中山大學
SUN YAT-SEN UNIVERSITY

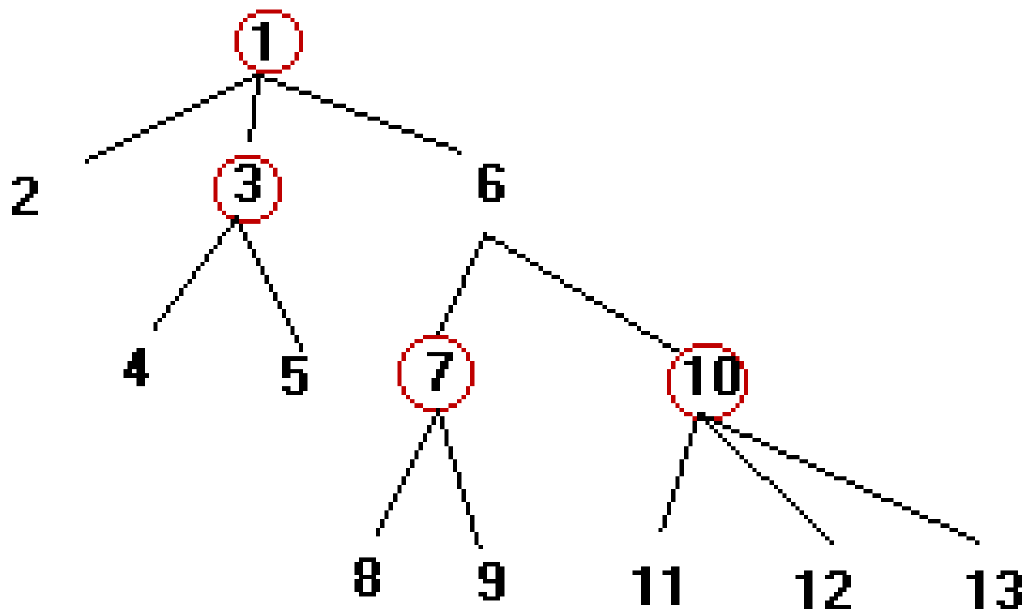
Lecture 8
Dynamic Programming
Part III

Algorithm Design and Analysis

zhangzizhen@gmail.com

Minimum vertex cover of a tree

- 问题描述：给出一个n个结点的树，要求选出其中的一些顶点，使得对于树中的每条边(u, v)，u和v至少有一个被选中. 请给出选中顶点数最少的方案.



Minimum vertex cover of a tree

- 在树结构中，每个结点都是某棵子树的根
- 把n个结点分别用0~n-1编号
- 用ans[i][0]表示：在不选择结点i的情况下，以i为根的子树，最少需要选择的点数；
- 用ans[i][1]表示：在选择结点i的情况下，以i为根的子树，最少需要选择的点数.
- 当i是叶子时，ans[i][0] = 0, ans[i][1] = 1；
- 否则，
  - ans[i][0] = Σans[j][1] (对于i的所有子结点j)
  - ans[i][1] = 1+Σmin(ans[j][0], ans[j][1]) (对于i的所有子结点j)

Traveling Salesman Problem

- Given n cities and the distances d_{ij} between any two of them, we wish to find the shortest tour going through all cities and back to the starting city. Generally, the TSP is given as a graph $G=(V,D)$ where $V=\{1,2, . . ., n\}$ is the set of cities, and D is the adjacency distance matrix, with $\forall i,j \in V$, $i \neq j$, $d_{ij}>0$, the problem is to find the tour with minimal distance weight, that starting at city 1 goes through all n cities and returns to city 1.

Traveling Salesman Problem

- The TSP is a well known NP-hard problem.
- There are n! feasible solutions. Enumerate them may take O(n!) time.
- What is the appropriate subproblem for the TSP?
  - Suppose we have started at city 1 as required, have visited a few cities, and are now in city *j*. What information do we need in order to extend this partial tour?
  - We need to know *j*, since this will determine which cities are most convenient to visit next.
  - We also need to know all the cities visited so far, so that we don’t repeat any of them.

Traveling Salesman Problem

- For a subset of cities $S \subseteq \{1,2,\ldots,n\}$ that includes *1*, and $j \in S$, let $C(S,j)$ be the length of the shortest path visiting each node in *S* exactly once, starting at *1* and ending at *j*.
- How to express $C(S,j)$ in terms of smaller sub-problems.
- We need to start at 1 and end at j; what should we pick as the second-to-last city? It has to be some $i \in S$, so the overall path length is the distance from 1 to *i*, namely, $C(S-\{j\}, i)$, plus the length of the final edge, d_{ij}.
- We pick the best *i*, then

$$C(S,j) = \min_{i \in S: i \neq j} C(S-\{j\}, i) + d_{ij}$$

Traveling Salesman Problem

- There are at most $2^n * n$ subproblems.
- Each one takes linear time to solve.
- The total time complexity is $O(2^n * n^2)$.
- The sub-problems are ordered by |S|. (Use a queue to extend *S*))

C({1},1) = 0
for s = 2 to n:
  for all subsets S ⊆ {1,2,...,n} of size s and containing 1:
    C(S,1) = ∞
    for all $j \in S, j \neq 1$:
      C(S, j) = $min\{C(S-\{j\}, i) + d_{ij} : i \in S, i \neq j\}$
return $min_j C(\{1, ..., n\}, j) + d_{j1}$

Traveling Salesman Problem

- How to represent the set *S*?
- Usually, we can Represent a set of *n* elements as an *n* bit numbers.
- Example: (n=5)

$S=\Phi$ → $(00000)_2$ → 0

$S=\{0\}$ → $(00001)_2$ → 1

$S=\{1,3\}$ → $(01010)_2$ → 10

$S=\{0,1,2,3,4\}$→ $(11111)_2$ → 15

Traveling Salesman Problem

- Check if element *i* is present in set *S*
- Find the resulting set when we add *i* to set *S*
- Iterating through all the subsets of size <= n

Backtracking

Algorithm Design and Analysis

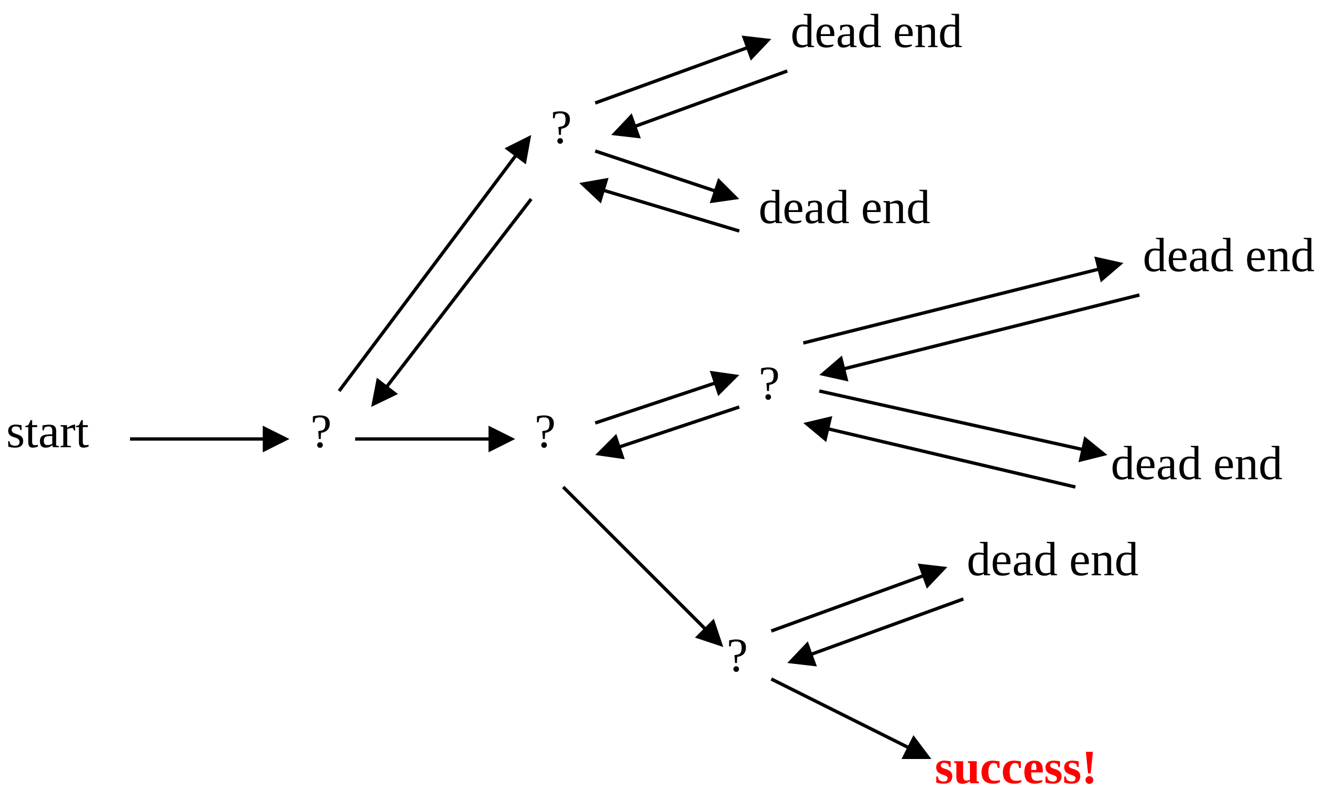
zhangzizhen@gmail.com

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"

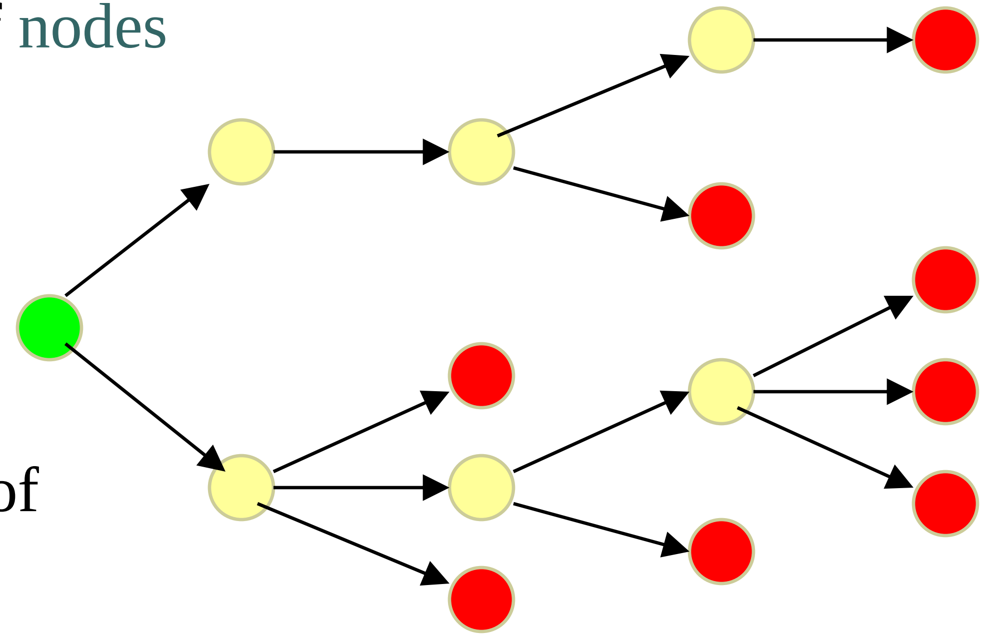
Backtracking

- 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**"

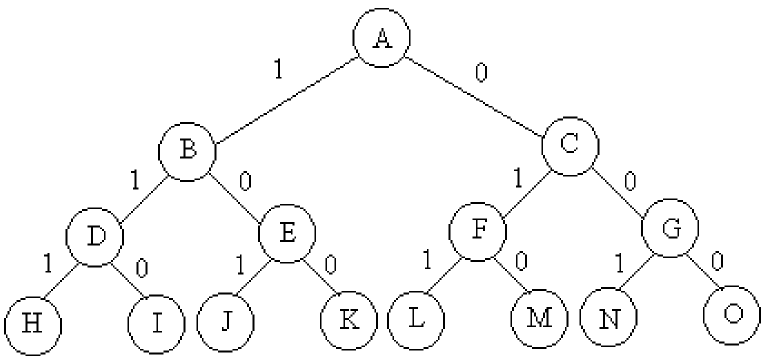
3. For each child C of N,

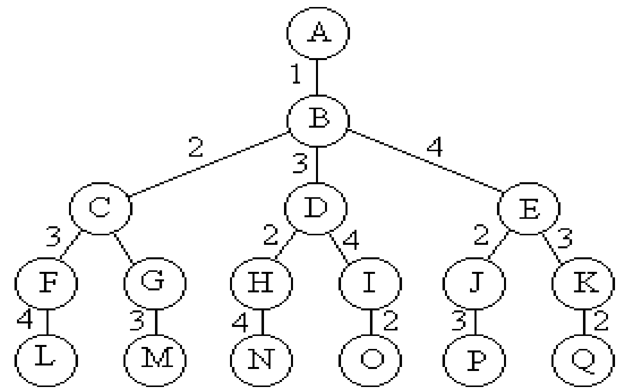
3.1. Explore C

3.1.1. If C was successful, return "success"

4. Return "failure"

Subset Tree and Permutation Tree





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

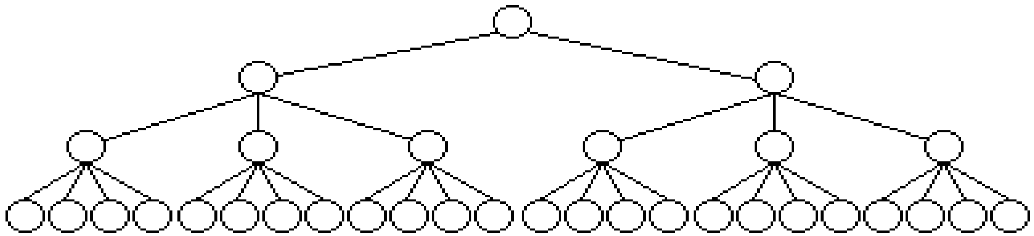
```
void backtrack(int t)
{
  if (t>n) output(x);
  else
      for (int i=0;i<=1;i++) {
        x[t]=i;
        if (legal(t)) backtrack(t+1);
      }
}
```

Enumerating all permutations take $O(n!)$

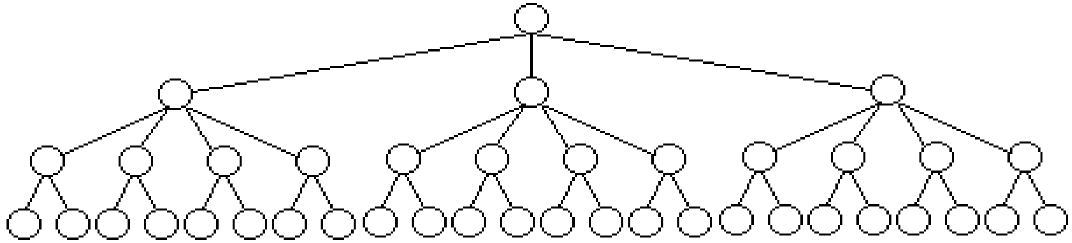
```
void backtrack(int t)
{
  if (t>n) output(x);
    else
      for (int i=t;i<=n;i++) {
        swap(x[t], x[i]);
        if (legal(t)) backtrack(t+1);
        swap(x[t], x[i]);
      }
}
```

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)

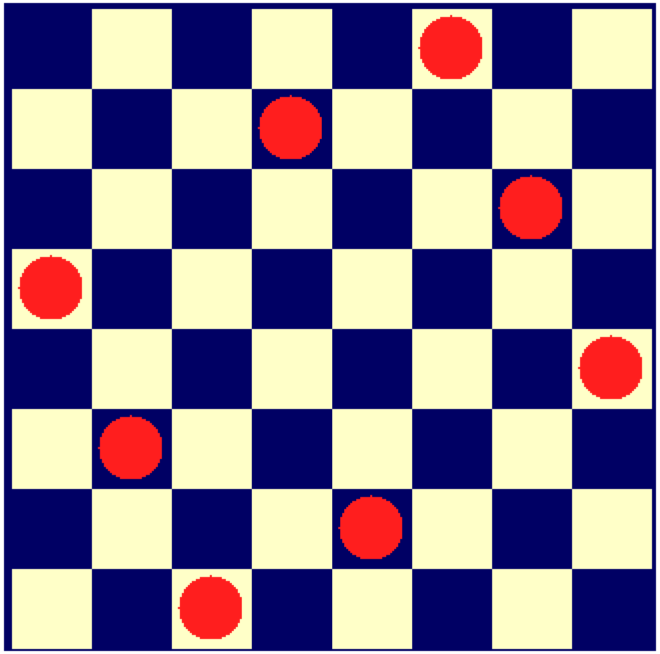


(b)

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。

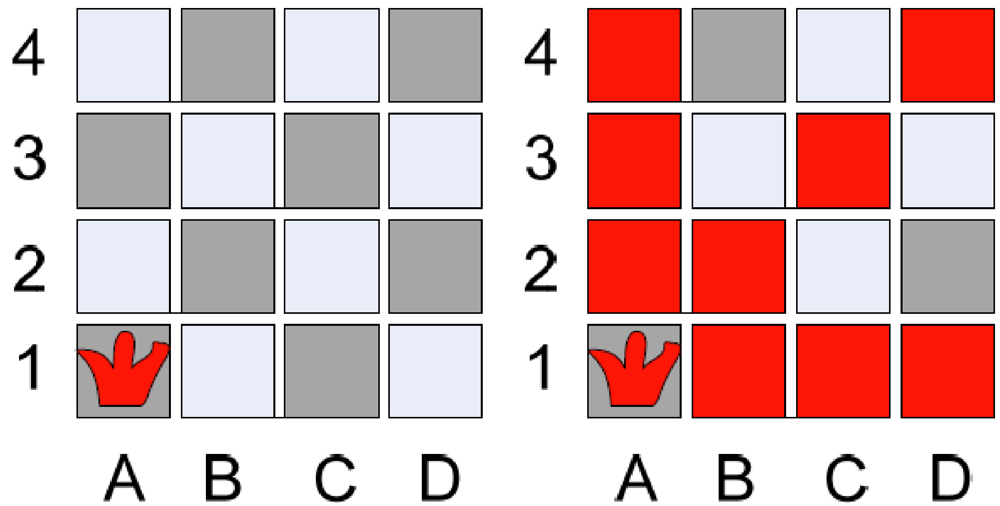
N Queen Problem

- 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.

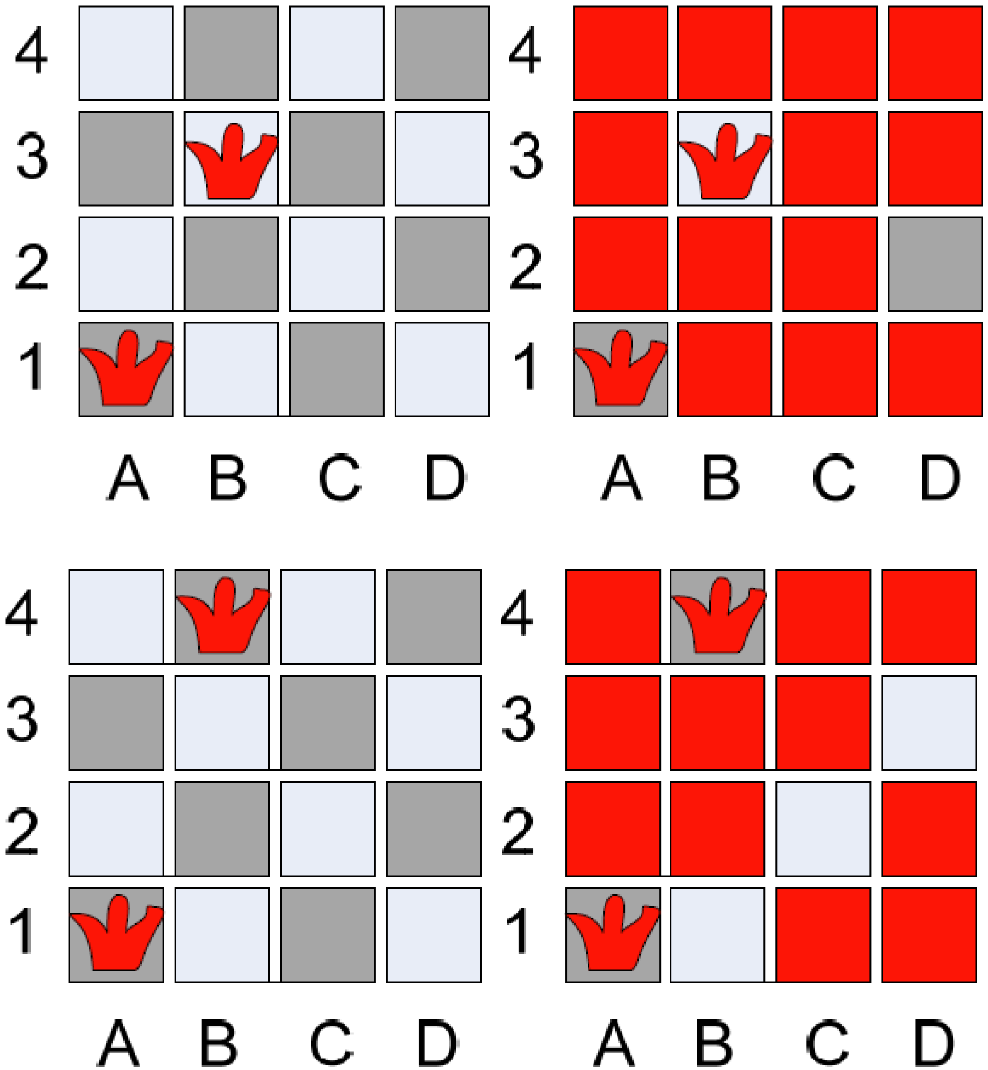


N Queen Problem

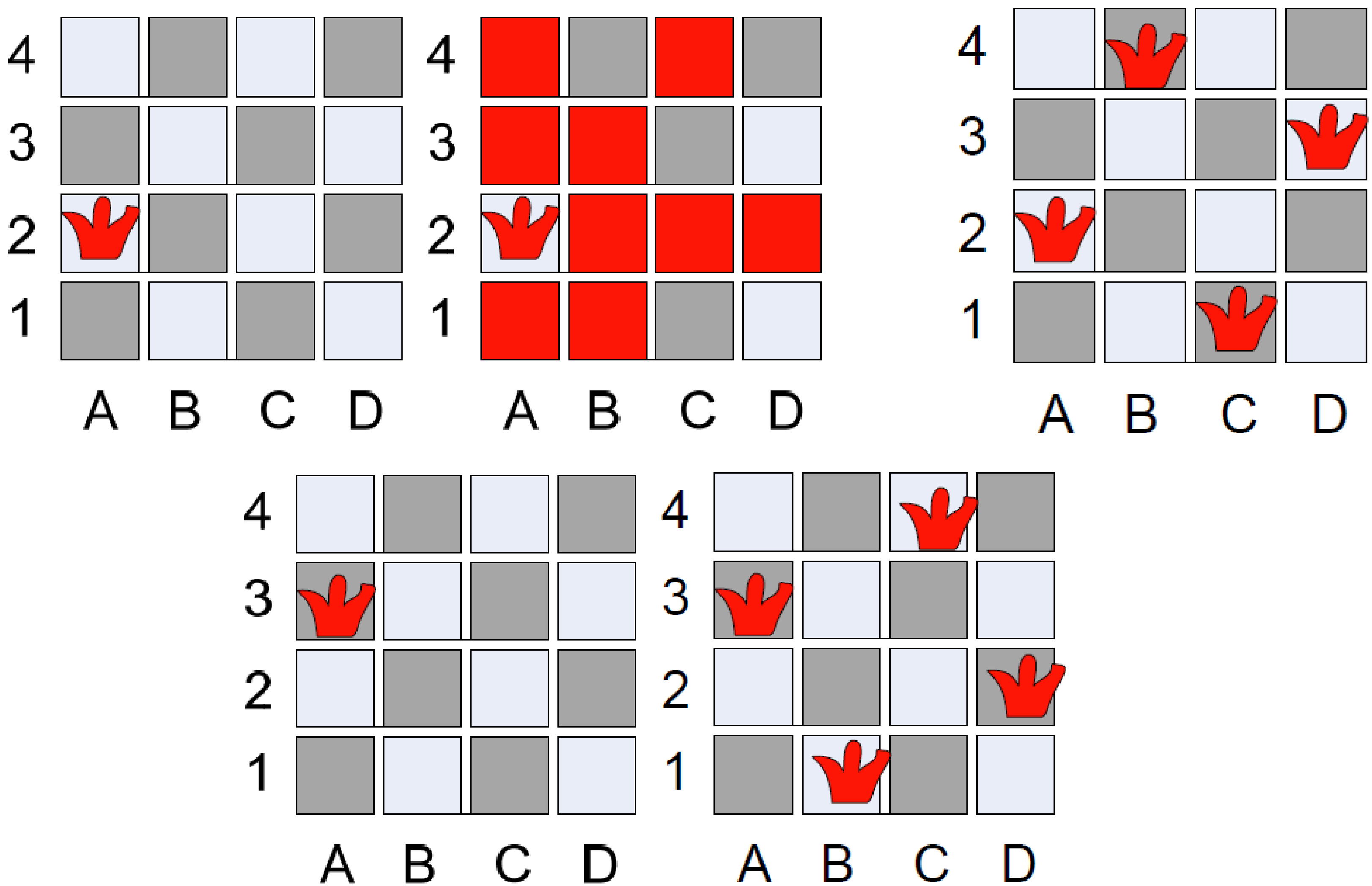
- Algorithm:
  - Start with one queen at the first column first row
  - Continue with second queen from the second column first row
  - Go up until find a permissible situation
  - Continue with next queen



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);
      }
 }
```

Exercise

- soj.acmm.club

  1152 1153 1093 1134 1140 1438

  1028 1029 1381 1206 1012 1034

- Choose at least 2 problems and write a report.
- Create a zip/rar file: ID_name_version.zip
- Submit it to 239o58336k.qicp.vip:55469
  - Account: login
  - Password: 123456

- Deadline: April 14.

Thank you!