约束满足性问题（CSP）

Wanjun Zhong Nov.20

内容

- 理论课内容回顾
  - 约束满足问题
  - Backtracking algorithm
  - Forward checking algorithm
- 实验课任务与报告提交

内容

- 理论课内容回顾
  - 约束满足问题
  - Backtracking algorithm
  - Forward checking algorithm
- 实验课任务与报告提交

约束满足问题（CSP）

对于一组变量，每个变量都有取值域。当每个变量都有自己的赋值，并且同时这些赋值满足所有关于变量的约束时，问题就被解决了。这类问题就叫做约束满足问题（Constraint satisfaction problem**）。**

- 问题组成

X：变量的集合（ variables）

D：值域的集合，每个变量有自己的值域（domain）

C：描述变量取值的约束的集合（constraint）

- 问题求解

定义域有限的约束满足问题通常利用搜索方法来解决，比如backtracking，以及forward-checking算法。

Backtracking （回溯）算法

回溯法属于暴力搜索算法，尤其适用于约束满足问题。它采用了试错的思想，分步解决问题。

- 算法描述：算法在解决问题时，逐步构造更多的候选解，当确定某一部分候选解不可能补全成正确解之后放弃搜索这个部分候选解本身以及其可以扩展出的子候选解，转而测试其它的部分候选解。
- 算法步骤：

1. 选择一个可赋值变量x
2. 在x的值域中选择一个值，为x赋值
3. 测试x的值是否满足约束条件，如果不满足，则回溯
4. 否则继续为其它可赋值节点赋值
5. 如果所有的变量都被赋值了，或者无法找到解，则算法结束

Backtracking （回溯）算法

```
BT(Level)
  If all variables assigned
       PRINT Value of each Variable
       RETURN or EXIT (RETURN for more solutions)
                      (EXIT for only one solution)
  V := PickUnassignedVariable()
  Assigned[V] := TRUE
  for d := each member of Domain(V) (the domain values of V)
       Value[V] := d
       ConstraintsOK = TRUE
       for each constraint C such that
            a) V is a variable of C and
            b) all other variables of C are assigned:
               ;(rarely the case initially high in the search tree)
          IF C is not satisfied by the set of current
            assignments:
            ConstraintsOK = FALSE
       If ConstraintsOk == TRUE:
            BT(Level+1)

  Assigned[V] := FALSE //UNDO as we have tried all of V's values
  return
```

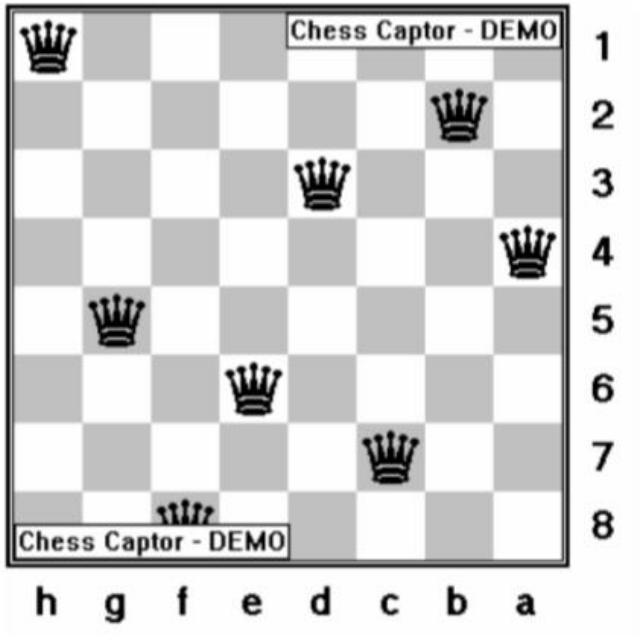
为V赋值并检查是否满足约束

约束不满足，回溯

约束满足，为其它变量赋值

例子：N皇后问题

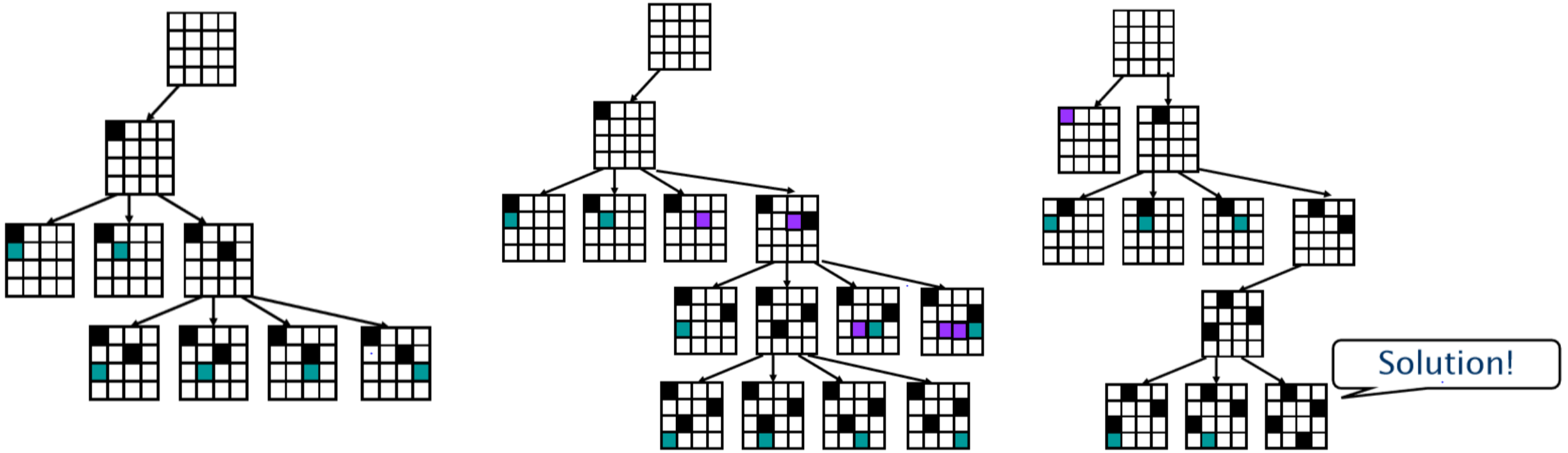
Place N Queens on an N X N chess board so that no Queen can attack any other Queen.



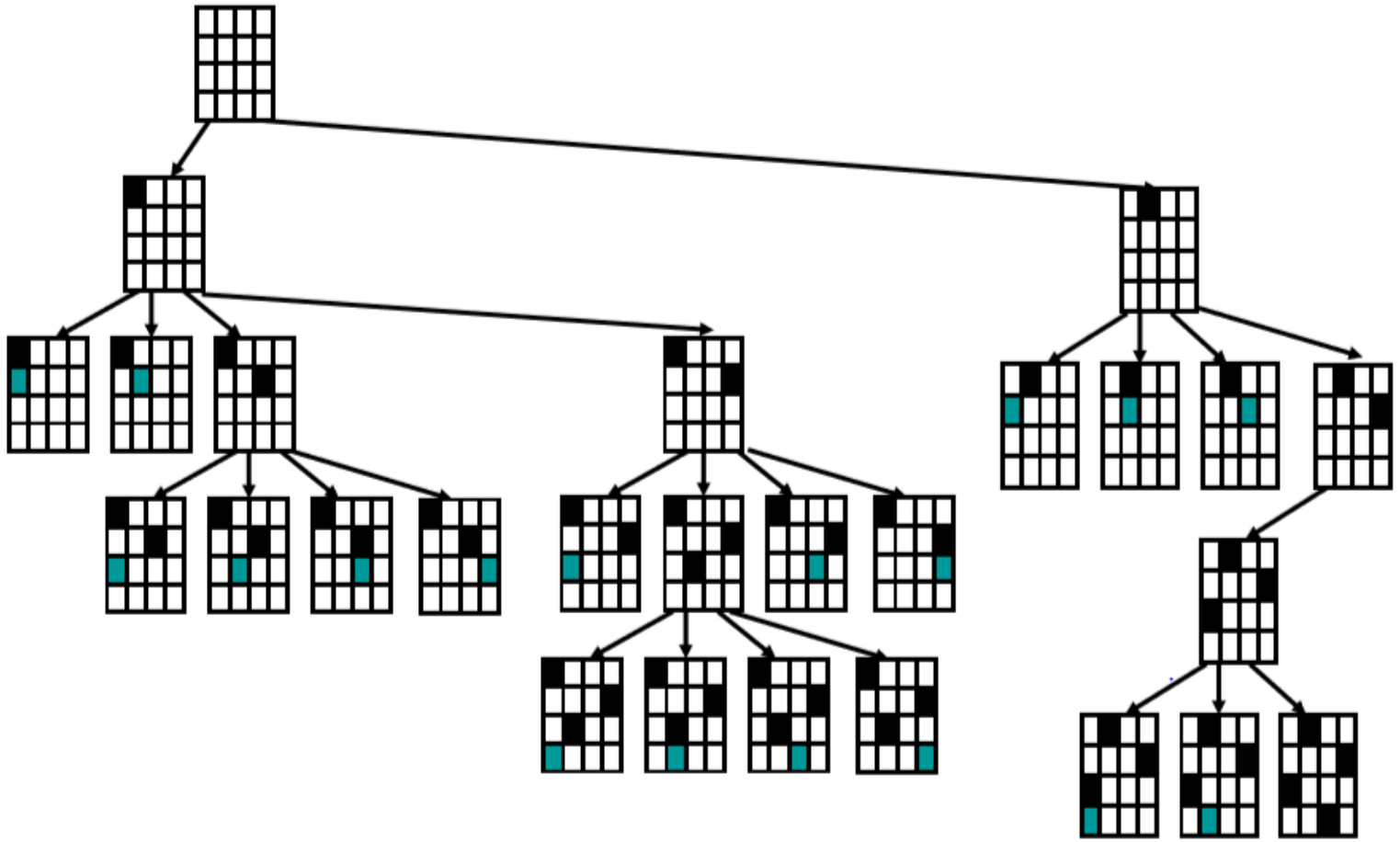
限制：

任意两个皇后都不能处于同一条横线，纵行，或者斜线上。

例子：4皇后问题（回溯法）



例子：4皇后问题（回溯法）



Backtracking （回溯）算法

算法缺陷：它要将一个解的各个部分全部生成后，才检查满足条件。因此时间复杂度较高。

解决办法：加入约束传播(constraint propagation)**步骤。约束传播会在算法的每一步，都"向前看"，检查那些还没有被赋值的变量的可取值状态。这种方法可以提前预测某条会失败的路径，或者改变可赋值变量的顺序。**

Forward-checking 算法

FC算法是backtracking算法的扩展。在搜索过程中，它尝试去预测为当前变量赋某一个值所造成的影响（约束传播）。

- 算法描述：在算法的每一步，算法都检查所有没有被赋值的变量，从它们的取值域中去除那些违背了约束的取值。如果某个未赋值节点的可取值为空，证明当前节点的赋值错误，算法可以提前回溯。在算法的某一步，也可以优先为那些可取值数少的节点赋值。

Forward-checking 算法

For a single constraint C:

```
FCCheck(C,x)
  // C is a constraint with all its variables already
  // assigned, except for variable x.
  for d := each member of CurDom[x]
     IF making x = d together with previous assignments
        to variables in scope C falsifies C
     THEN remove d from CurDom[x]
IF CurDom[x] = {} then return DWO (Domain Wipe Out)
ELSE return ok
```

对于某个限制C以及未赋值变量x，检查x的值域D中的每个值d是否满足约束，如果不满足，则从值域中删除取值d。若值域为空，则返回DWO。

Forward-checking 算法

```
FC(Level) /*Forward Checking Algorithm */
   If all variables are assigned
        PRINT Value of each Variable
        RETURN or EXIT (RETURN for more solutions)
                       (EXIT for only one solution)
   V := PickAnUnassignedVariable()
   Assigned[V] := TRUE
   for d := each member of CurDom(V)
        Value[V] := d
        DWOoccured:= False
        for each constraint C over V such that
              a) C has only one unassigned variable X in its scope
           if(FCCheck(C,X) == DWO)     /* X domain becomes empty*/
              DWOoccurred:= True
              break /* stop checking constraints */
        if(not DWOoccured) /*all constraints were ok*/
           FC(Level+1)
        RestoreAllValuesPrunedByFCCheck()
   Assigned[V] := FALSE //undo since we have tried all of V's values
   return;
```

为V赋值d之后，检查所有可检查的约束。

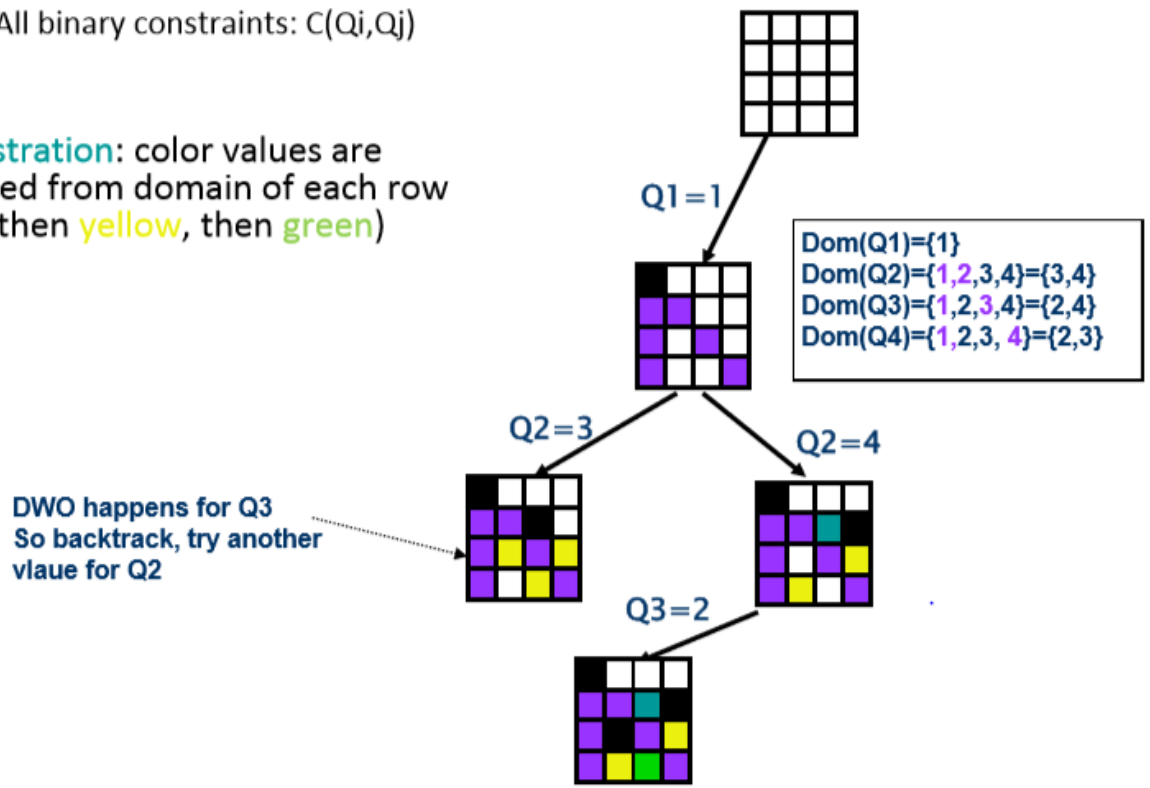
若DWO出现，则放弃取值d。

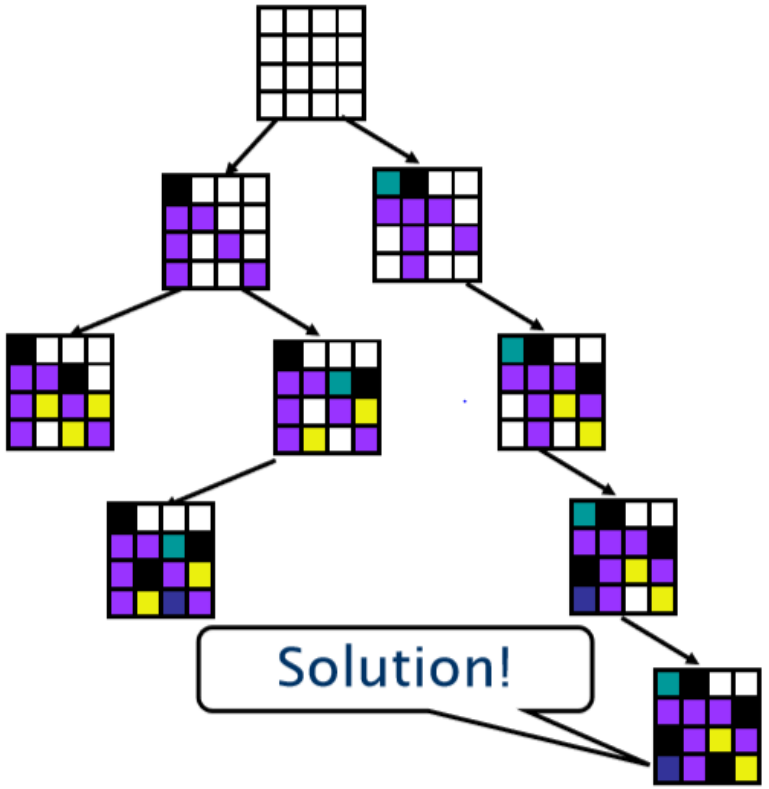
否则，为其它变量赋值。

Forward-checking 算法 – 4皇后问题

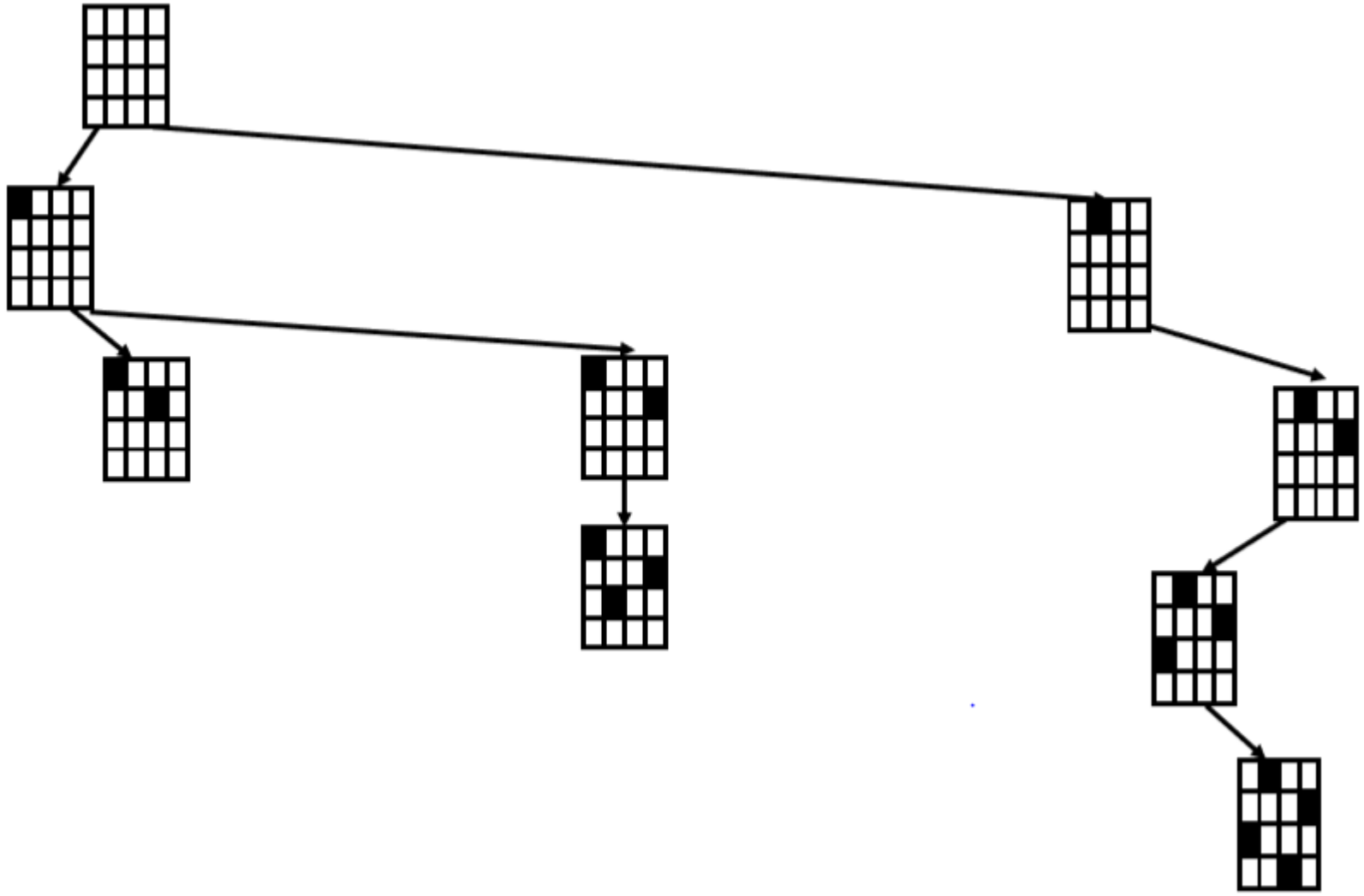
- 4X4 Queens
  - Q1,Q2,Q3,Q4 with domain {1..4}
  - All binary constraints: C(Qi,Qj)

- FC illustration: color values are removed from domain of each row (blue, then yellow, then green)





Forward-checking 算法 – 4皇后问题



内容

- 理论课内容回顾
  - 约束满足问题
  - Backtracking algorithm
  - Forward checking algorithm
- 实验课任务与报告提交

实验任务

- 使用backtracking以及forward-checking 求解N皇后问题 （输入N，求解可行的解，报告中至少展示N=8 以及 N=10)
- 代码要求使用python 或者C++
- 报告要求

① 报告中需要包含对两个算法的原理解释

② 需要包含两种方法在性能上的对比和分析

- 提交文件

① 实验报告：16******_wangxiaoming.pdf

② 代码：16******_wangxiaoming.zip 如果代码分成多个文件，需要写readme

- DDL：2018-12-04 23:00:00

Thanks!

Wanjun Zhong Nov.20