Design and Analysis of Algorithms

Intractability & Backtracking

Dr. Soharab Hossain Shaikh
BML Munjal University

1

Tractable and Intractable

- Problems are divided into two broad categories
  - **Tractable problem**: a problem that is solvable by a polynomial time algorithm. The upper bound of running time is polynomial. (e.g. Sorting, Searching, MST, Shortest path)
  - **Intractable problem**: a problem that cannot be solved by a polynomial time algorithm. The lower bound is exponential. (e.g. TSP, Graph coloring, SAT etc.)

2

Examples

- Tractable
  - Searching in a unordered list
  - Sorting a list
  - Multiplication of integers
  - Finding a minimum spanning tree
- Intractable
  - Tower of Hanoi
  - List all permutations of n numbers
  - Satisfiability problem
  - N-queen problem
  - Sum of subsets / Subset Sum Problem

3

Backtracking

> Backtracking is a general problem solving methodology. It can be applied to solve computationally hard problems.

> It follows by making a **DFS search** in a tree (state space tree) representing various intermediate states of the solutions to a problem.

> When the search space for a particular problem grows exponentially, backtracking can be used to search for a suitable solution in reasonable time (if we are interested in a single solution).

> If we need all the solutions to a problem, we may need to search the entire state-space (exponential time required).

> Search begins at the root of the tree and new nodes are generated and checked if it satisfies the **constraints** of a problem; if a constraint is violated, we don't move further, rather backtrack to the immediate parent and look for other possible options. Thus, discarding a major portion of the state-space tree.

4

Backtracking

- Backtracking is a more intelligent variation of exhaustive search approach
- The principal idea of Backtracking is to construct solutions one component at a time and evaluate such partially constructed candidates as follows:
  - If a partially constructed solution can be developed further without violating the problem's constraints, it is done by taking the first remaining legitimate option for the next component.
  - If there is no legitimate option for the next component, no alternatives for any remaining component need to be considered.
  - In this case, the algorithm backtracks to replace the last component of the partially constructed solution with its next option.

5

State-Space Tree

- Backtracking is done by constructing a tree of choices being made, called the *state-space tree*.
- Its **root** represents an initial state before the search for a solution begins.
- The **nodes of the first level** in the tree represent the choices made for the first component of a solution, the nodes of the second level represent the choices for the second component, and so on.
- A node in a state-space tree is said to be ***promising*** if it corresponds to a partially constructed solution that may still lead to a complete solution; otherwise, it is called ***non-promising.***
- **Leaves** represent either nonpromising dead ends or complete solutions found by the algorithm.
- In the majority of cases, a state-space tree for a backtracking algorithm is constructed in the manner of depth first search.

6

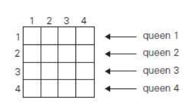
State-Space Tree

- If the current node is promising, its child is generated by adding the first remaining legitimate option for the next component of a solution, and the processing moves to this child.
- If the current node turns out to be nonpromising, the algorithm backtracks to the node's parent to consider the next possible option for its last component; if there is no such option, it backtracks one more level up the tree, and so on.
- Finally, if the algorithm reaches a complete solution to the problem, it either stops (if just one solution is required) or continues searching for other possible solutions.

7

N-Queens Problem

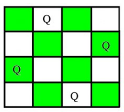
- **The problem is to place n queens on an n × n chessboard so that no two queens attack each other by being in the same row or in the same column or on the same diagonal.**
- **For n = 1, the problem has a trivial solution, and it is easy to see that there is no solution for n = 2 and n = 3.**
- **Example: 4-queens**



Board for the four-queens problem.

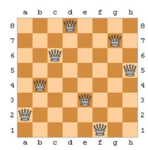
8

Computational Challenge in N-Queens Problem



In a 4×4 board, we have to choose positions for 4 identical queens from 16 different possible squares. This can be done in

$$_{16}C_4 = \frac{16 \cdot 15 \cdot 14 \cdot 12}{4 \cdot 3 \cdot 2 \cdot 1} = 1820 \text{ ways}$$



However, for an 8×8 board, we have to choose positions for 8 identical queens from 64 different squares, which can be done in

$$_{64}C_8 = 178462987637760 \text{ ways!}$$

9

Computational Challenge in N-Queens Problem

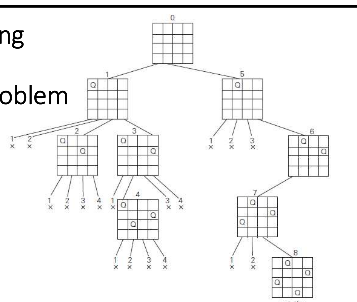
We can tighten this up quite dramatically by enforcing one queen in each row through the use of 8 nested loops as shown below

```
for (int i=0; i < 8; i++) {
  for (int j=0; j < 8; j++) {
    for (int k=0; k < 8; k++) {
      for (int l=0; l < 8; l++) {
              .
              .
            for (int p=0; p < 8; p++) {
                //check validity of (i,j,k,l,…,p);
            }
              .
              .
      }
    }
  }
}
```

Using such nested loops, we would only have to look at $4^4 = 256$ configurations for a 4×4 board and 8^8 configurations for a $8 \times 8 = 16777216$ board -- but this is still far, far too many!

10

Backtracking Solving 4-Queens Problem



State-space tree of solving the four-queens problem by backtracking. × denotes an unsuccessful attempt to place a queen in the indicated column. The numbers above the nodes indicate the order in which the nodes are generated.

Only one queen can be placed in every row.

We need to keep track of the column numbers in every row to identify the position of a queen in that row.

Nodes at level *i* is representing the possible positions (column numbers) where the queen *i* can be placed.

11

Solution to 8-Queens Problem

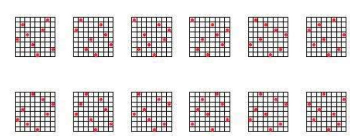
We can solve the 8-Queens problem - generally there are 92 solutions possible, excluding symmetry only 12 unique solutions exist.

Number of queens N =8 and Queens: Q1,Q2....Q8

			Q1				
					Q2		
							Q3
	Q4						
						Q5	
Q6							
		Q7					
				Q8			

Fig.: Solution space table for 8-queens

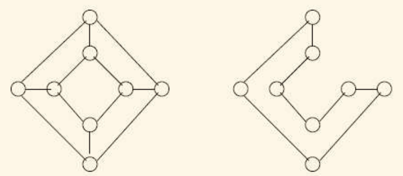
Hence solution vector for 8 queens is (4,6,8,2,7,1,3,5).



12

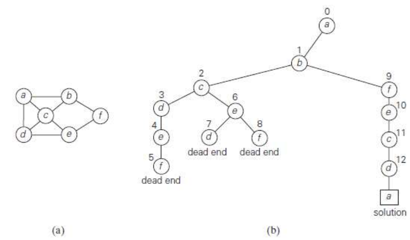
Hamiltonian Cycles

- A *Hamiltonian cycle* is a cycle that contains every vertex exactly once (except that the initial and final vertices will be equal).
- Here we show a graph and, on the right, we highlight a Hamiltonian cycle through that graph.



13

Hamiltonian Cycle

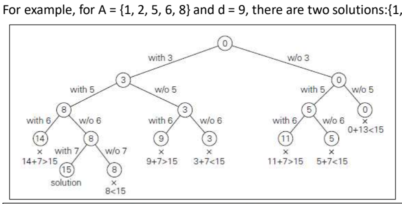


(a) Graph. (b) State-space tree for finding a Hamiltonian circuit. The numbers above the nodes of the tree indicate the order in which the nodes are generated.

14

Subset-Sum Problem

- Find a subset of a given set A = {a1, . . . , an } of n positive integers whose sum is equal to a given positive integer d.
- Example: For example, for A = {1, 2, 5, 6, 8} and d = 9, there are two solutions:{1, 2, 6} and {1, 8}.



Complete state-space tree of the backtracking algorithm applied to the instance $A = \{3, 5, 6, 7\}$ and $d = 15$ of the subset-sum problem. The number inside a node is the sum of the elements already included in the subsets represented by the node. The inequality below a leaf indicates the reason for its termination.

15

End of Lecture

16