How to tackle those difficult problems...

There are two principal approaches to tackling NP-hard problems or other "intractable" 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

Exact solutions

The exact solution approach includes the strategies:

- *exhaustive search* (brute force)
  - useful only for small instances
- *backtracking*
  - eliminates some cases from consideration
- *branch-and-bound*
  - further cuts down on the search
  - fast solutions for most instances
  - worst case is still exponential

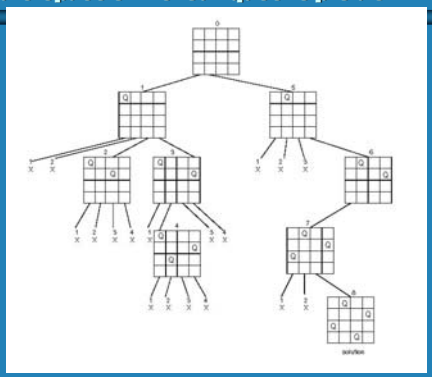
Backtracking

- Construct the *state space tree*:
  - nodes: partial solutions
  - edges: choices in completing solutions
- Explore the state space tree using depth-first search
- "Prune" non-promising nodes
  - dfs stops exploring subtree rooted at nodes leading to no solutions and...
  - "backtracks" to its parent node

Example: The *n*-Queen problem

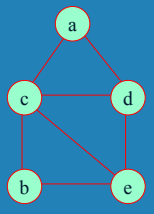
- Place n queens on an n by n chess board so that no two of them are on the same row, column, or diagonal

State-space of the four-queens problem



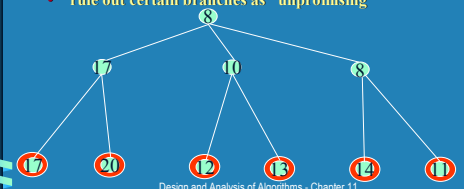
Example: The *m*-coloring problem

- Given a graph and an integer m, color its vertices using no more than m colors so that no two adjacent vertices are the same color.



Branch-and-Bound

- An enhancement of backtracking
- Applicable to optimization problems
- Uses a lower bound for the value of the objective function for each node (partial solution) so as to:
  - guide the search through state-space
  - rule out certain branches as "unpromising"



Example: The assignment problem

Select one element in each row of the cost matrix *C* so that:
- no two selected elements are in the same column; and
- the sum is minimized

For example:

	Job 1	*Job* 2	*Job* 3	*Job* 4
Person *a*	9	2	7	8
Person *b*	6	4	3	7
Person *c*	5	8	1	8
Person *d*	7	6	9	4

Lower bound: Any solution to this problem will have total cost of at least:

Assignment problem: lower bounds

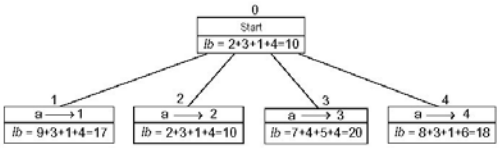


Figure 11.5 Levels 0 and 1 of the state-space tree for the instance of the assignment problem being solved with the best-first branch-and-bound algorithm. The number above a node shows the order in which the node was generated. A node's fields indicate the job number assigned to person *a* and the lower bound value, *lb*, for this node.

State-space levels 0, 1, 2

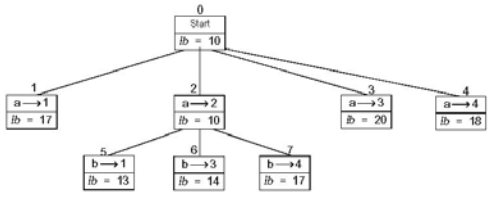


Figure 11.6 Levels 0, 1, and 2 of the state-space tree for the instance of the assignment problem being solved with the best-first branch-and-bound algorithm

Complete state-space

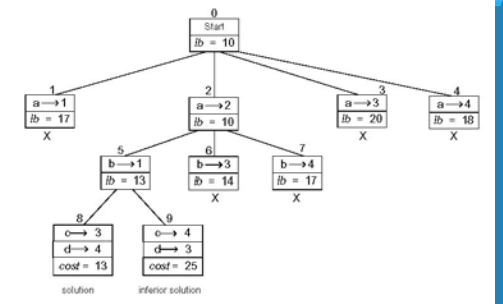


Figure 11.7 Complete state-space tree for the instance of the assignment problem solved with the best-first branch-and-bound algorithm

Traveling salesman example: