中山大學
SUN YAT-SEN UNIVERSITY

Lecture 8
Backtracking

Algorithm Design

zhangzizhen@gmail.com

QQ group: 117282780

Coping with NP-C and NP-H problems

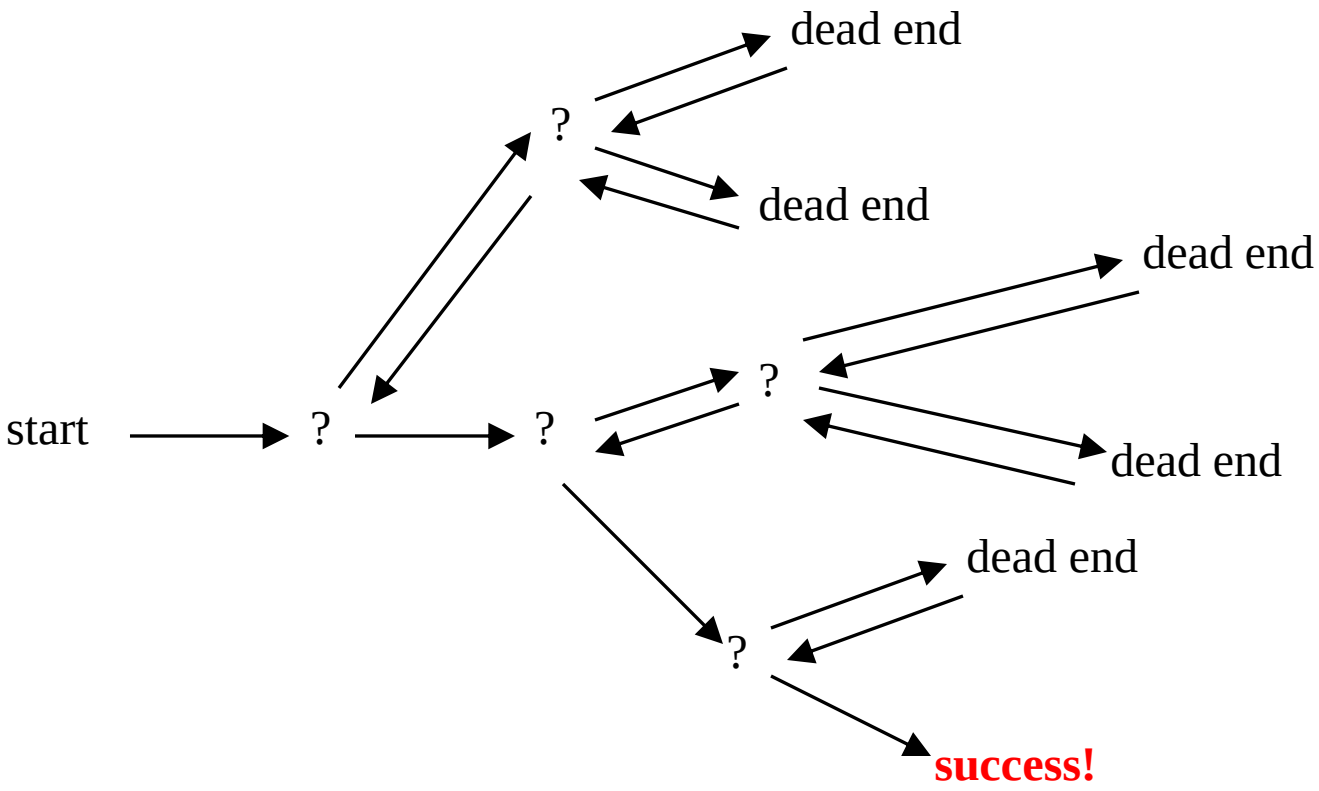
- Dynamic programming
- Backtracking
- Branch-and-bound
- Approximation algorithms
- Heuristics
- Meta-heuristics

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"

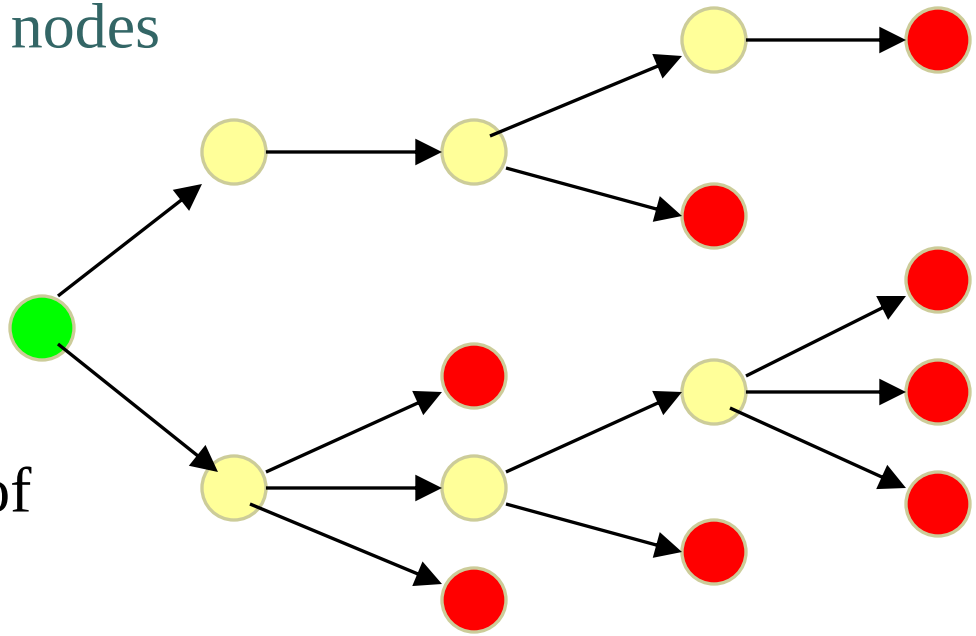
Backtracking

- Example: Decision making process.



Search Tree

A tree is composed of nodes



There are three kinds of nodes:

- The (one) root node
- Internal nodes
- Leaf nodes

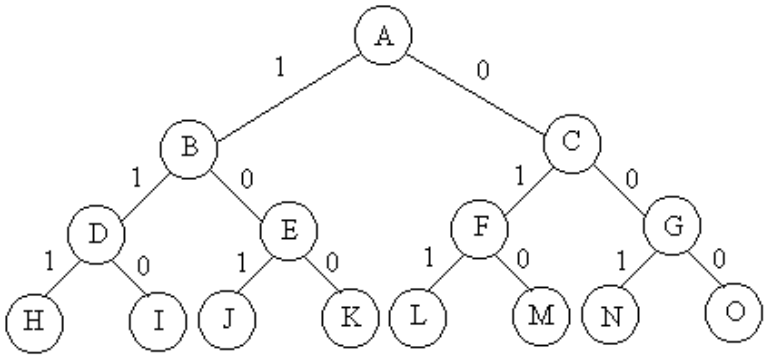
Backtracking can be thought of as searching a tree for a particular "goal" leaf node

The backtracking algorithm

- Backtracking is really quite simple -- we **recursively** "explore" each node, as follows:
- To "explore" node N:

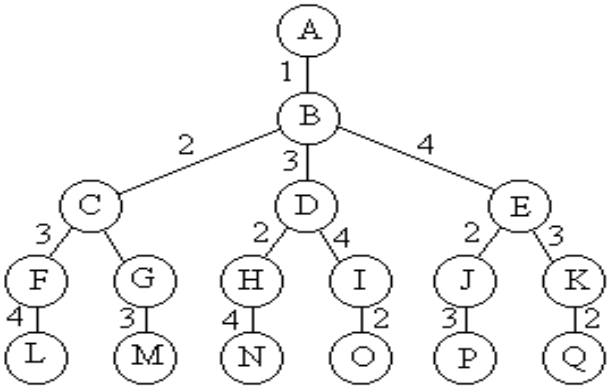
  1. If N is a goal node, return "**success**"
  2. If N is a leaf node, return "**failure**"
  3. For each child C of N,
  3.1. Explore C
  3.1.1. If C was successful, return "success"
  4. Return "failure"

Subset Tree and Permutation Tree



Enumerating all subsets take $O(2^n)$

```
void backtrack(int t)
{
  if (t>n) output(x);
  else
      for (int i=0;i<=1;i++) {
        x[t]=i;
        if (legal(t)) backtrack(t+1);
      }
}
```

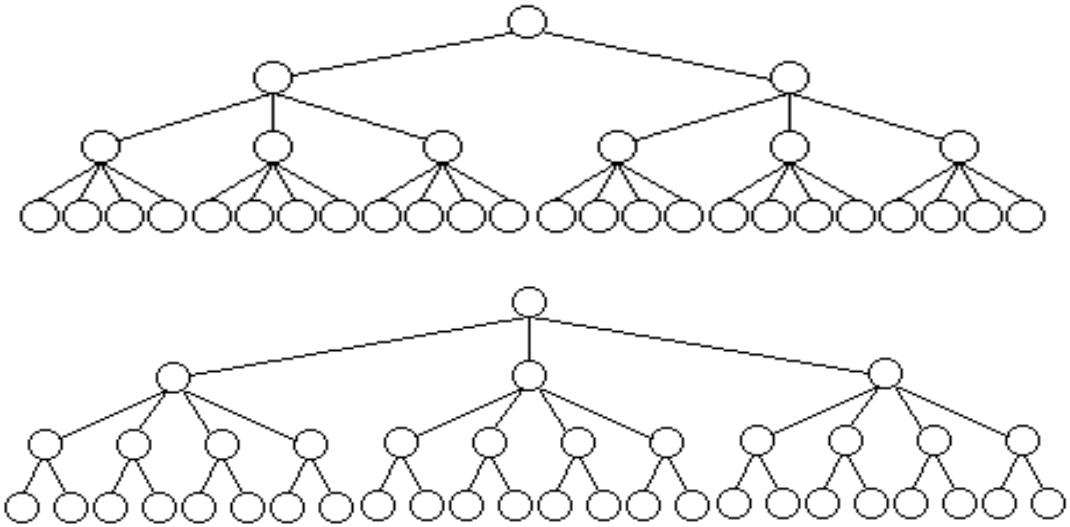


Enumerating all permutations take O(n!)

```
void backtrack(int t)
{
  if (t>n) output(x);
    else
      for (int i=t;i<=n;i++) {
        swap(x[t], x[i]);
        if (legal(t)) backtrack(t+1);
        swap(x[t], x[i]);
      }
}
```

重排原理

对于许多问题而言，在搜索试探时选取x[i]的值顺序是任意的。**在其它条件相当的前提下，让可取值最少的x[i]优先**。从图中关于同一问题的2棵不同解空间树，可以体会到这种策略的潜力。



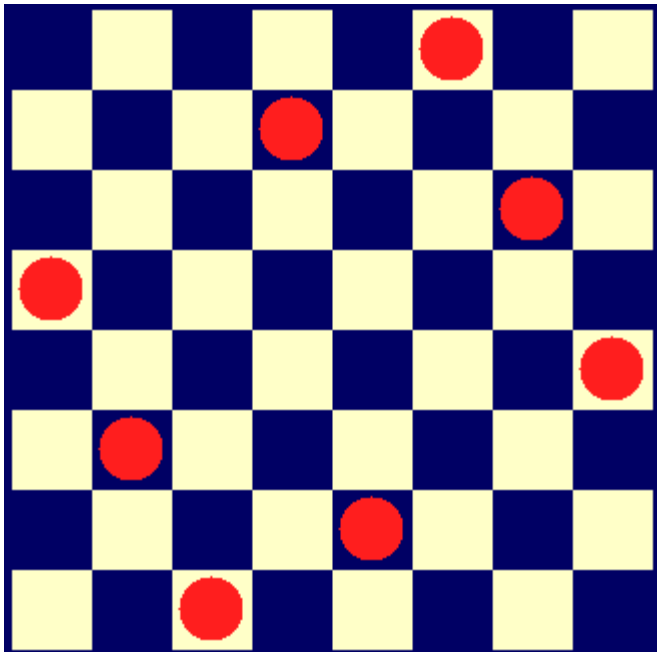
(a)

(b)

图(a)中，从第1层剪去1棵子树，则从所有应当考虑的3元组中一次消去12个3元组。对于图(b)，虽然同样从第1层剪去1棵子树，却只从应当考虑的3元组中消去8个3元组。前者的效果明显比后者好。

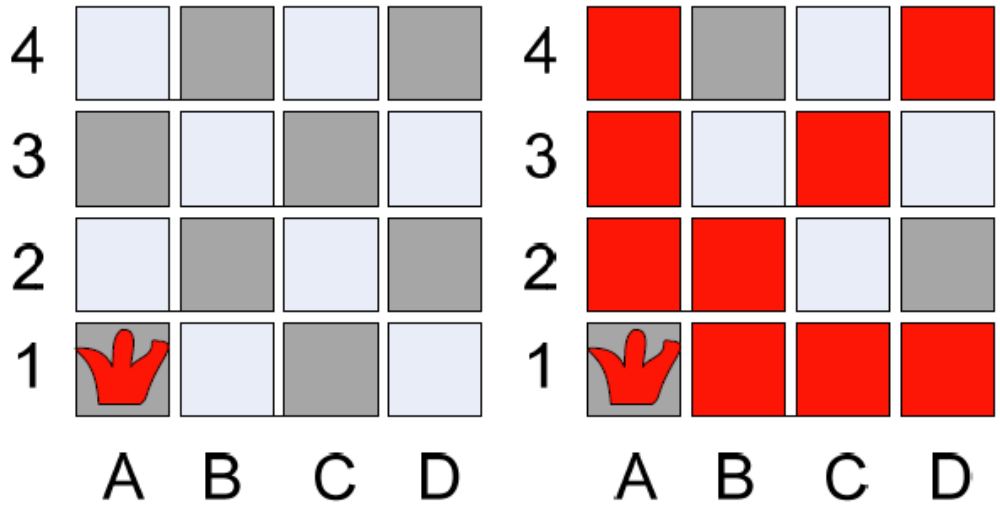
N Queen Problem

- In chess, a queen can move as far as she pleases, horizontally, vertically, or diagonally. A chess board has 8 rows and 8 columns. The standard 8 by 8 Queen's problem asks how to place 8 queens on an ordinary chess board so that none of them can hit any other in one move.

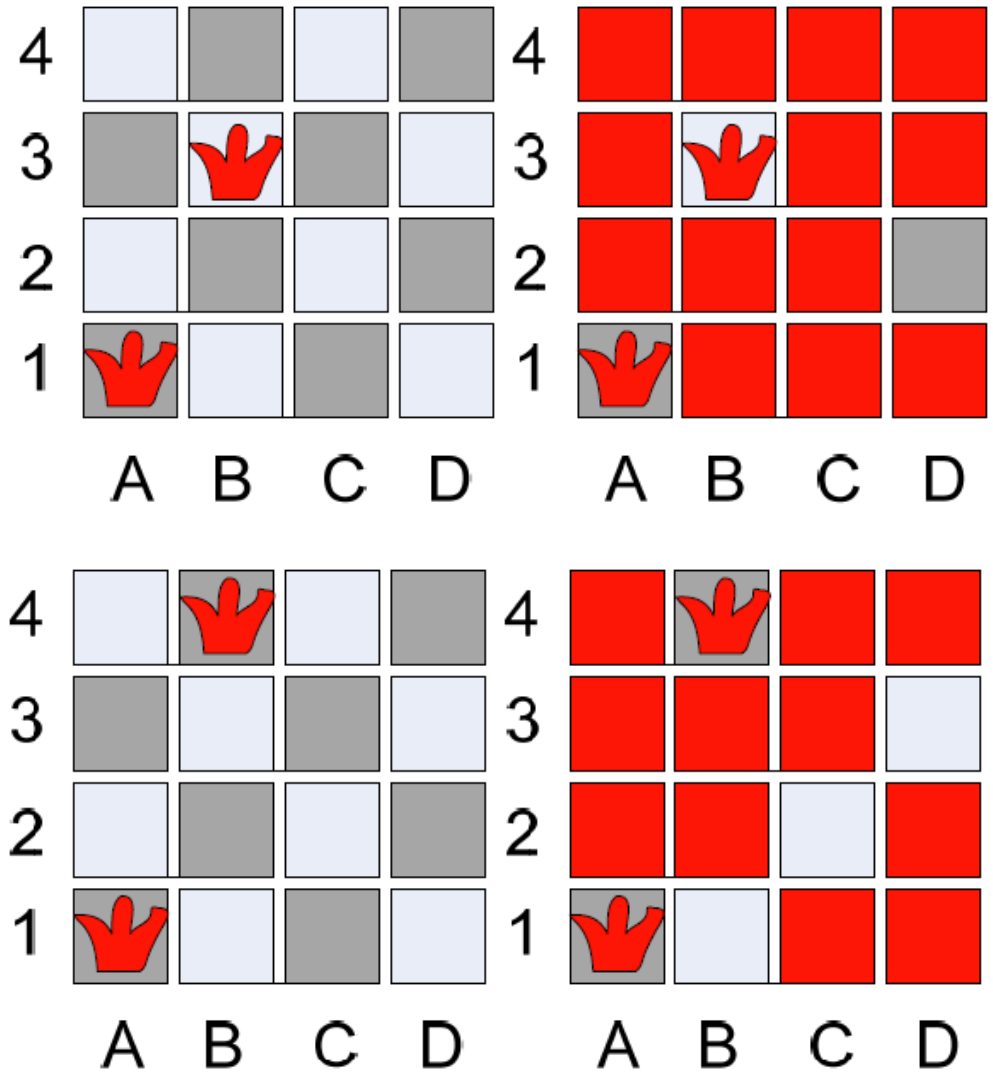


N Queen Problem

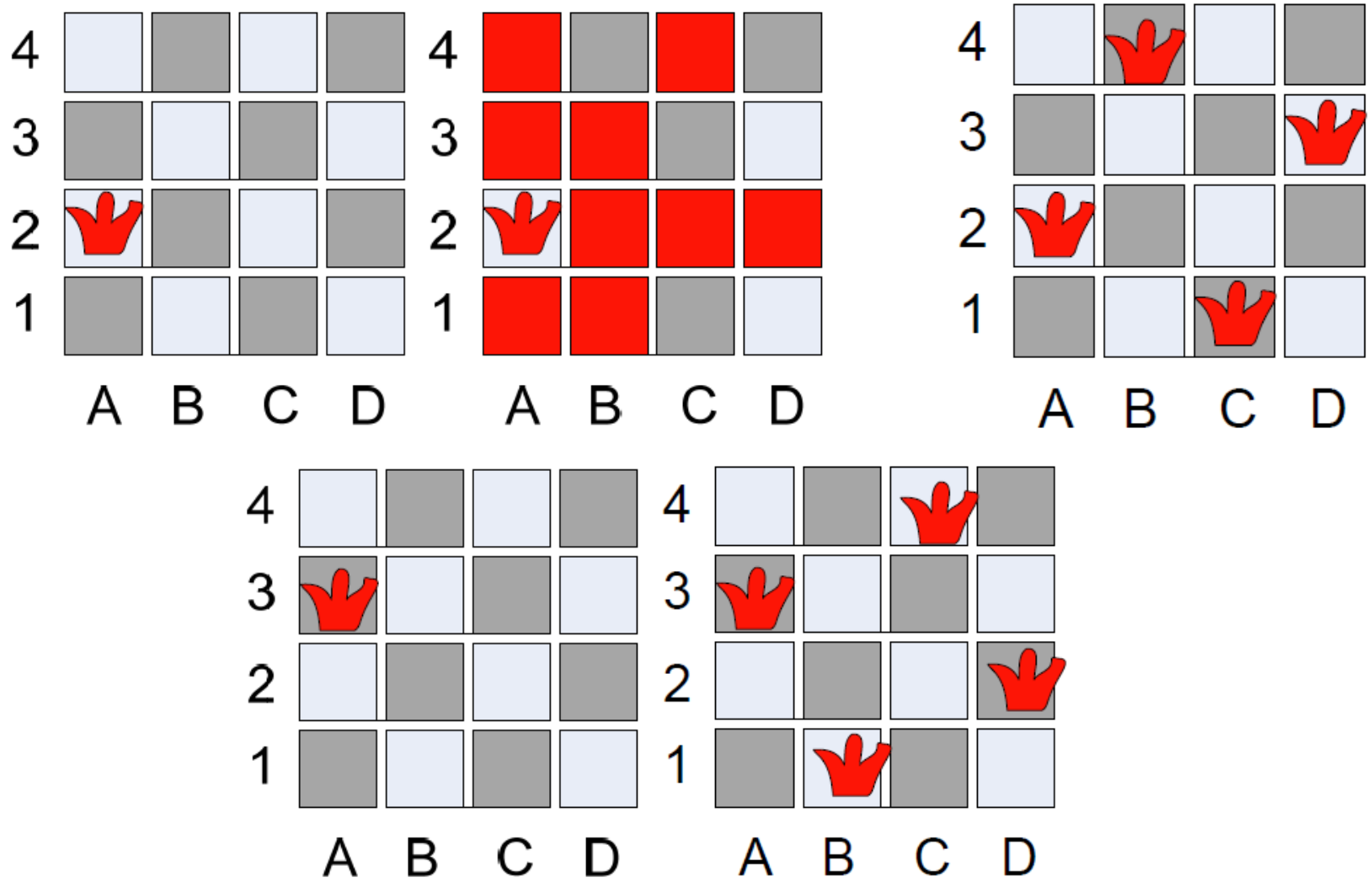
- Algorithm:
  - Start with one queen at the first column first row
  - Continue with second queen from the second column first row
  - Go up until find a permissible situation
  - Continue with next queen



N Queen Problem



N Queen Problem



N Queen Problem

- Different column: $x_i \neq x_j$
- Different diagonal: $|i-j| \neq |x_i - x_j|$

```
bool Queen::Place(int k)
{
  for (int j=1;j<k;j++)
    if ((abs(k-j)==abs(x[j]-x[k]))||(x[j]==x[k])) return false;
  return true;
}
void Queen::Backtrack(int t)
{
  if (t>n) sum++;
    else
      for (int i=1;i<=n;i++) {
        x[t]=i;
        if (Place(t)) Backtrack(t+1);
      }
 }
```

Thank you!