labeled such that an edge from a level i node to a level $i + 1$ node represents a value for x_i which is either 0 or 1
    * Solution space is defined by all paths from root node to a leaf node
    * Left subtree defines all subsets containing w_1; right subtree defines all subsets not containing w_1
    * 2^n leaf nodes representing all possible tuples
- Terminology

  Problem state is each node in the depth-first search tree

  State space is the set of all paths from root node to other nodes

  Solution states are the problem states s for which the path from the root node to s defines a tuple in the solution space
    - In variable tuple size formulation tree, all nodes are solution states
    - In fixed tuple size formulation tree, only the leaf nodes are solution states
    - Partitioned into disjoint sub-solution spaces at each internal node

  Answer states are those solution states s for which the path from root node to s defines a tuple that is a member of the set of solutions
    - These states satisfy implicit constraints

  State space tree is the tree organization of the solution space

  Static trees are ones for which tree organizations are independent of the problem instance being solved
    - Fixed tuple size formulation
    - Tree organization is independent of the problem instance being solved

  Dynamic trees are ones for which organization is dependent on problem instance

  Live node is a generated node for which all of the children have not been generated yet

  E-node is a live node whose children are currently being generated or explored

  Dead node is a generated node that is not to be expanded any further
    - All the children of a dead node are already generated
    - Live nodes are killed using a **bounding function** to make them dead nodes
- Backtracking is depth-first node generation with bounding functions
- Backtracking on 4-queens problem
  - Bounding function: