FCT/Unesp – Presidente Prudente
Departamento de Matemática e Computação

Projeto e Análise de Algoritmos

Força Bruta – Backtracking

Prof. Danilo Medeiros Eler
danilo.eler@unesp.br

Departamento de Matemática e Computação
Faculdade de Ciências e Tecnologia
Unesp – Universidade Estadual Paulista
Presidente Prudente/SP, Brasil

DEPARTAMENTO DE MATEMÁTICA E COMPUTAÇÃO

Força Bruta

- É a abordagem mais direta para resolver um problema
- Geralmente ela é baseada nas regras e definições dos conceitos envolvidos
- Essa estratégia é dependente do poder computacional (força) e não da inteligência na modelagem do algoritmo

Força Bruta

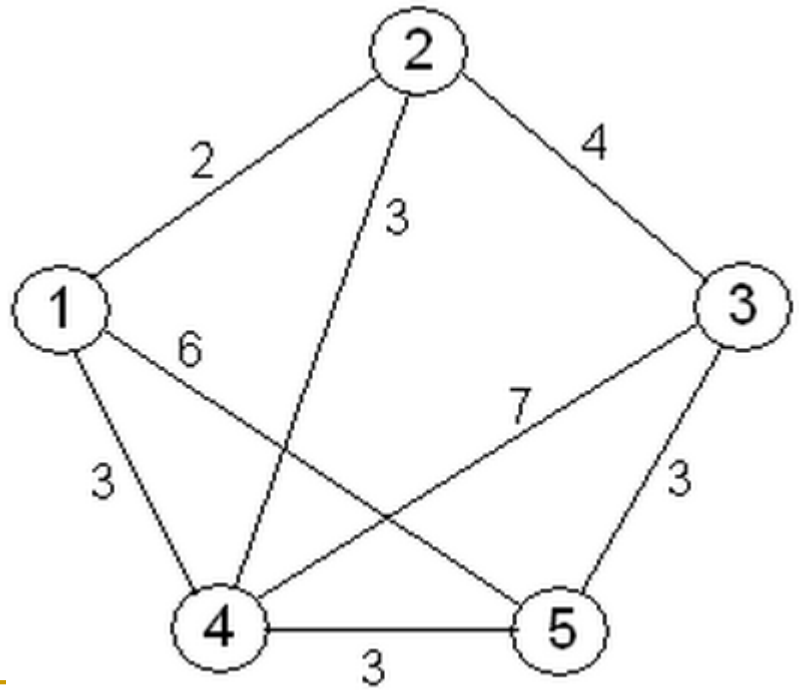
- Exemplos
  - Ordenar elementos em ordem crescente
    - Selection Sort
    - Bubble Sort
  - Busca sequencial
  - Verificar o casamento de strings
  - Multiplicação de matrizes

Força Bruta

- Alguns problema não possuem uma solução algorítmica eficiente
- Nesses casos, o esforço computacional necessário para a sua resolução cresce exponencialmente com o tamanho do problema
- Heurísticas e afins podem ser empregadas, mas não asseguram a obtenção de uma solução ótima

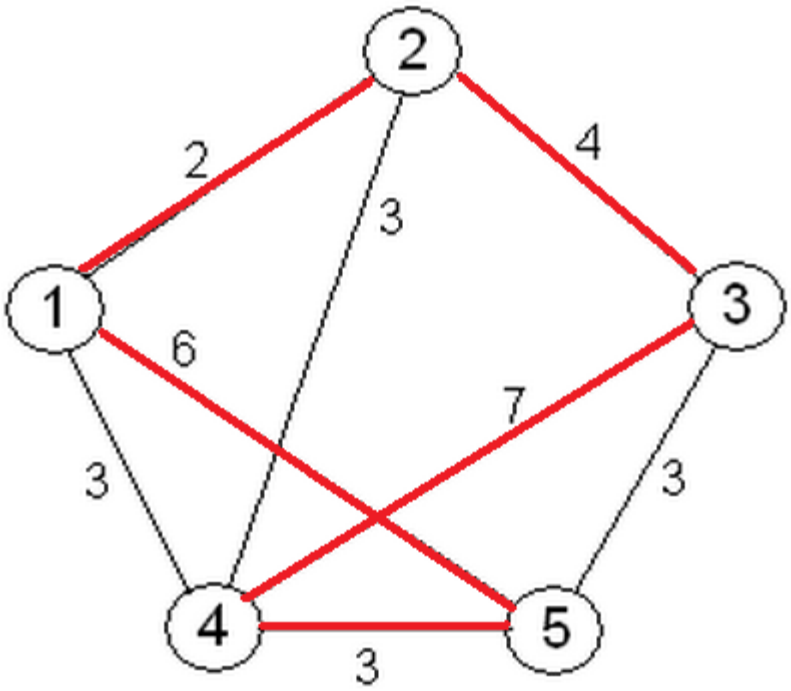
Força Bruta

- Ex: problema do caixeiro viajante
  - Passar por cada cidade uma única vez e voltar à origem, considerando o circuito de custo mínimo



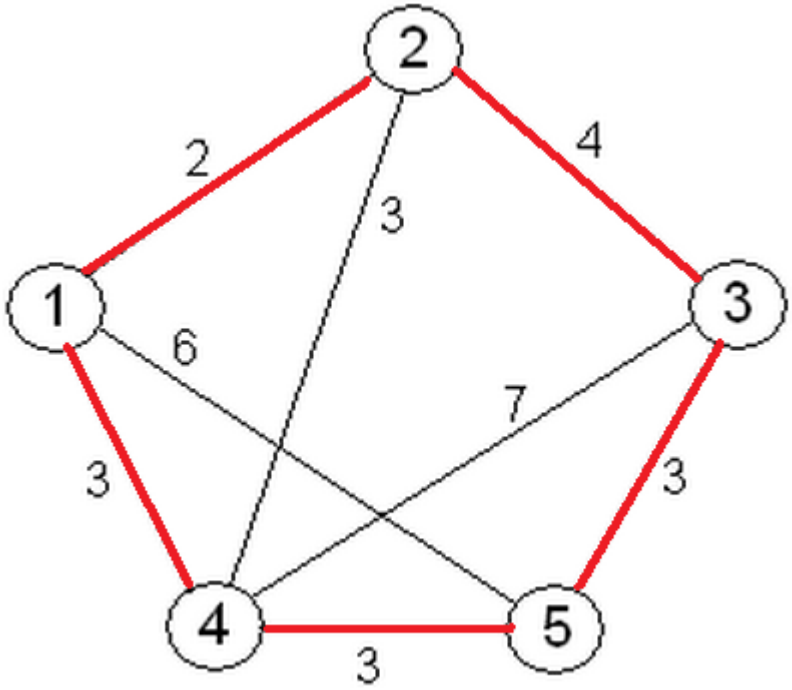
Força Bruta

- Ex: problema do caixeiro viajante
  - Passar por cada cidade uma única vez e voltar à origem, considerando o menor caminho para esse circuito
- [1, 2, 3, 4, 5, 1]
- Custo = 22



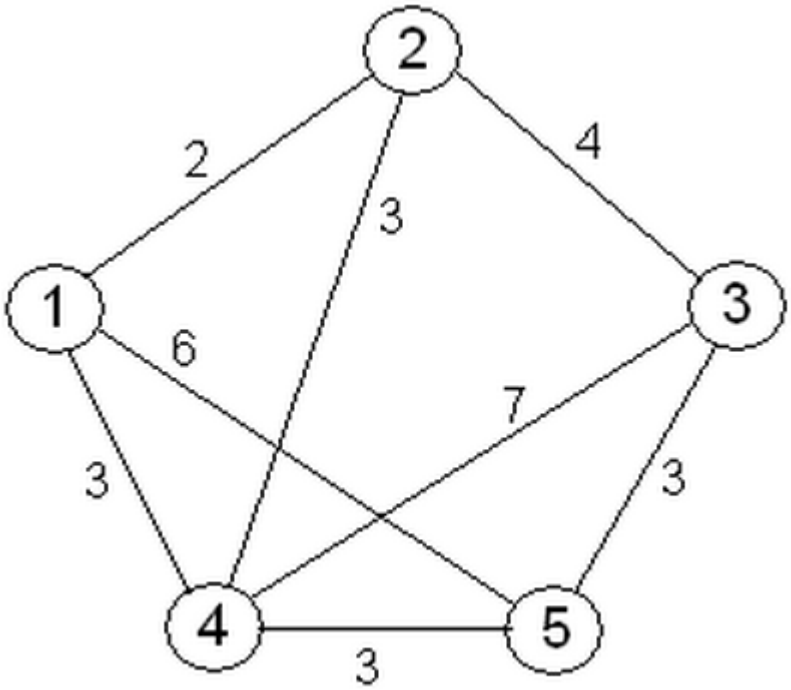
Força Bruta

- Ex: problema do caixeiro viajante
  - Passar por cada cidade uma única vez e voltar à origem, considerando o menor caminho para esse circuito
- [1, 2, 3, 5, 4, 1]
- Custo = 15



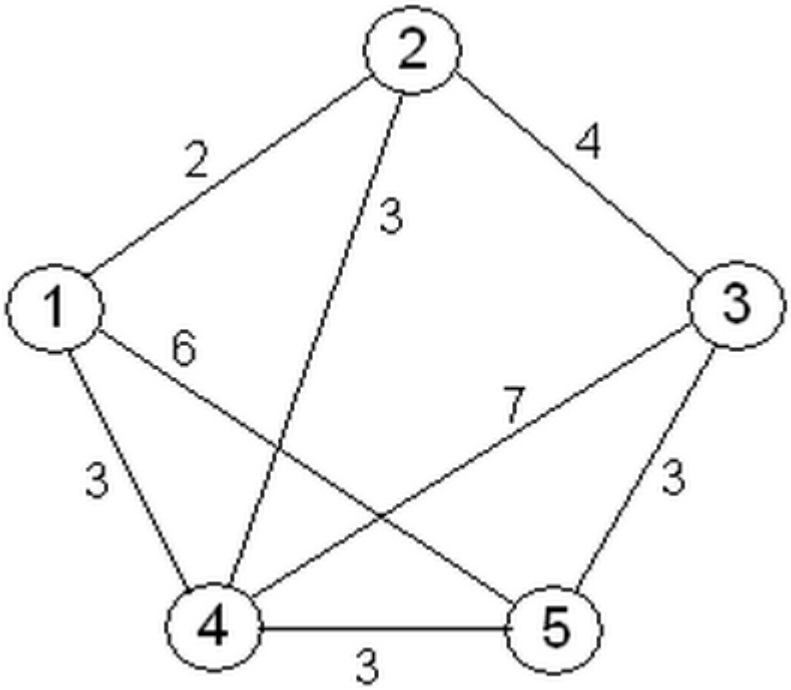
Força Bruta

- Ex: problema do caixeiro viajante
  - Passar por cada cidade uma única vez e voltar à origem, considerando o menor caminho para esse circuito
- [1, 2, 5, 3, 4, 1]
- Custo = ?



Força Bruta

- Ex: problema do caixeiro viajante
  - Passar por cada cidade uma única vez e voltar à origem, considerando o menor caminho para esse circuito
- [1, 2, 5, 3, 4, 1]
- Custo = ?
- NÃO HÁ CIRCUITO



Tentativa e Erro

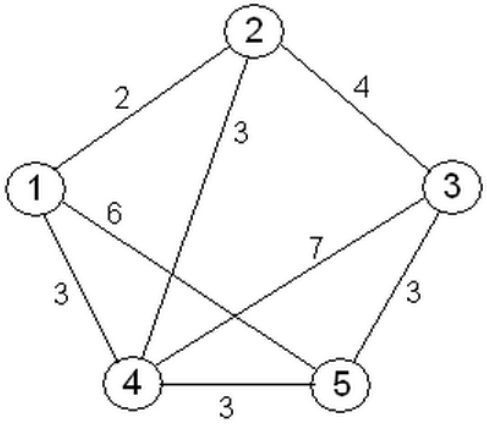
- Existem vários problemas que são difíceis de resolver, isto é, difíceis de propor uma solução algorítmica eficiente
- Existem duas técnicas de projeto de algoritmos que visam diminuir o espaço em busca da solução
  - Embora, no pior caso, elas ainda enfrentem a explosão exponencial da busca exaustiva pela solução

Tentativa e Erro

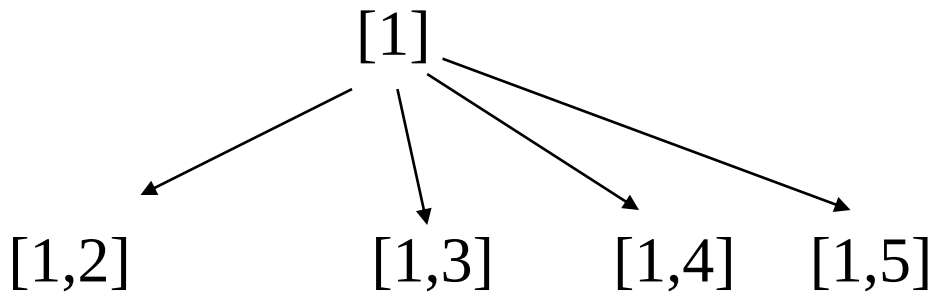
- Para melhorar a busca exaustiva das possíveis soluções duas técnicas são propostas
  - *Backtracking*
    - Do inglês: recuar, retroceder
  - *Branch-and-bound*
    - Do inglês: ramificar e limitar
- Ambas são baseadas na construção de uma árvore de estados, na qual os nós refletem uma escolha feita em direção da solução

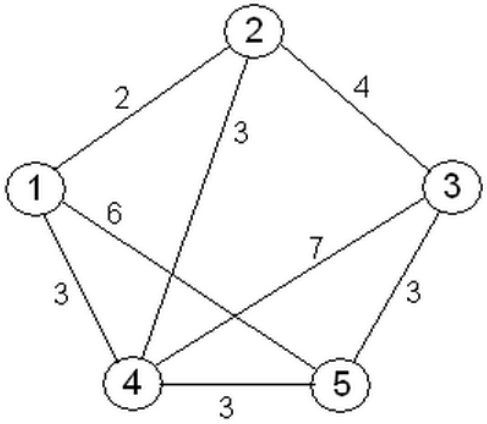
Tentativa e Erro

[1]

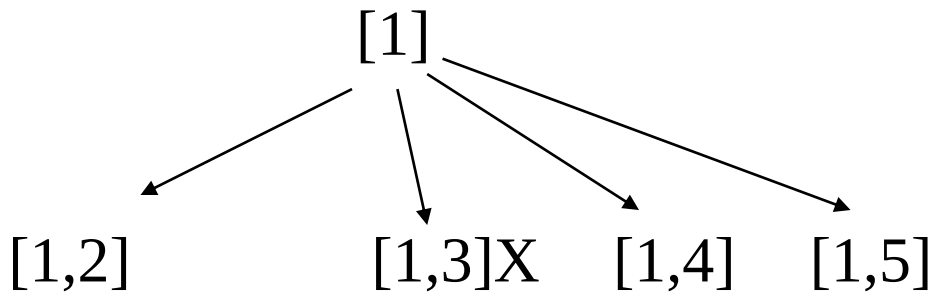


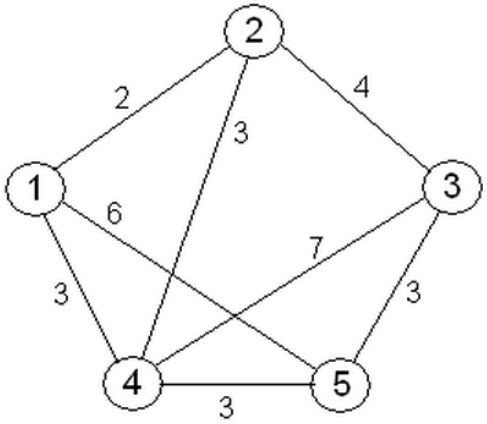
Tentativa e Erro



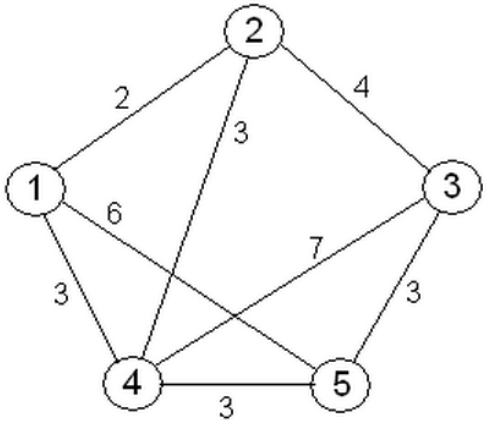


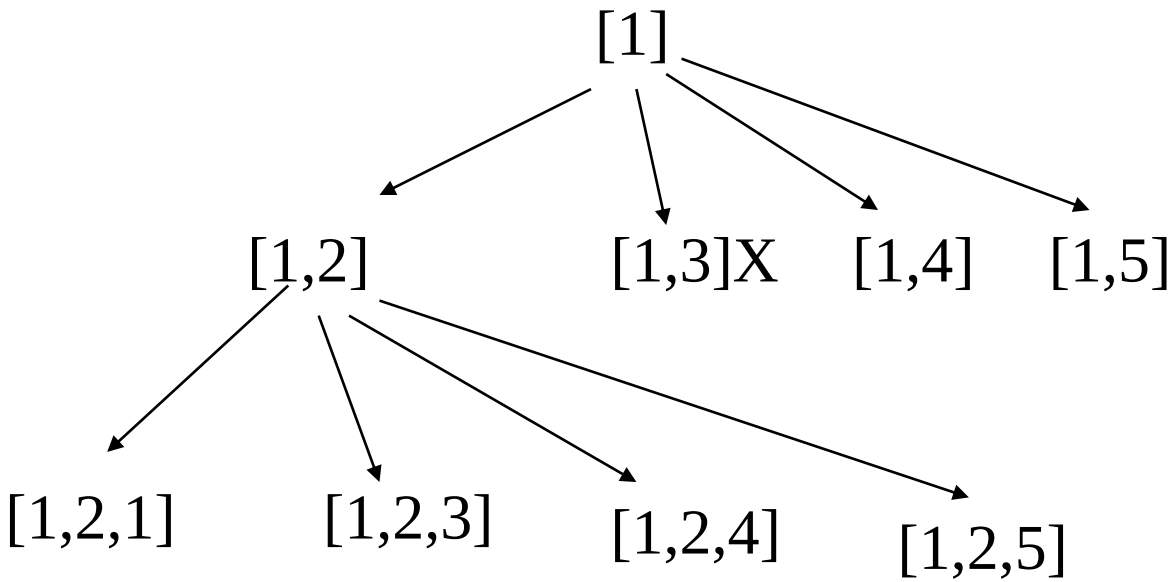
Tentativa e Erro



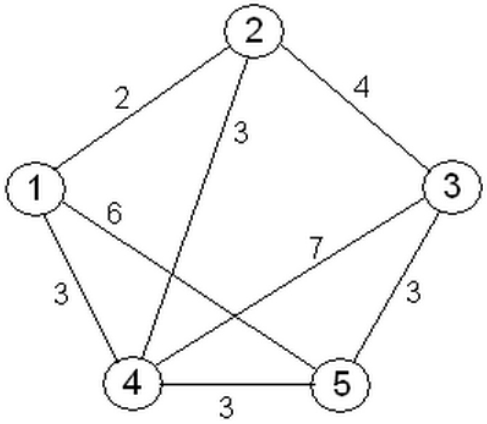


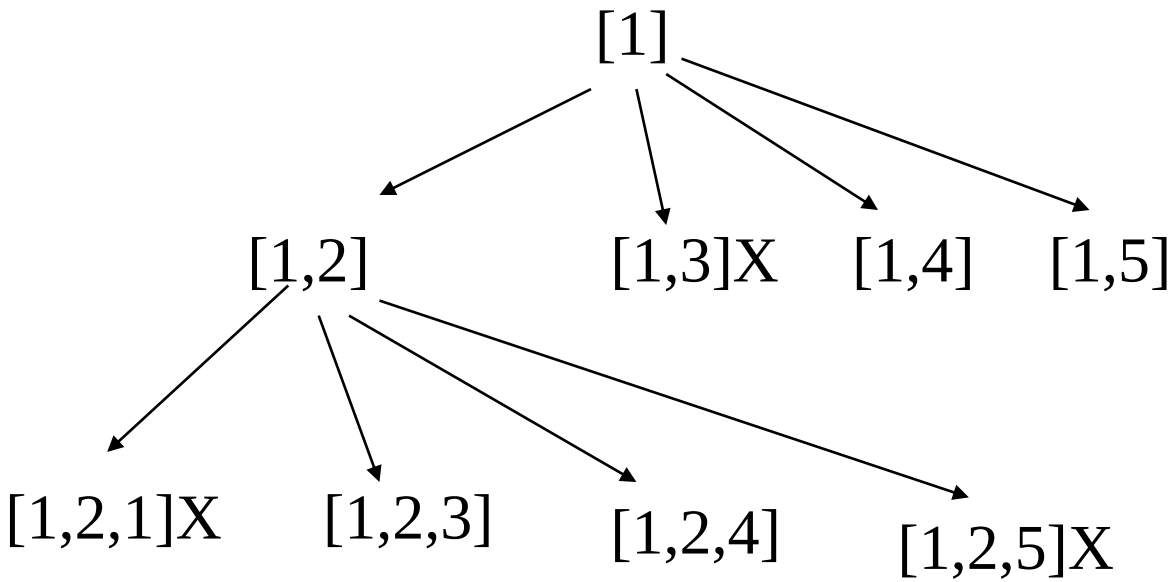
Tentativa e Erro



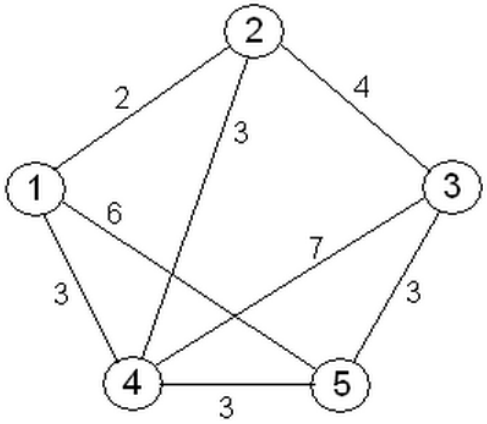


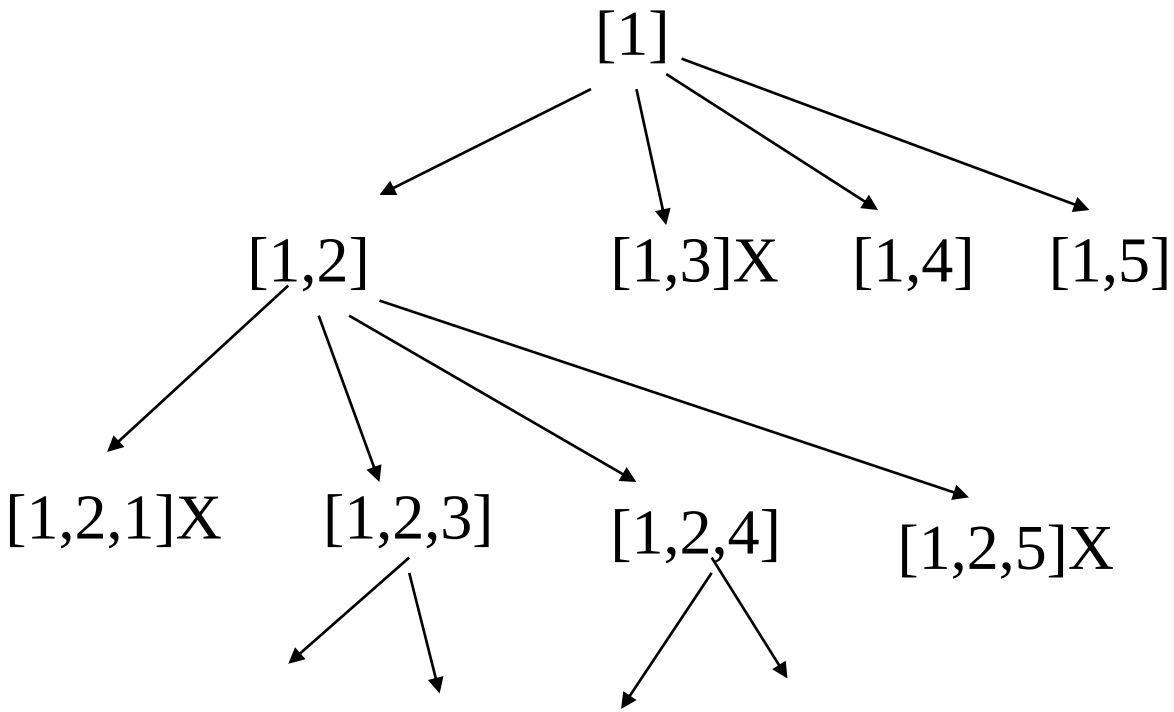
Tentativa e Erro



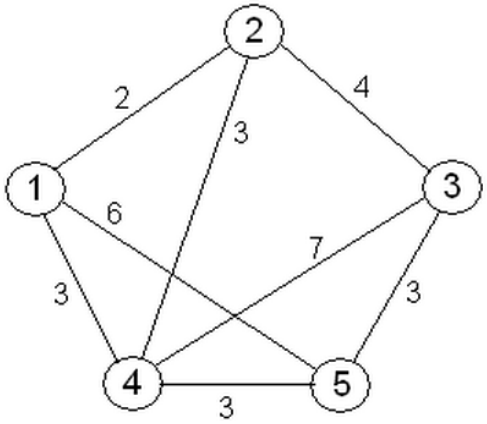


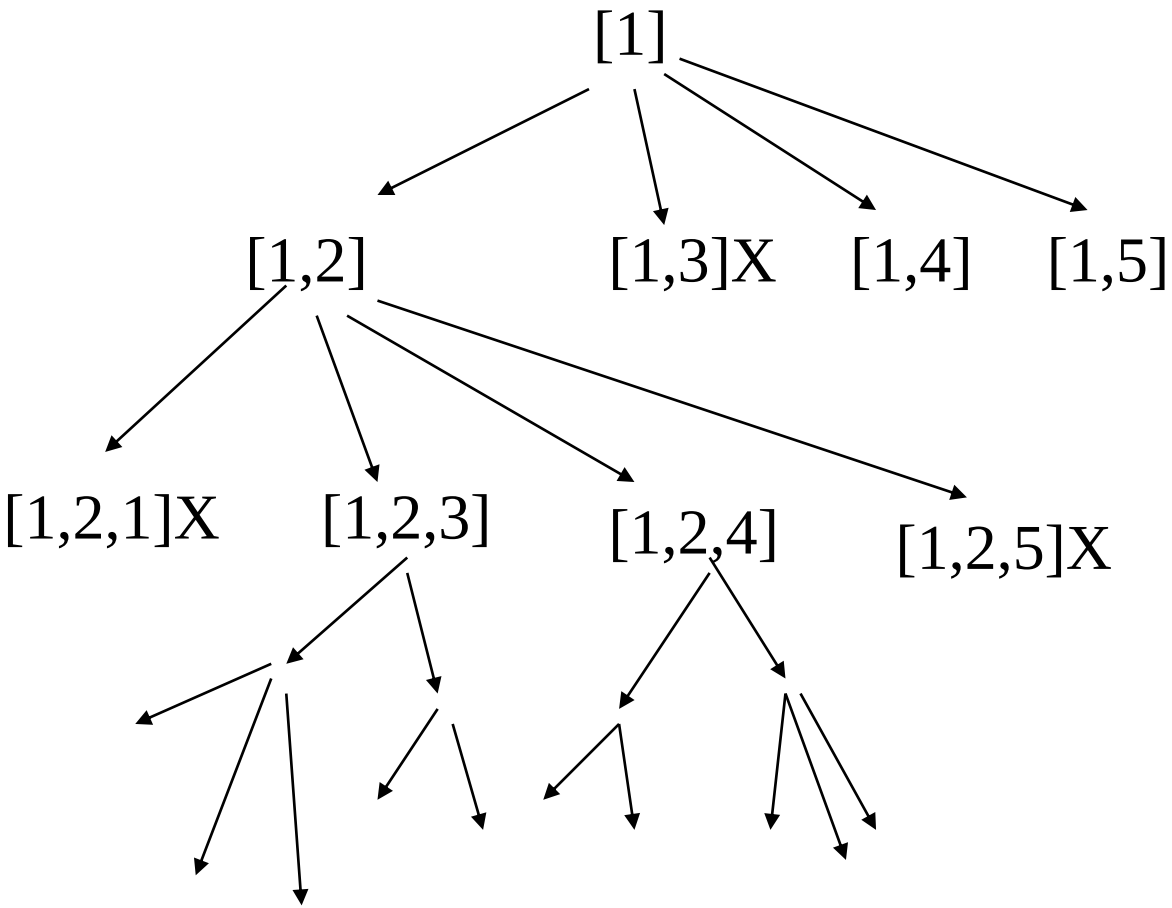
Tentativa e Erro



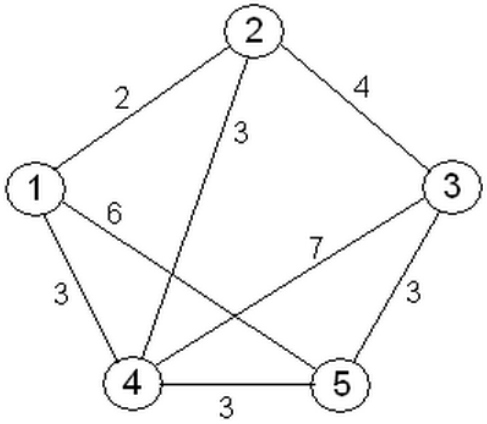


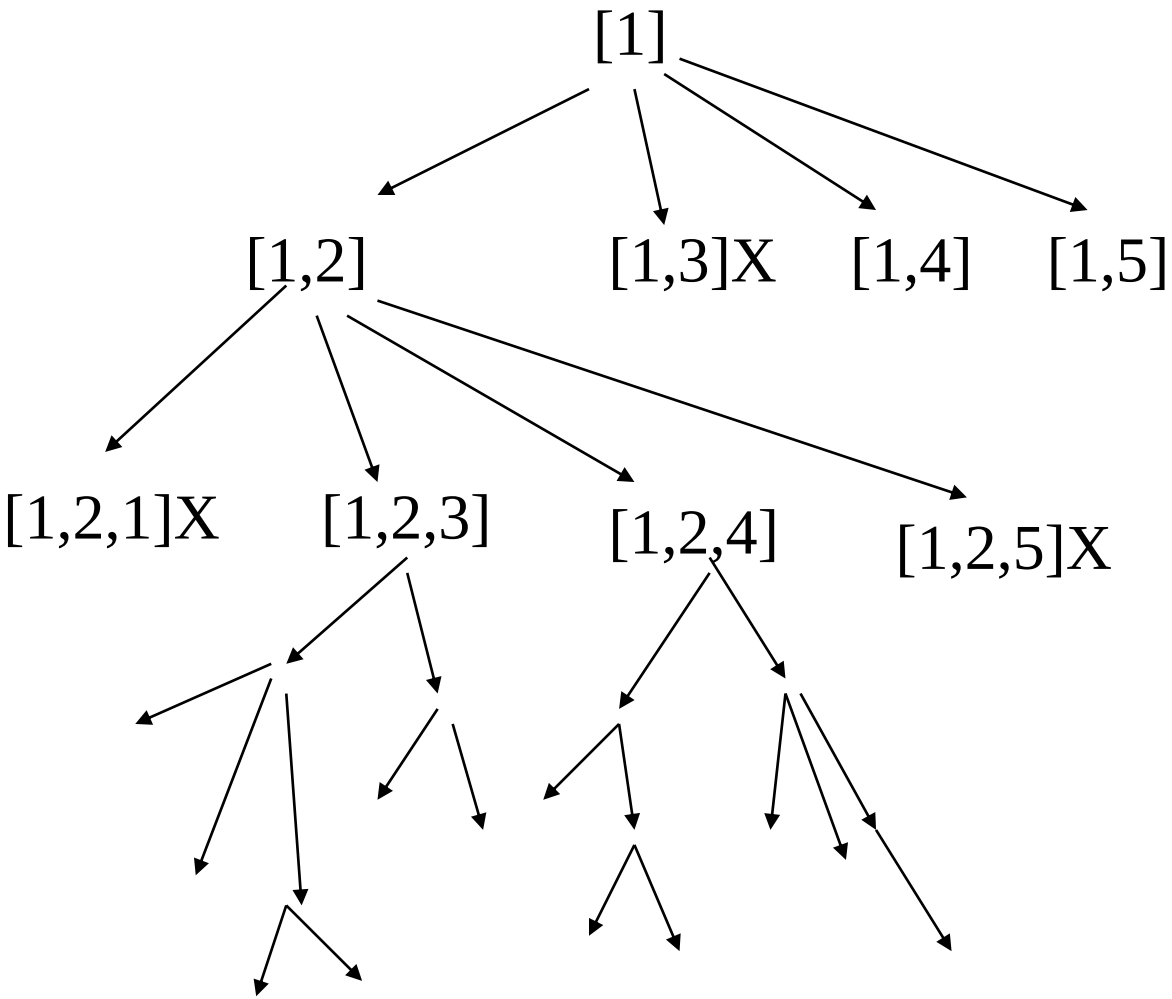
Tentativa e Erro



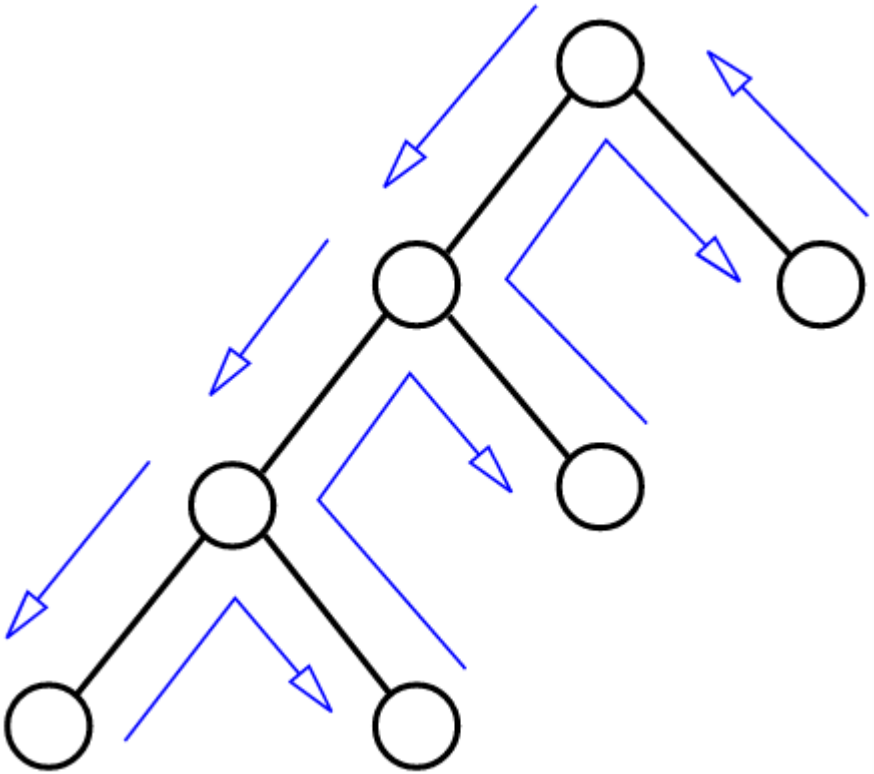


Tentativa e Erro

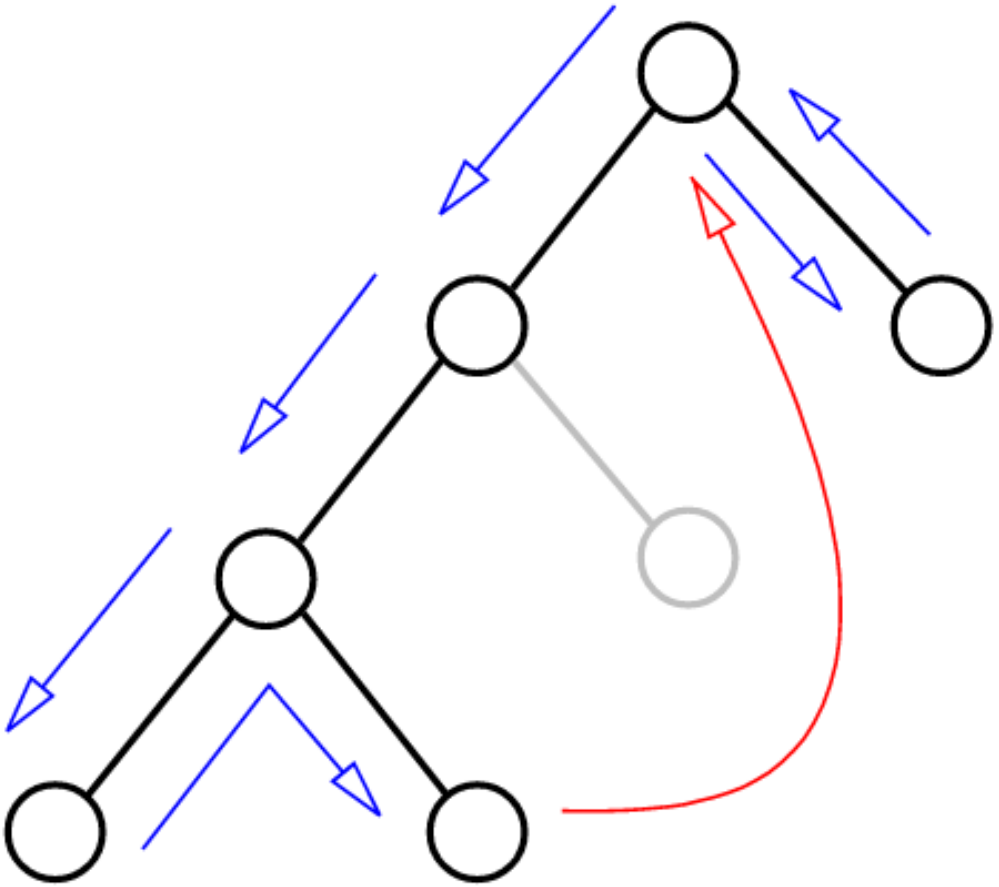




Backtracking



Backtracking

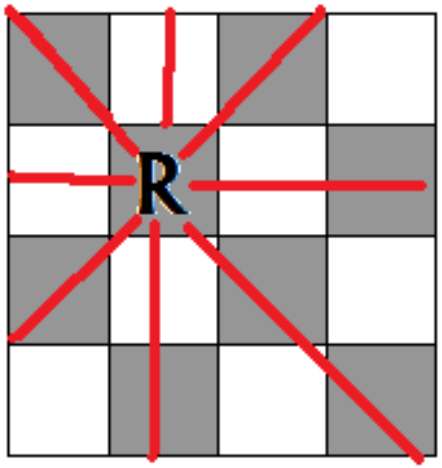


Backtracking

- Exemplo: Labirinto
  - Acessados em 03/02/2015
    - https://www.youtube.com/watch?v=h0aXgiL-lws
  - Acessados em 2013
    - http://athena.ecs.csus.edu/~wang/backtrack/maze/maze1/mazetraversal.html
    - http://www.cs.lafayette.edu/~collinsw/maze/MazeApplet.html

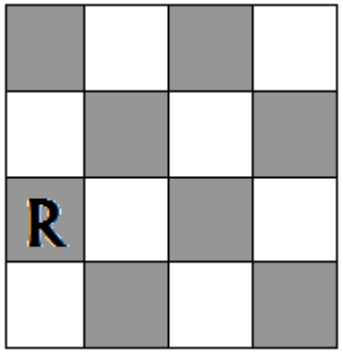
Problema das N Rainhas

- Tentar posicionar N Rainhas em um tabuleiro NxN sem que uma rainha mate a outra
  - N >= 4
- Uma rainha consegue matar outra rainha se ela estiver na mesma linha, na mesma coluna ou na mesma diagonal



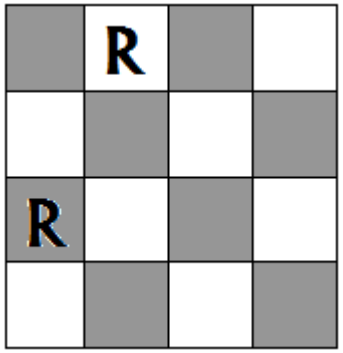
Problema das N Rainhas

- Tentar posicionar N Rainhas em um tabuleiro NxN sem que uma rainha mate a outra
  - N >= 4
- Uma rainha consegue matar outra rainha se ela estiver na mesma linha, na mesma coluna ou na mesma diagonal



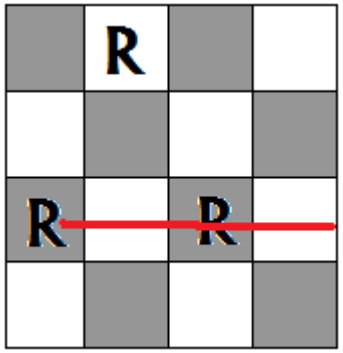
Problema das N Rainhas

- Tentar posicionar N Rainhas em um tabuleiro NxN sem que uma rainha mate a outra
  - N >= 4
- Uma rainha consegue matar outra rainha se ela estiver na mesma linha, na mesma coluna ou na mesma diagonal



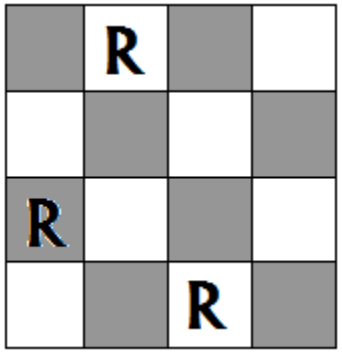
Problema das N Rainhas

- Tentar posicionar N Rainhas em um tabuleiro NxN sem que uma rainha mate a outra
  - N >= 4
- Uma rainha consegue matar outra rainha se ela estiver na mesma linha, na mesma coluna ou na mesma diagonal



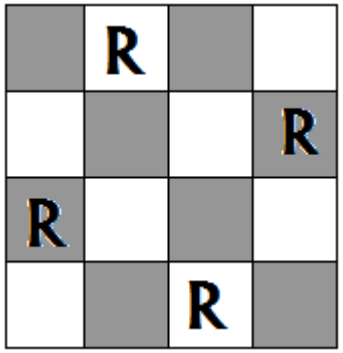
Problema das N Rainhas

- Tentar posicionar N Rainhas em um tabuleiro NxN sem que uma rainha mate a outra
  - N >= 4
- Uma rainha consegue matar outra rainha se ela estiver na mesma linha, na mesma coluna ou na mesma diagonal



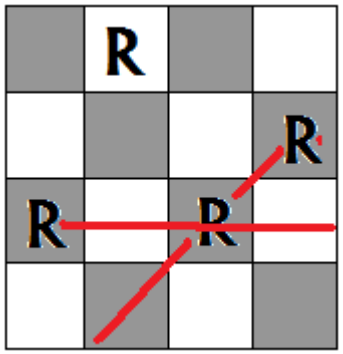
Problema das N Rainhas

- Tentar posicionar N Rainhas em um tabuleiro NxN sem que uma rainha mate a outra
  - N >= 4
- Uma rainha consegue matar outra rainha se ela estiver na mesma linha, na mesma coluna ou na mesma diagonal

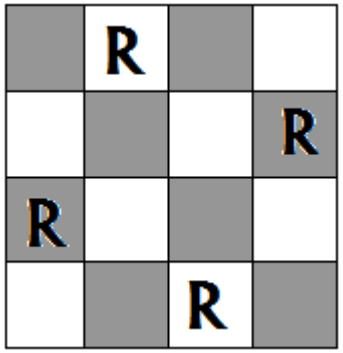


Problema das N Rainhas

- Podemos considerar uma lista, em que a posição na lista indica a coluna do tabuleiro e o valor armazenado indica a linha do tabuleiro
- Ex: [3, 1, 3, 2]



[3, 1, 4, 2]



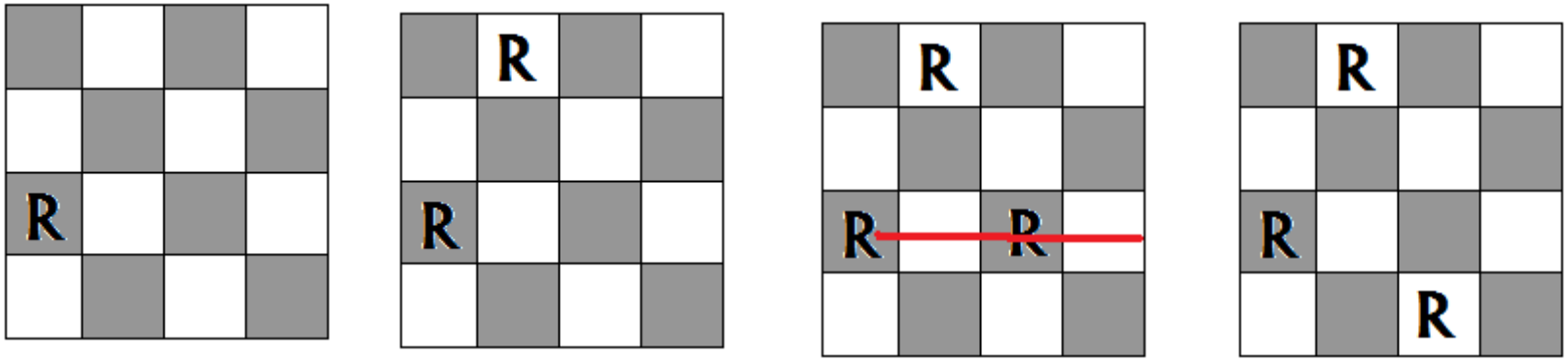
Problema das N Rainhas

- Podemos utilizar o Backtracking para auxiliar na resolução deste problema
- Partindo de um estado inicial, faremos o caminho pela árvore de soluções, procurando um estado em que as N rainhas estejam posicionas no tabuleiro de tamanho NxN, sem conflitos

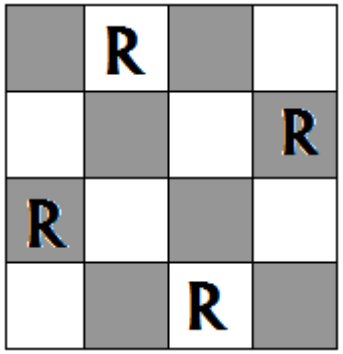
Problema das N Rainhas

- Exemplo de estados

[3, 0, 0, 0] [3, 1, 0 ,0] [3, 1, 3, 0] [3, 1, 4, 0]



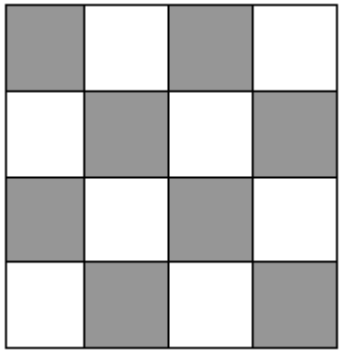
Solução [3, 1, 4, 2]



Problema das N Rainhas

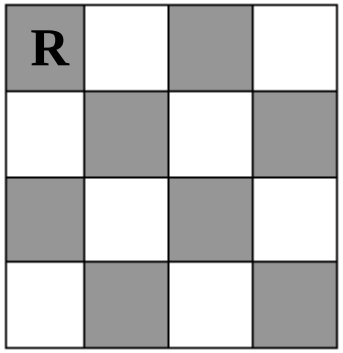
- Construir árvore de estados e executar o backtracking para um exemplo com N = 4

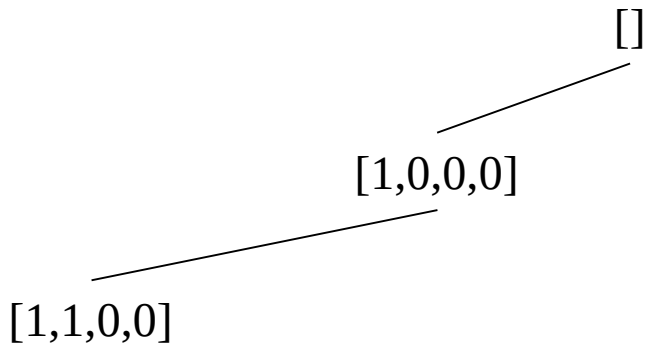
[]

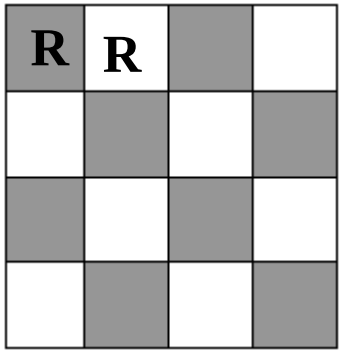


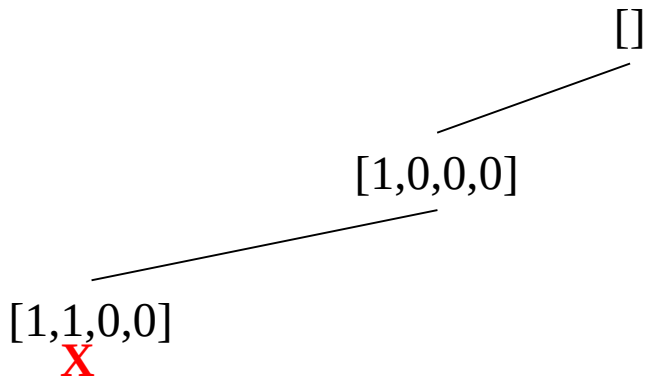
[]

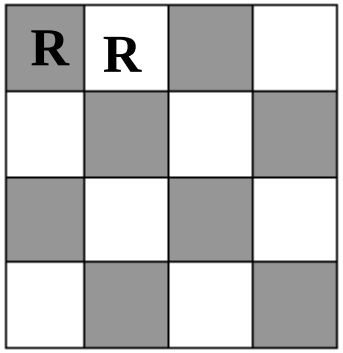
[1,0,0,0]

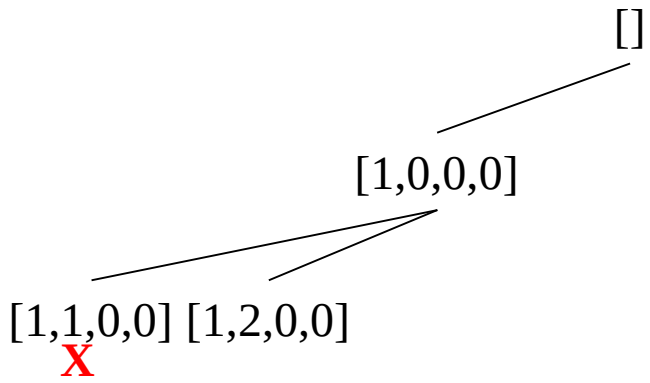


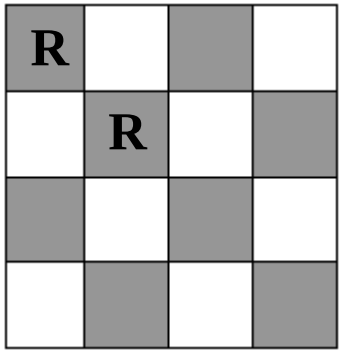

[]
[1,0,0,0]
[1,1,0,0]

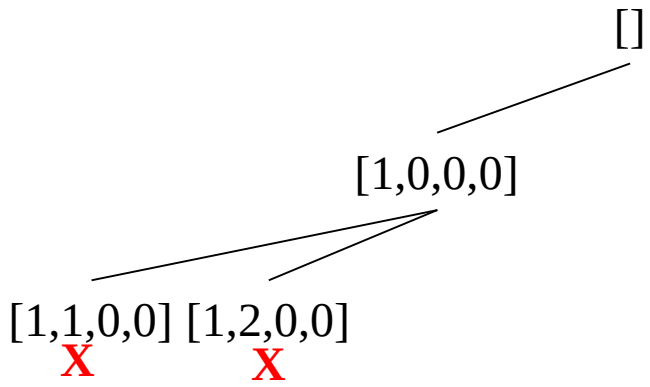

R
R

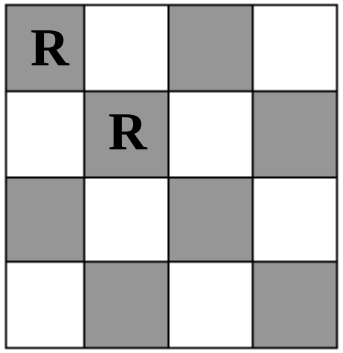

[]
[1,0,0,0]
[1,1,0,0]
X

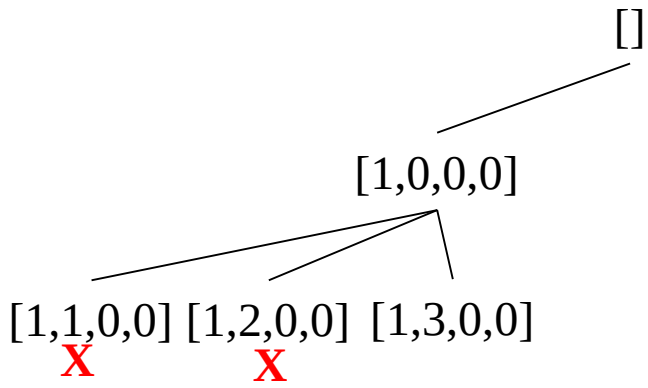

R
R

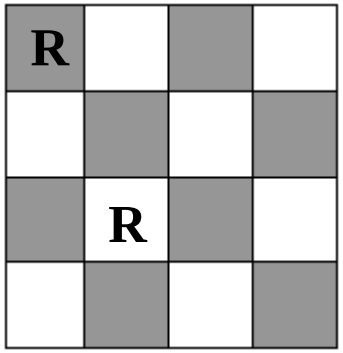

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
X

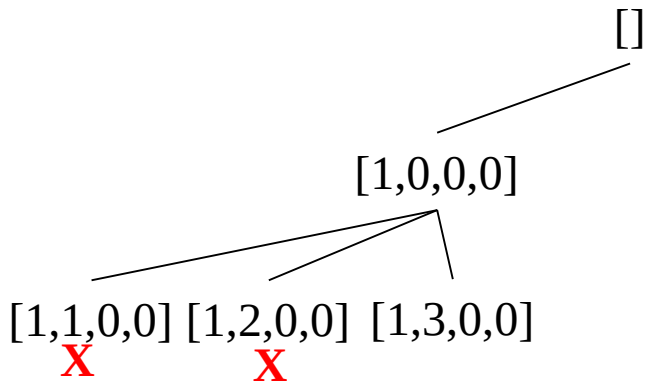

R
R

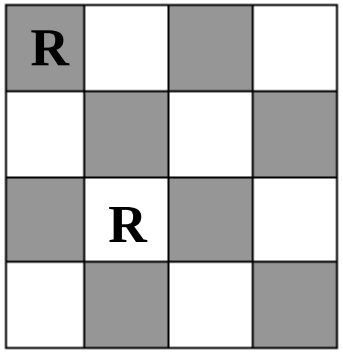

[]
[1,0,0,0]
[1,1,0,0]
X
[1,2,0,0]
X

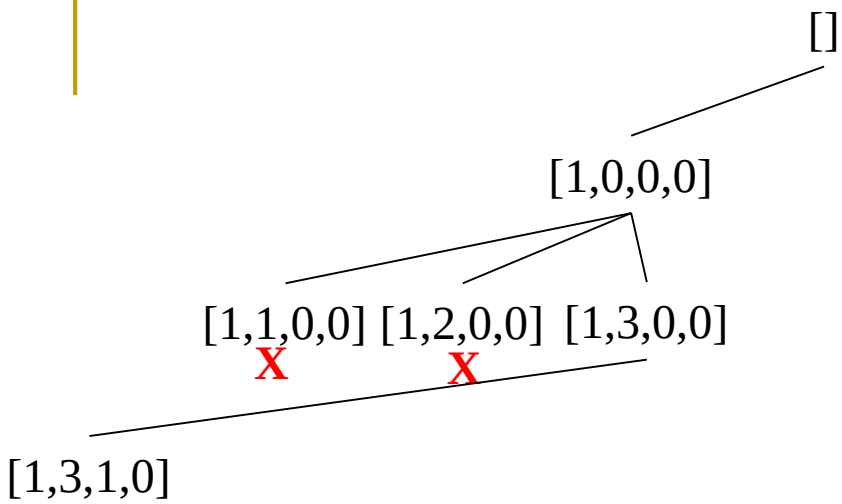

R
R

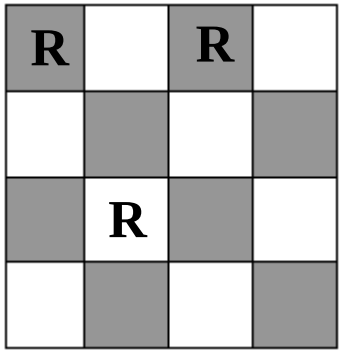

[]
[1,0,0,0]
[1,1,0,0]
X
[1,2,0,0]
X
[1,3,0,0]

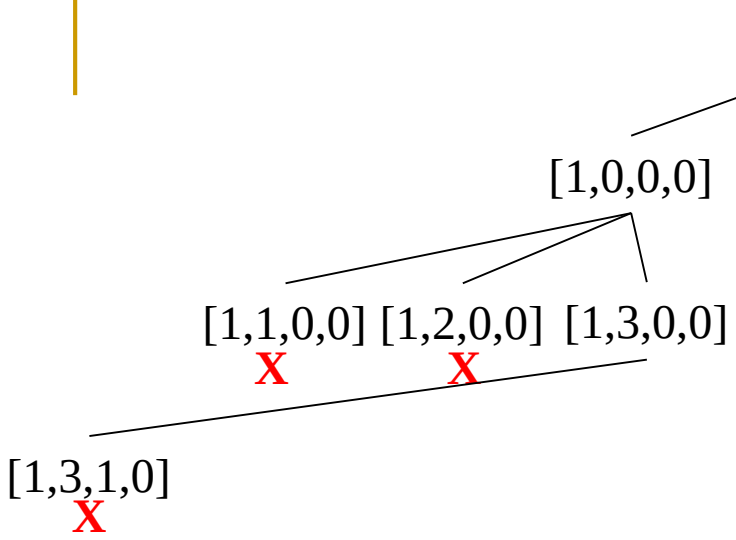

R
R

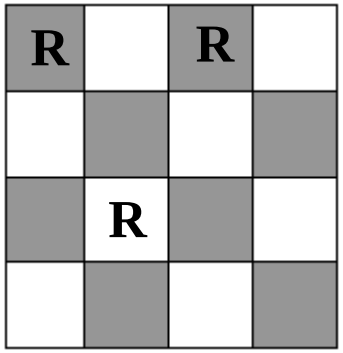

[]
[1,0,0,0]
[1,1,0,0]
X
[1,2,0,0]
X
[1,3,0,0]

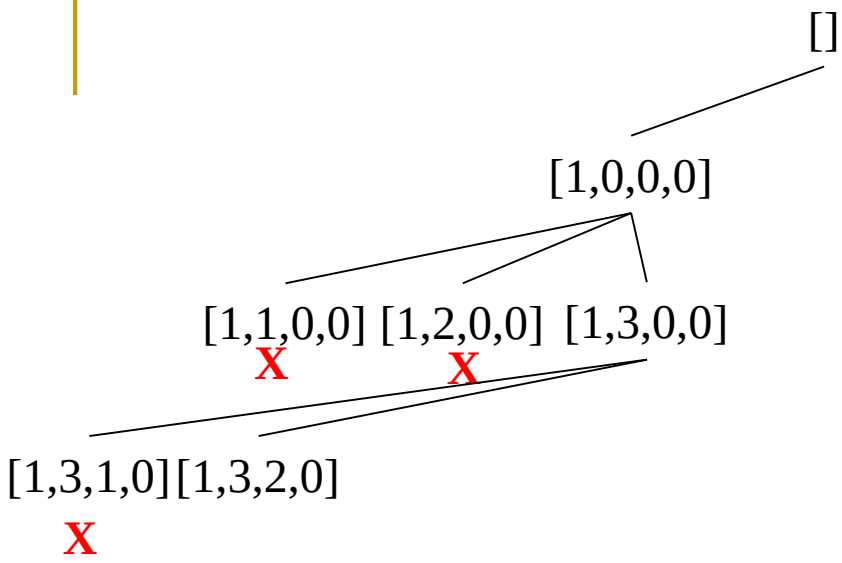

R
R

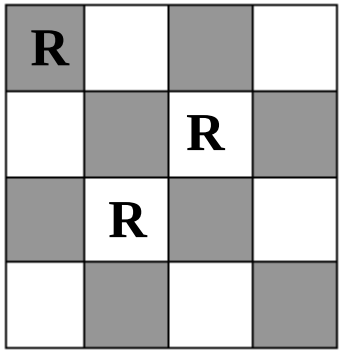

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
X
X
[1,3,1,0]

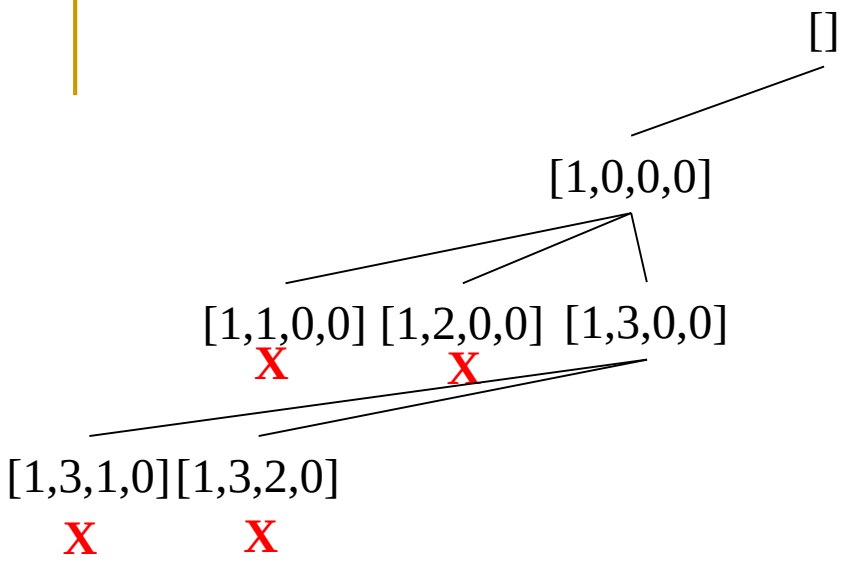

R
R
R

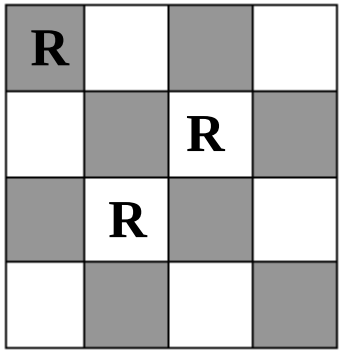

[]
[1,0,0,0]
[1,1,0,0]
X
[1,2,0,0]
X
[1,3,0,0]
[1,3,1,0]
X

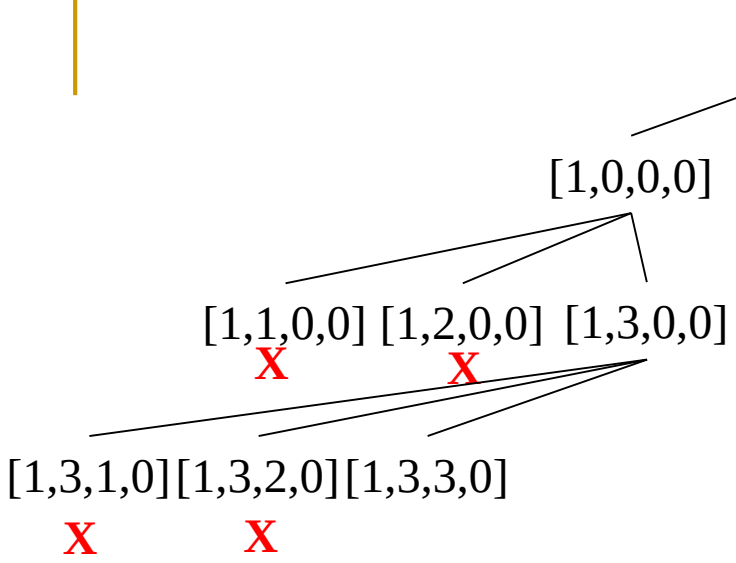

R
R
R

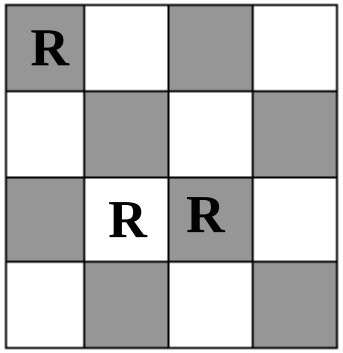

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
X

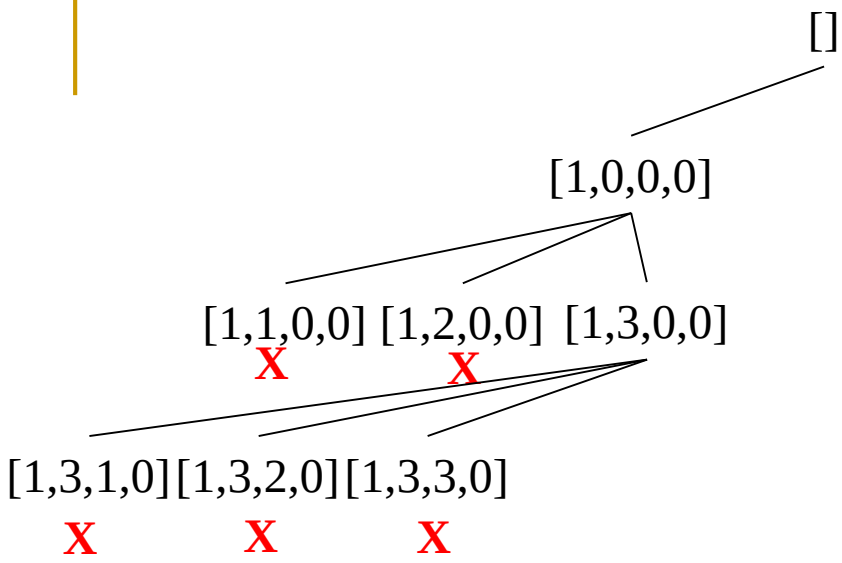

R
R
R

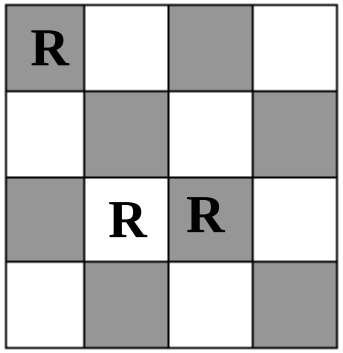

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
X
X

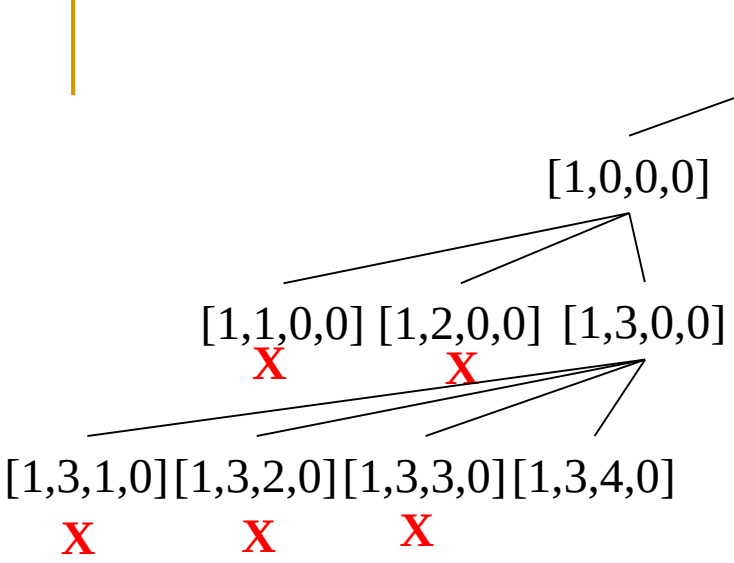

R
R
R

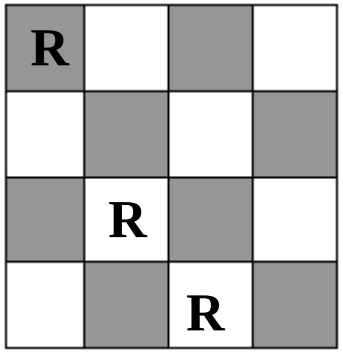

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
X
X

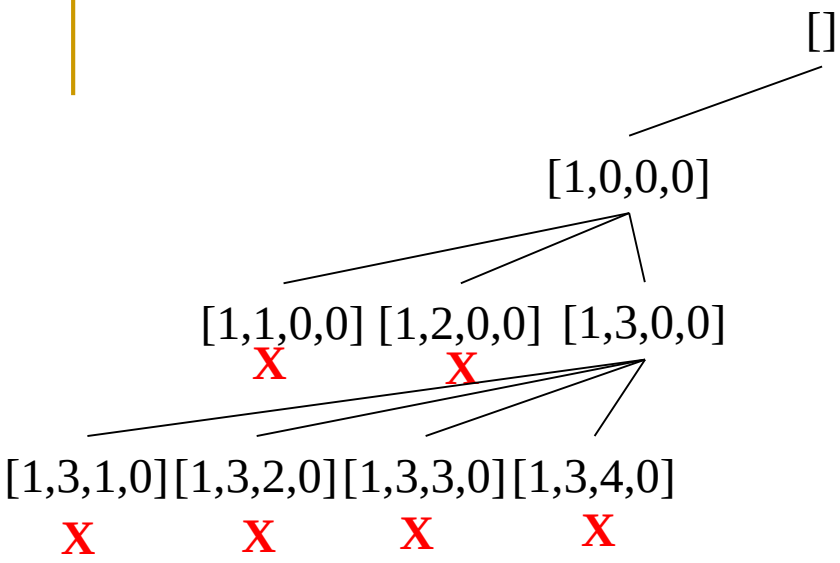

R
R
R

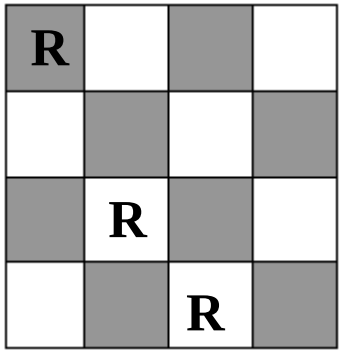

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
X
X
X

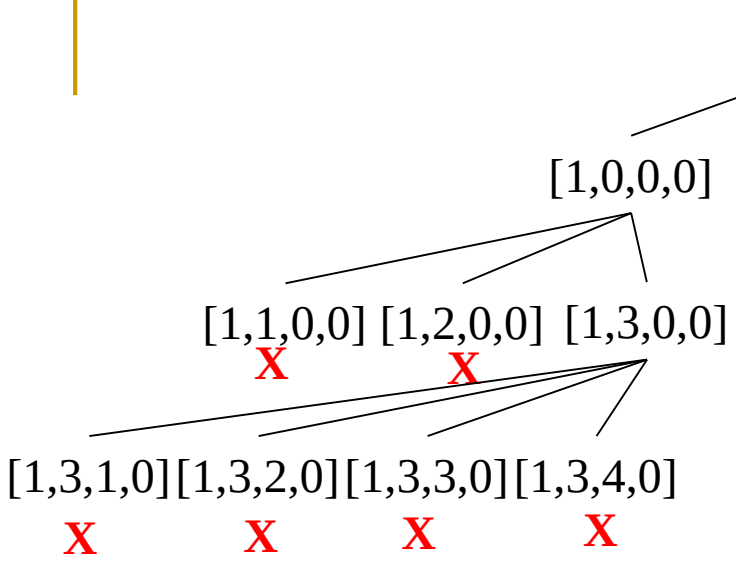

R
R
R


[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
X
X
X

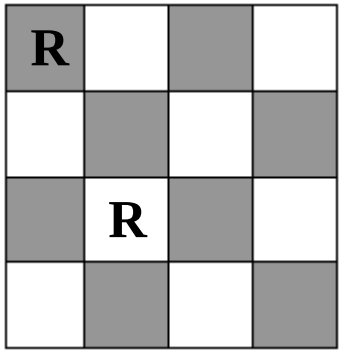

R
R
R

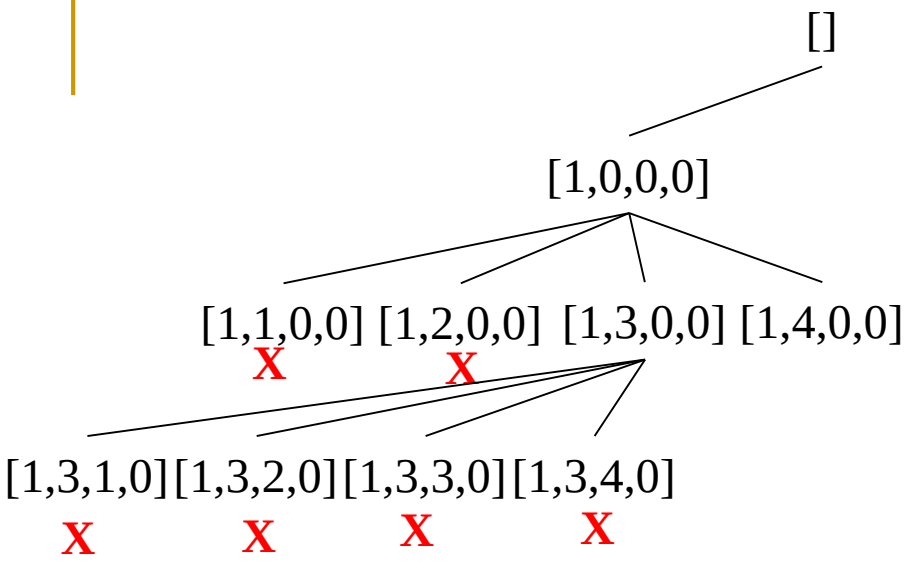

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
X
X
X
X

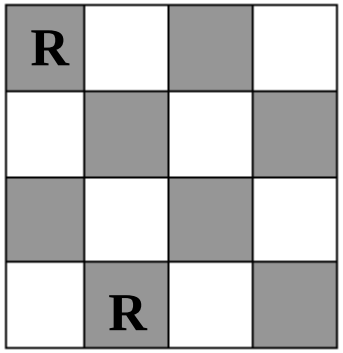

R
R
R

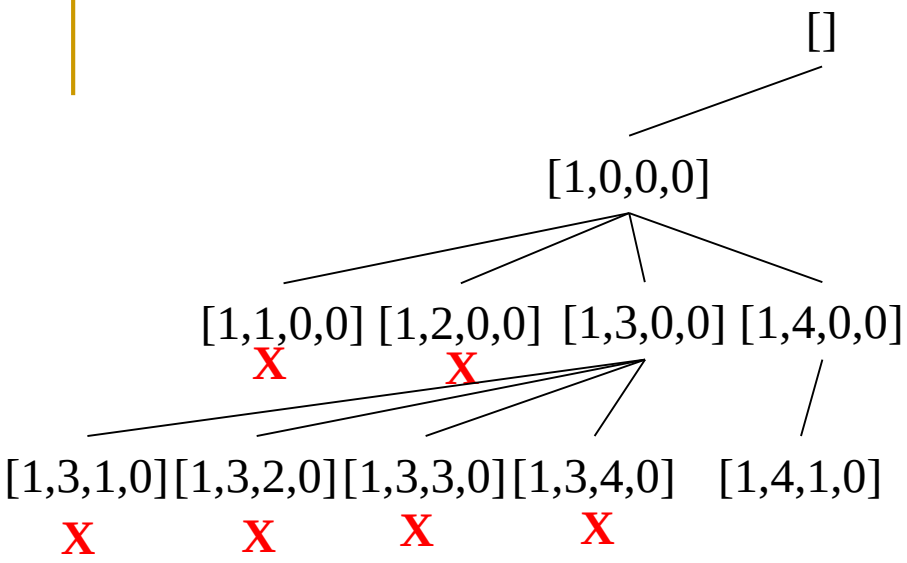


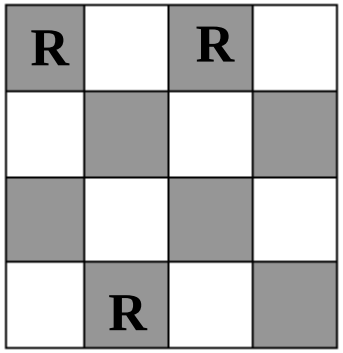
BACKTRACKING

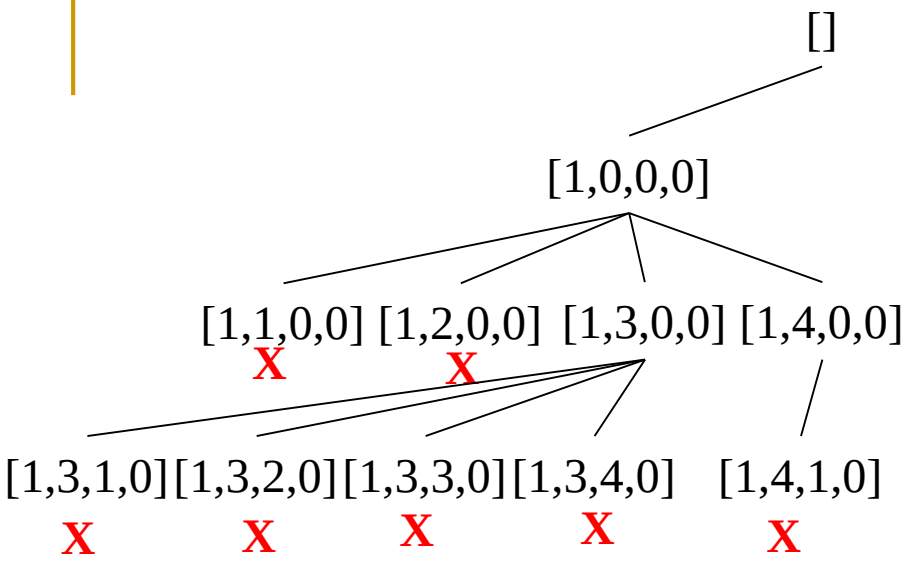


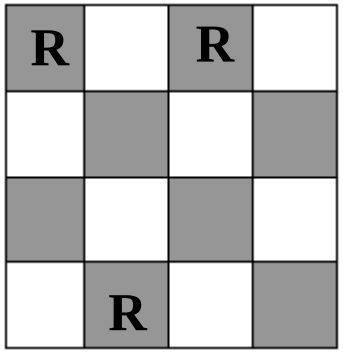

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
X
X
X
X

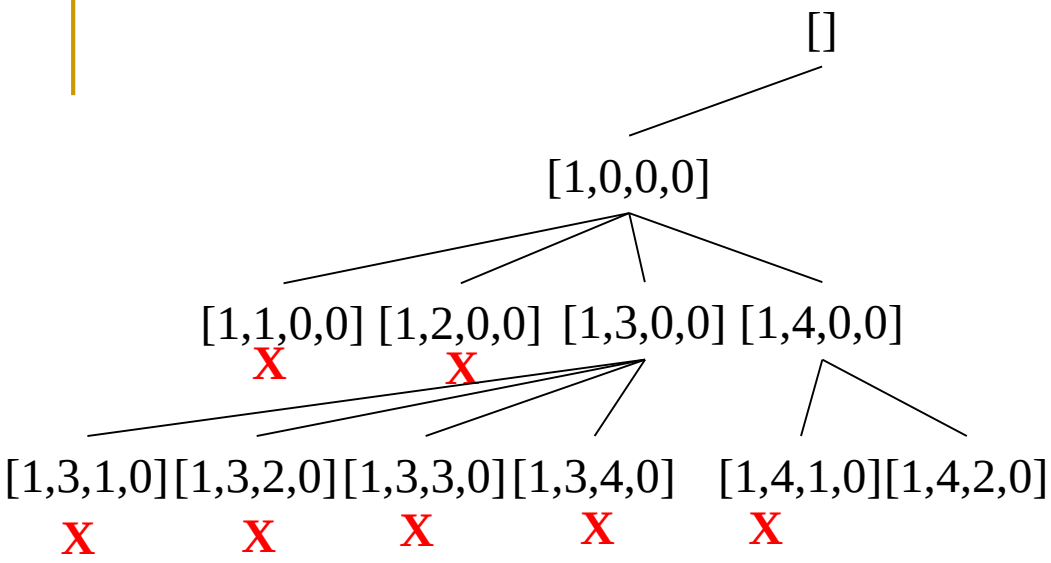

R
R

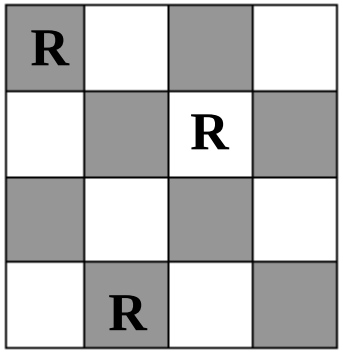

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
X
X
X
X

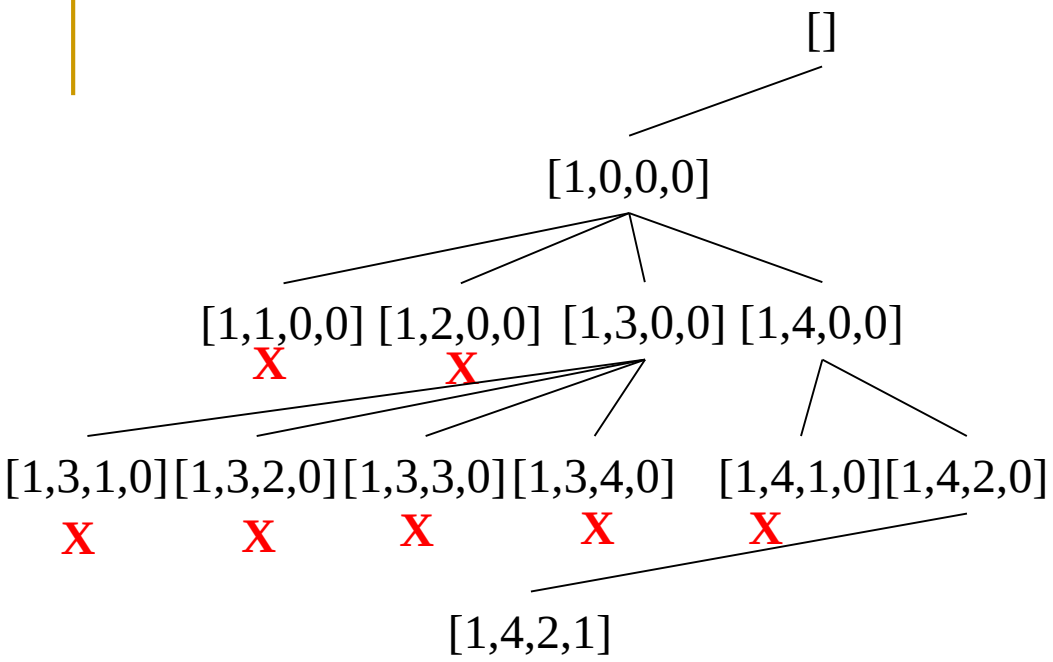

R
R
R

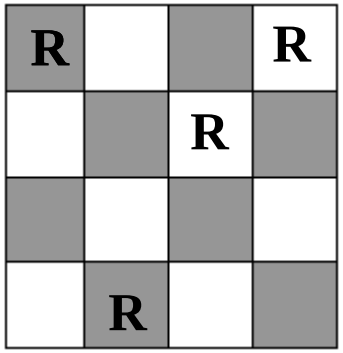

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
X
X
X
X
X

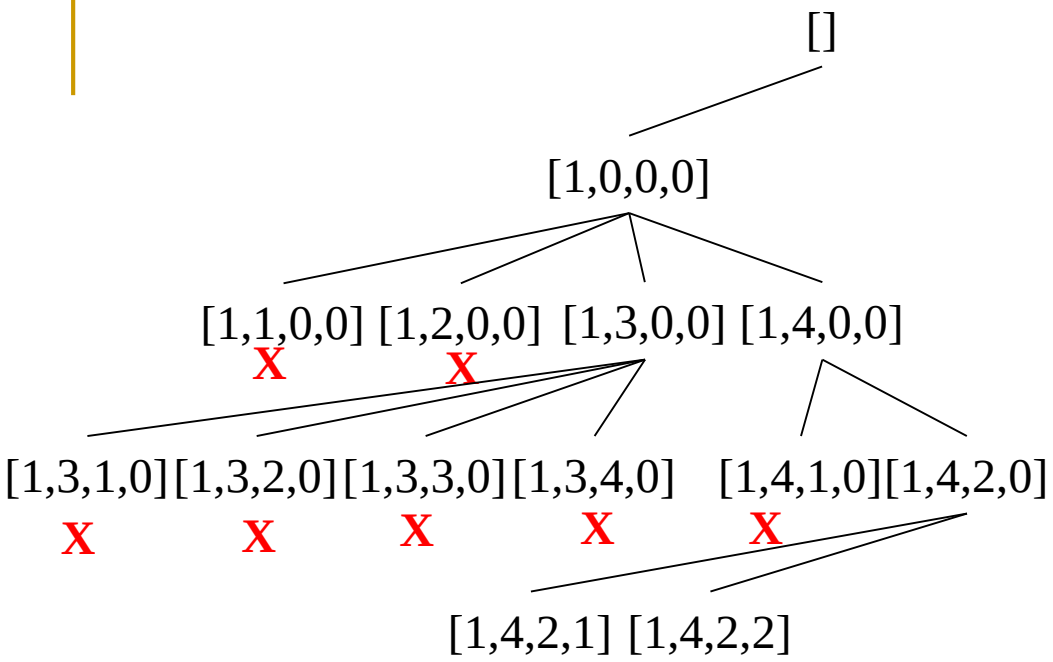

R
R
R

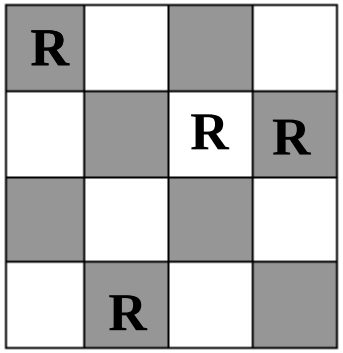

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
X
X
X
X
X

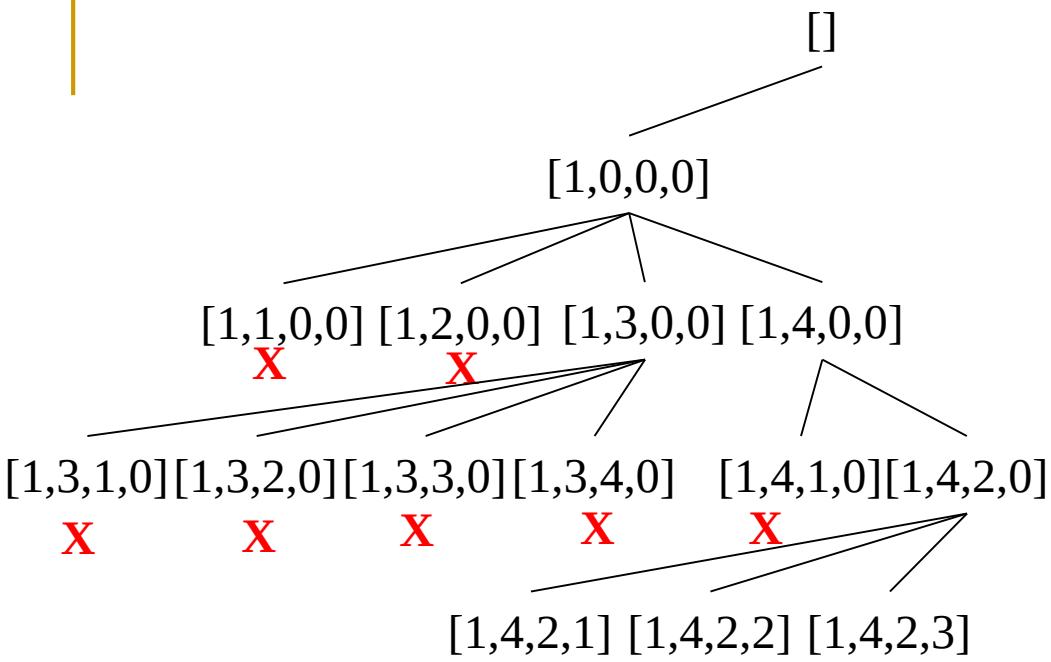

R
R
R

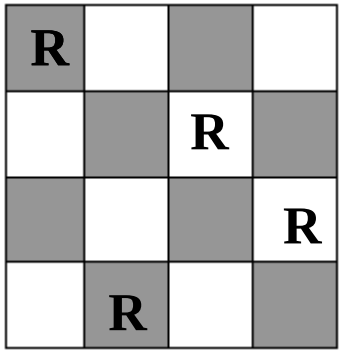

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
X
X
X
X
X
[1,4,2,1]

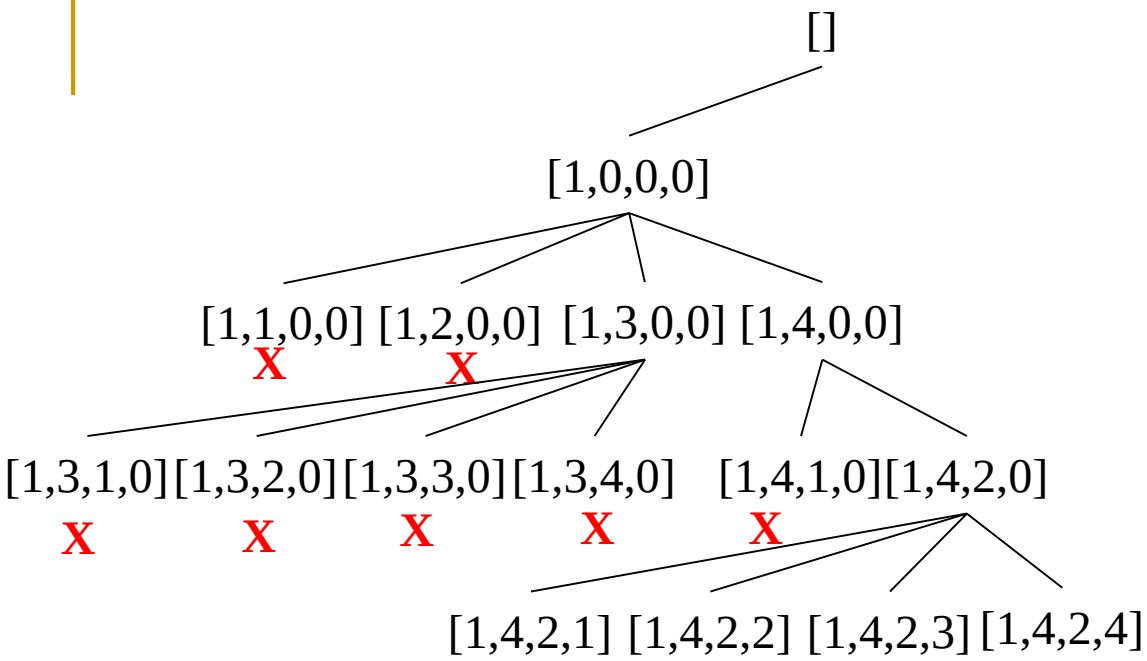

R
R
R
R

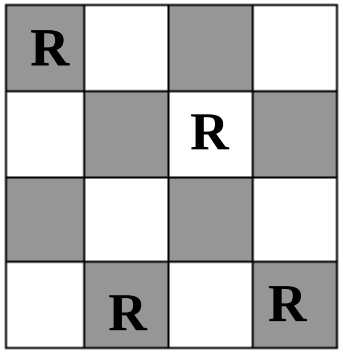

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]

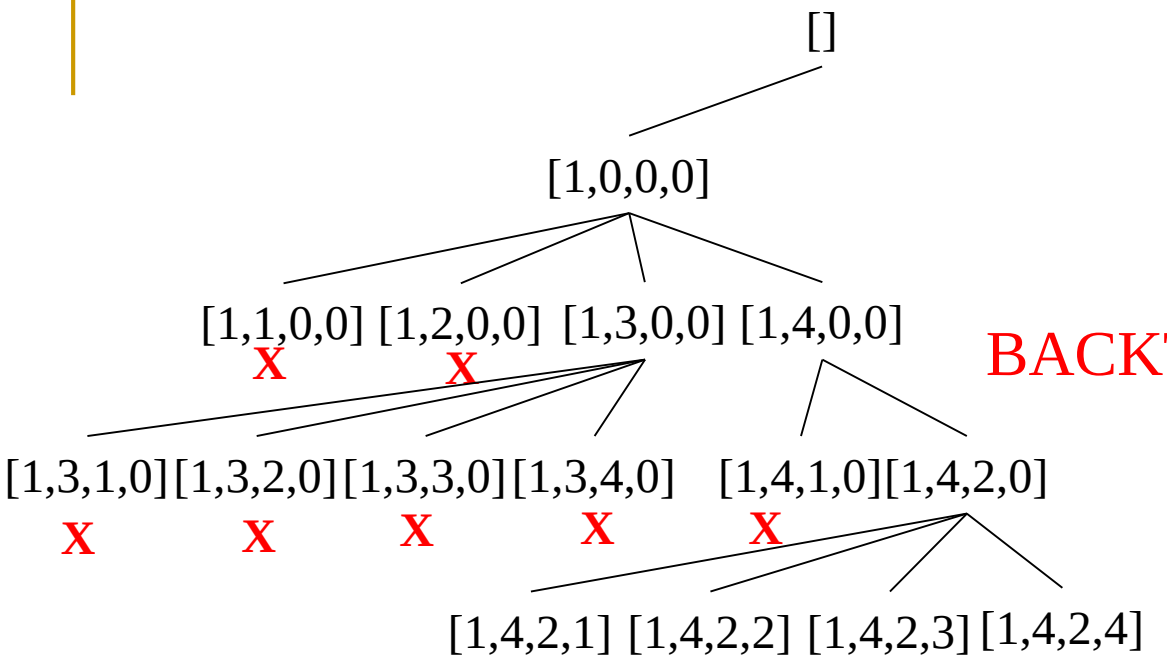

R
R
R
R


[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]

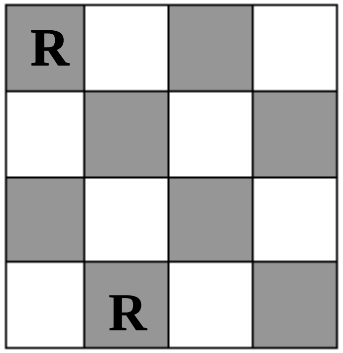

R
R
R
R

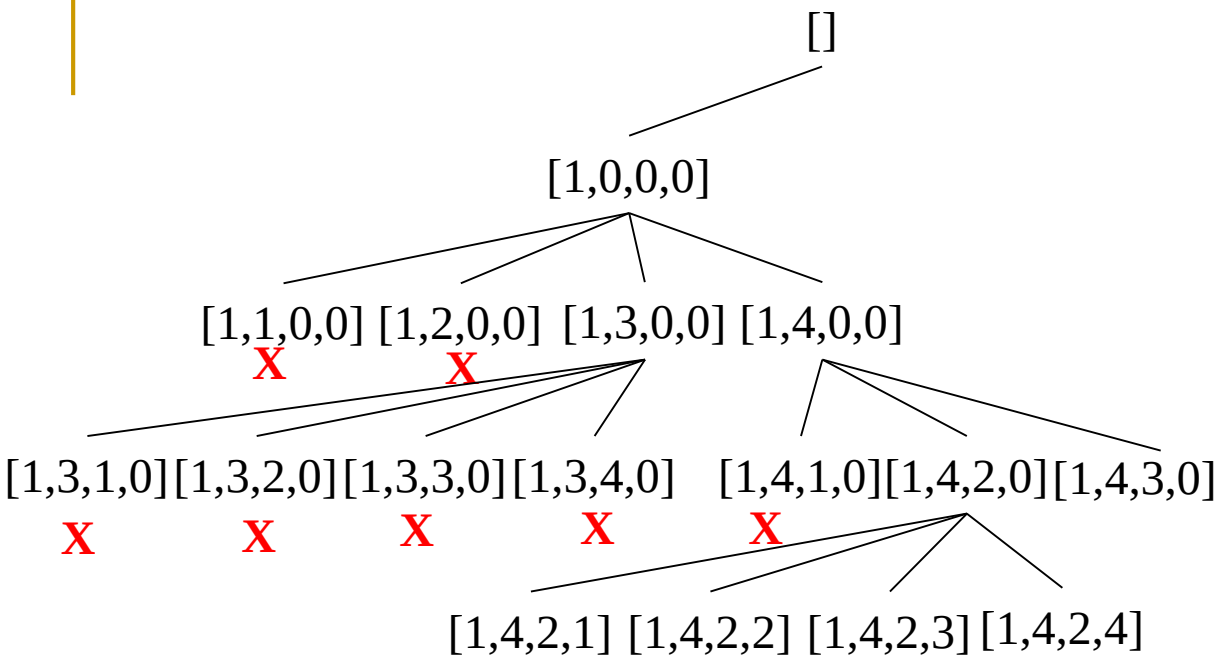

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

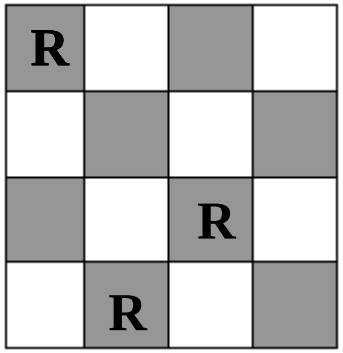

R
R
R
R

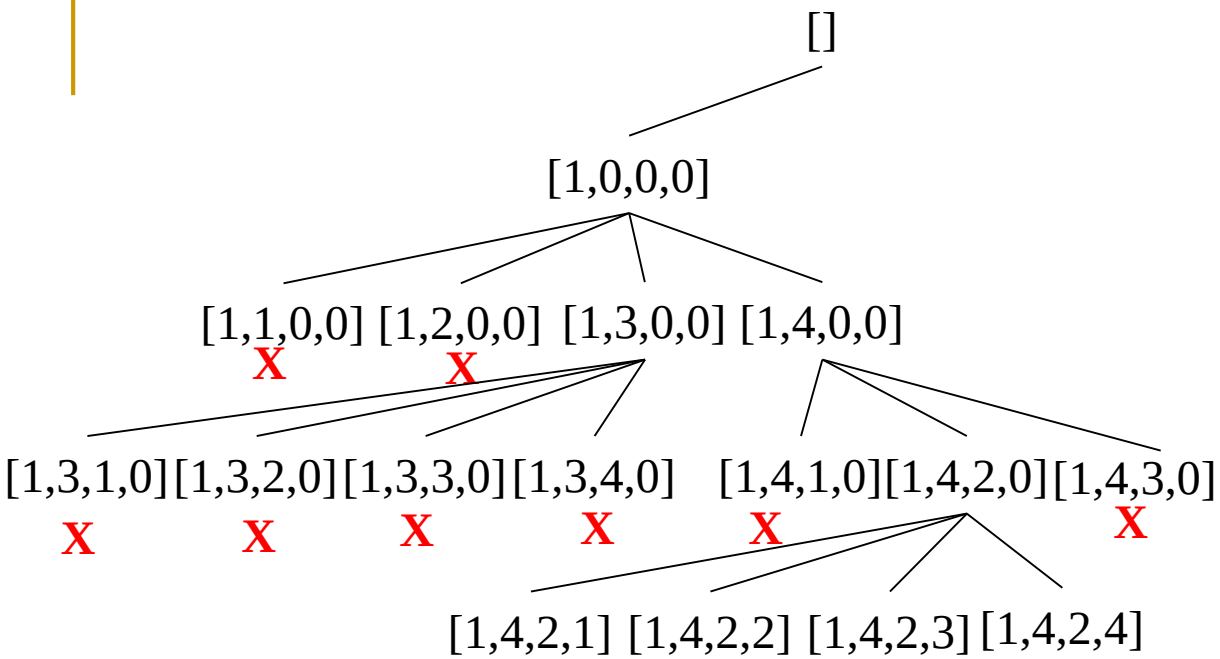


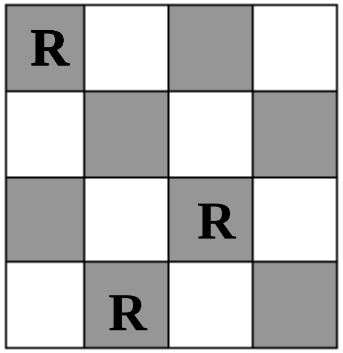
BACKTRACKING

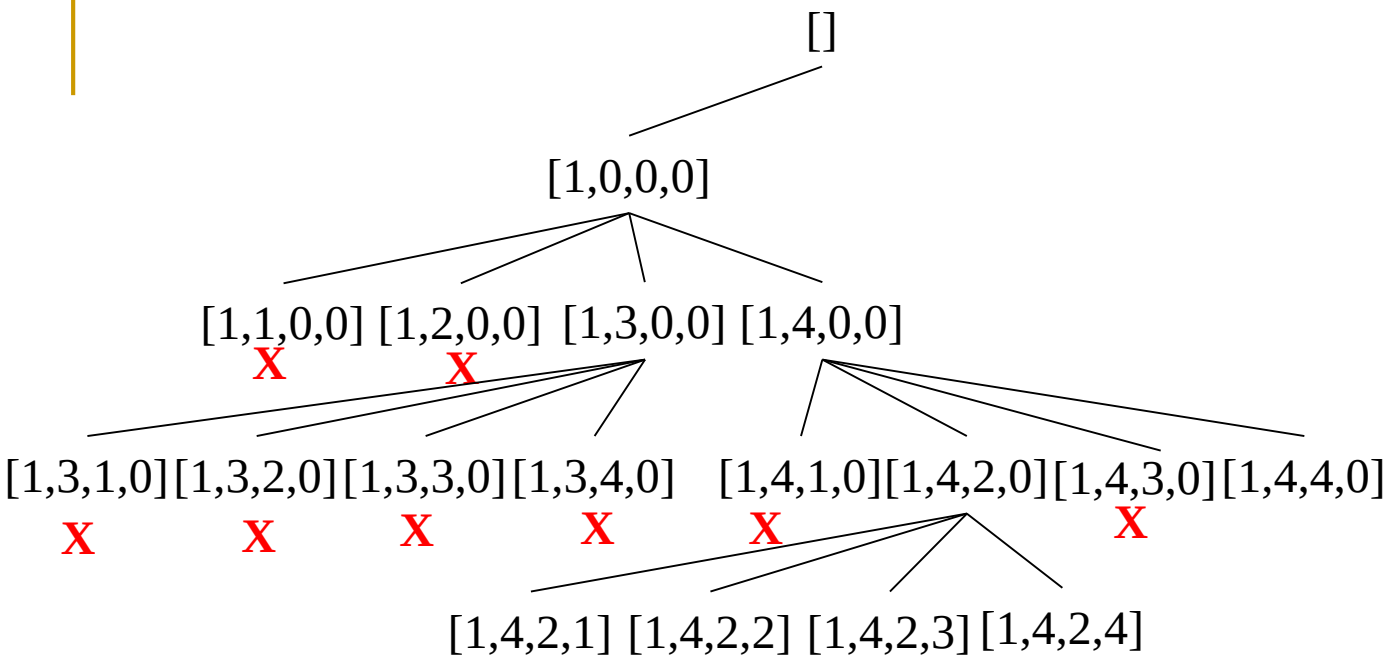


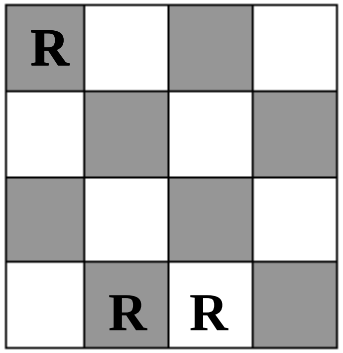

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

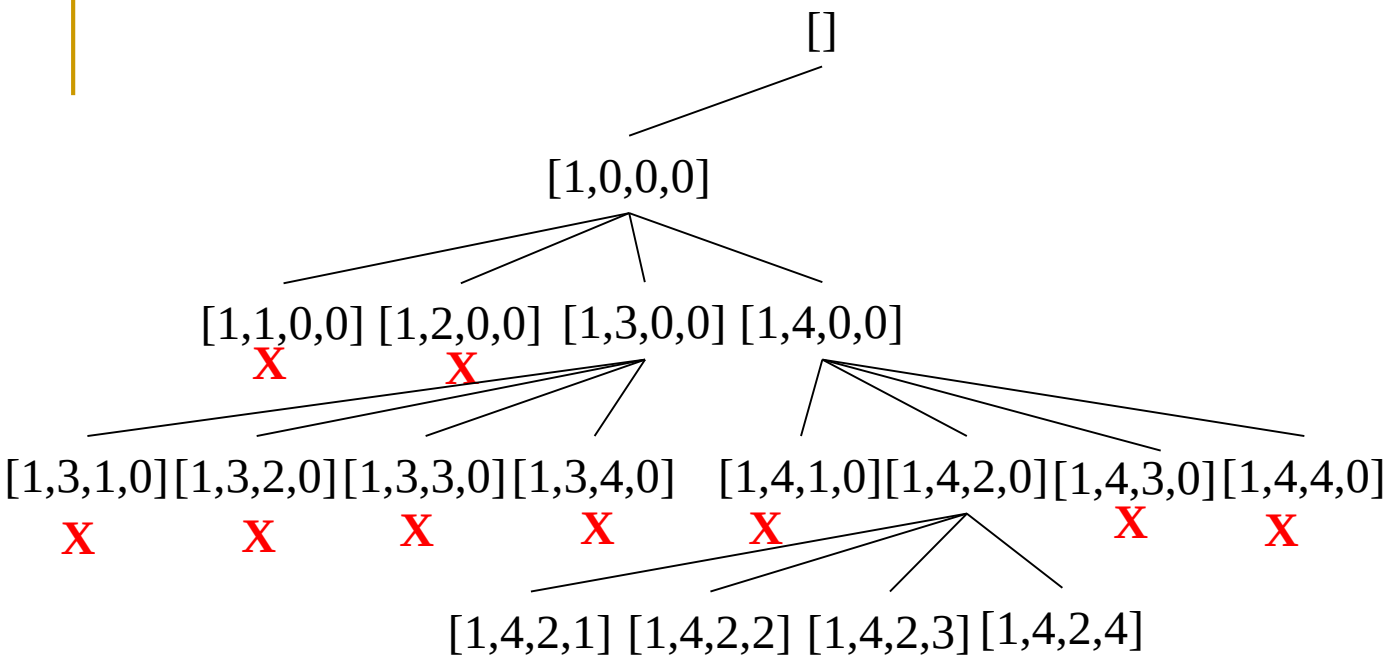

R
R
R

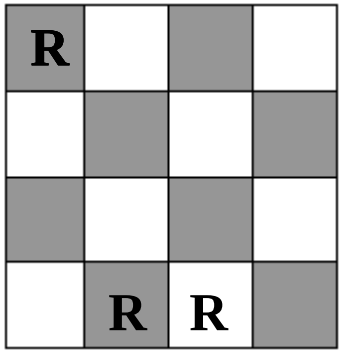

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

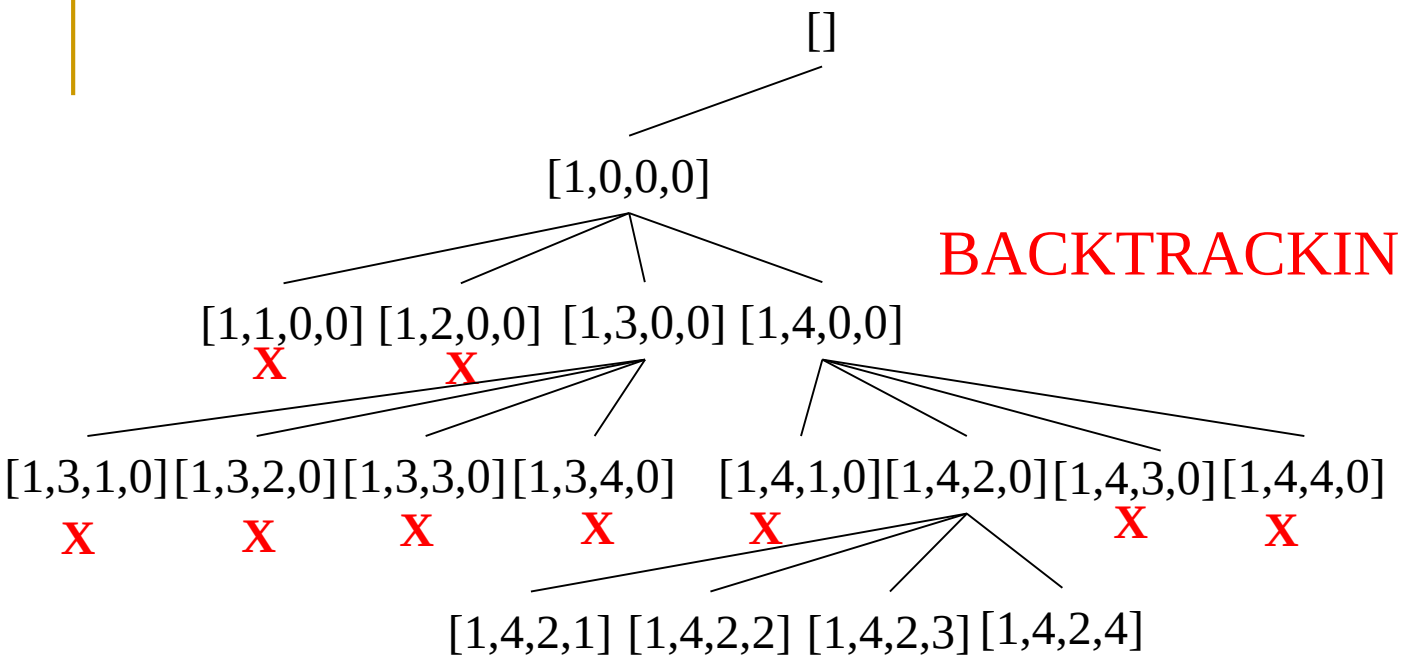

R
R
R


[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

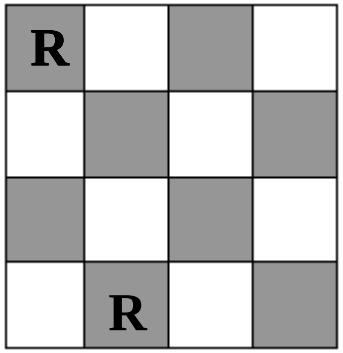

R
R
R

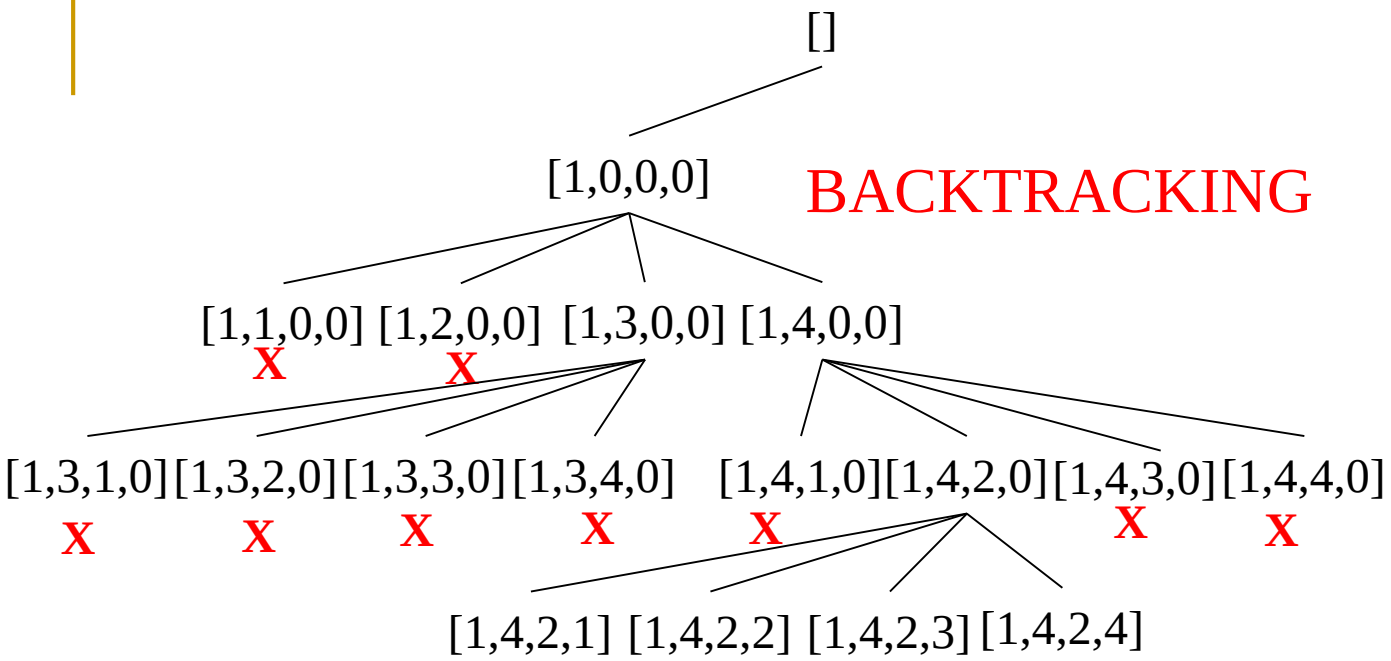

[]
[1,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

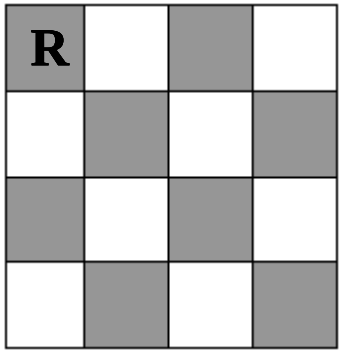

R
R
R



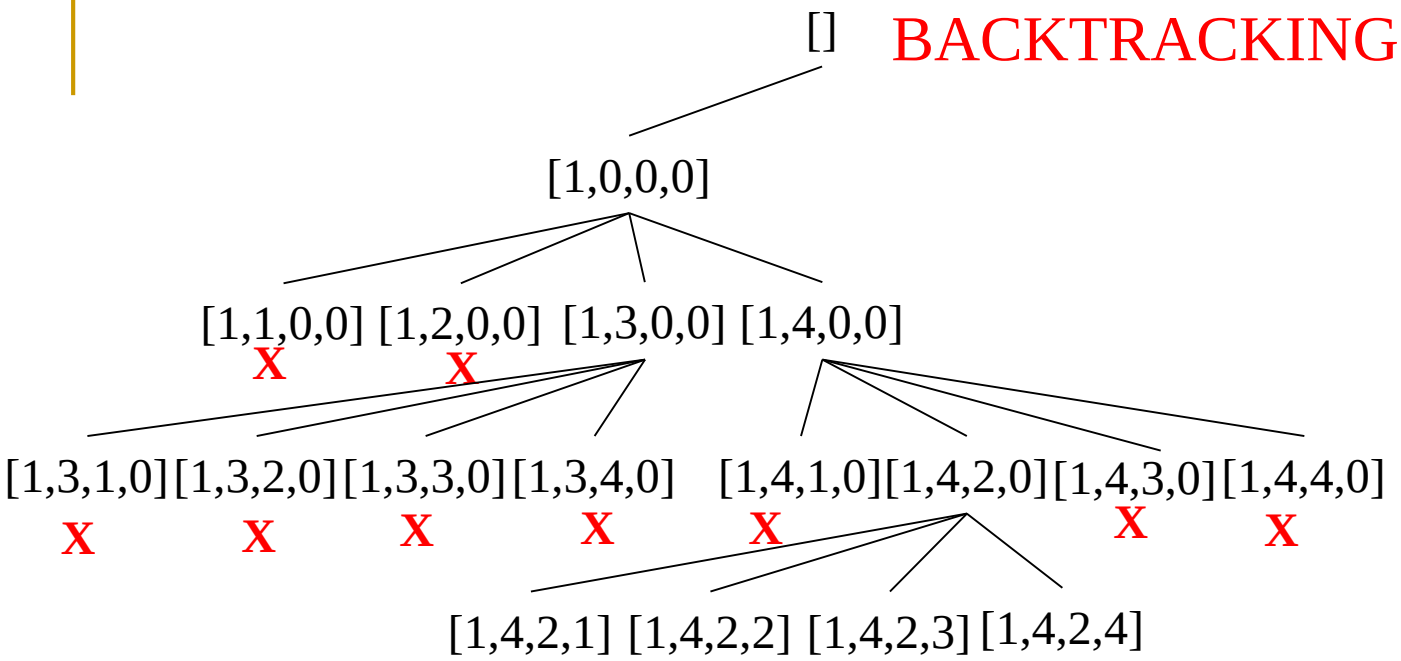
BACKTRACKING

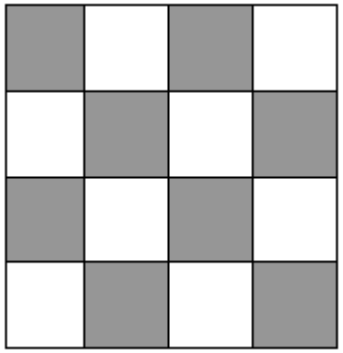


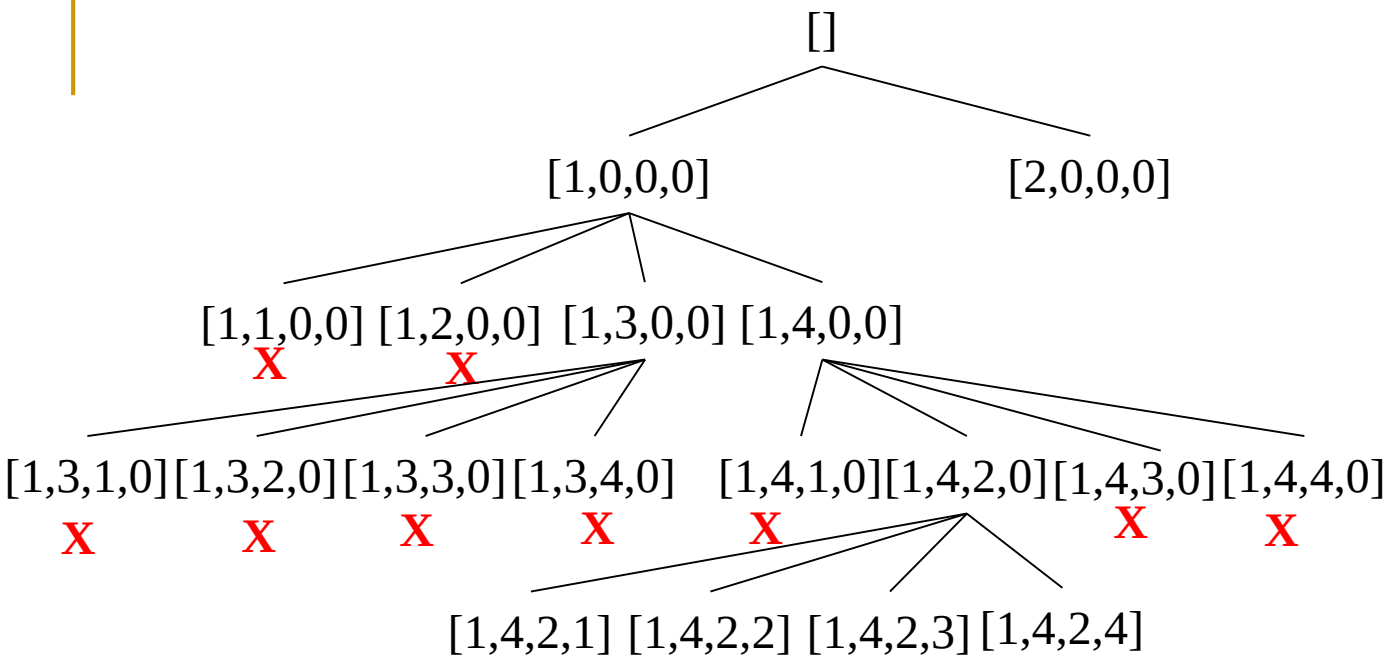

[]
[1,0,0,0]
BACKTRACKING
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

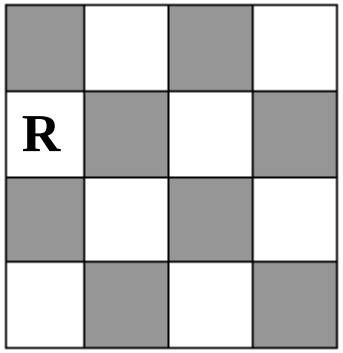

R

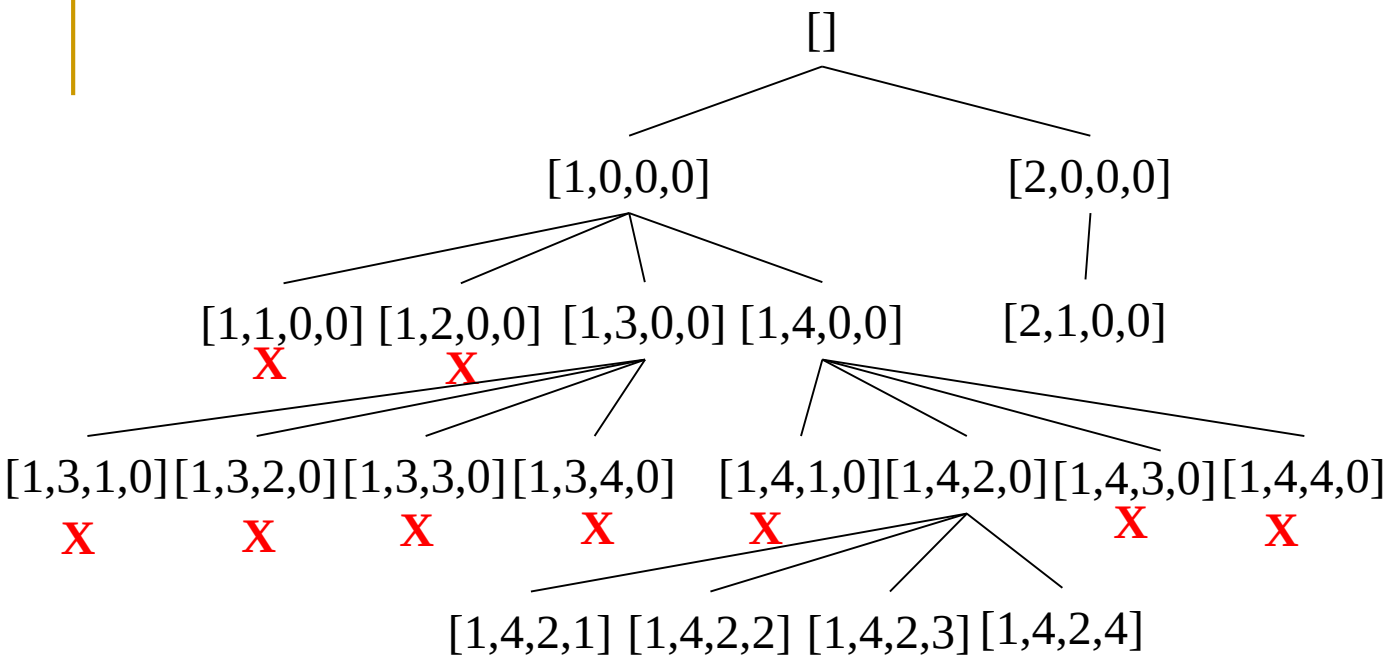
BACKTRACKING

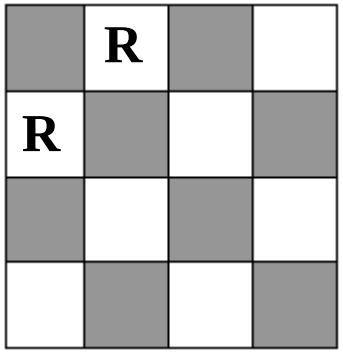


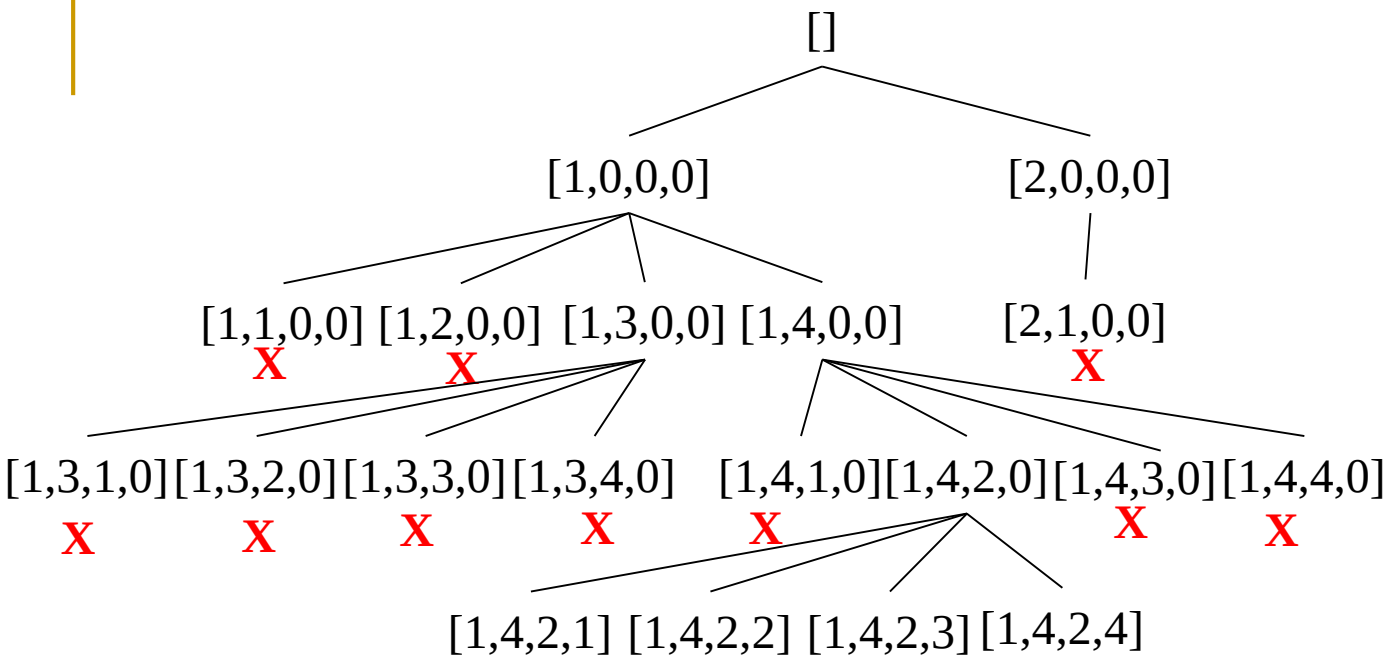


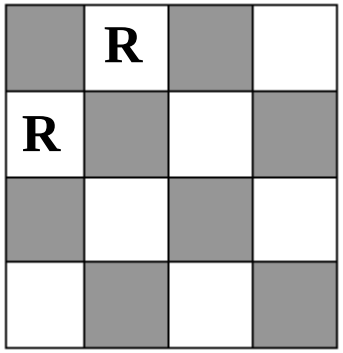

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

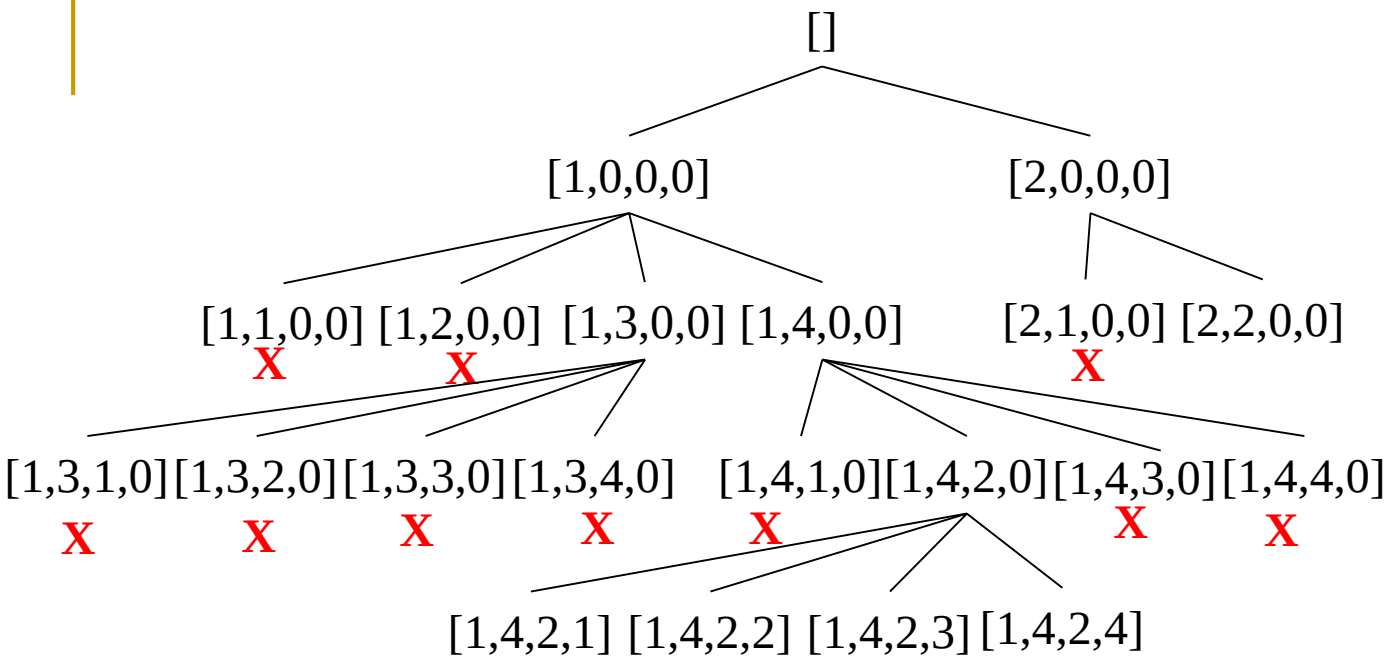

R

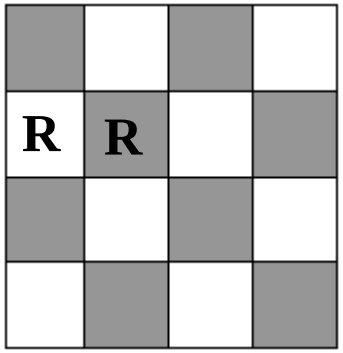

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

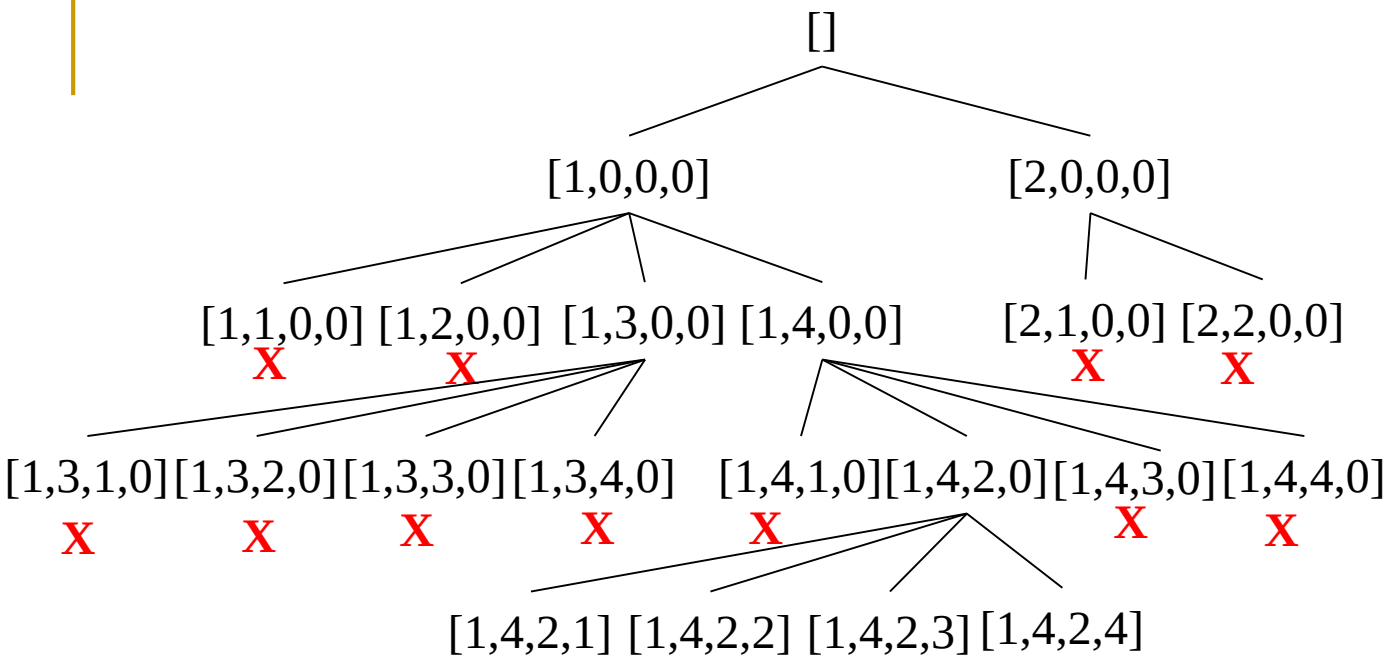

R
R

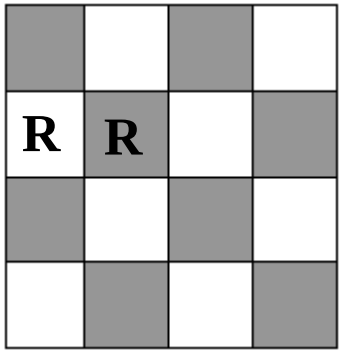

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
X
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

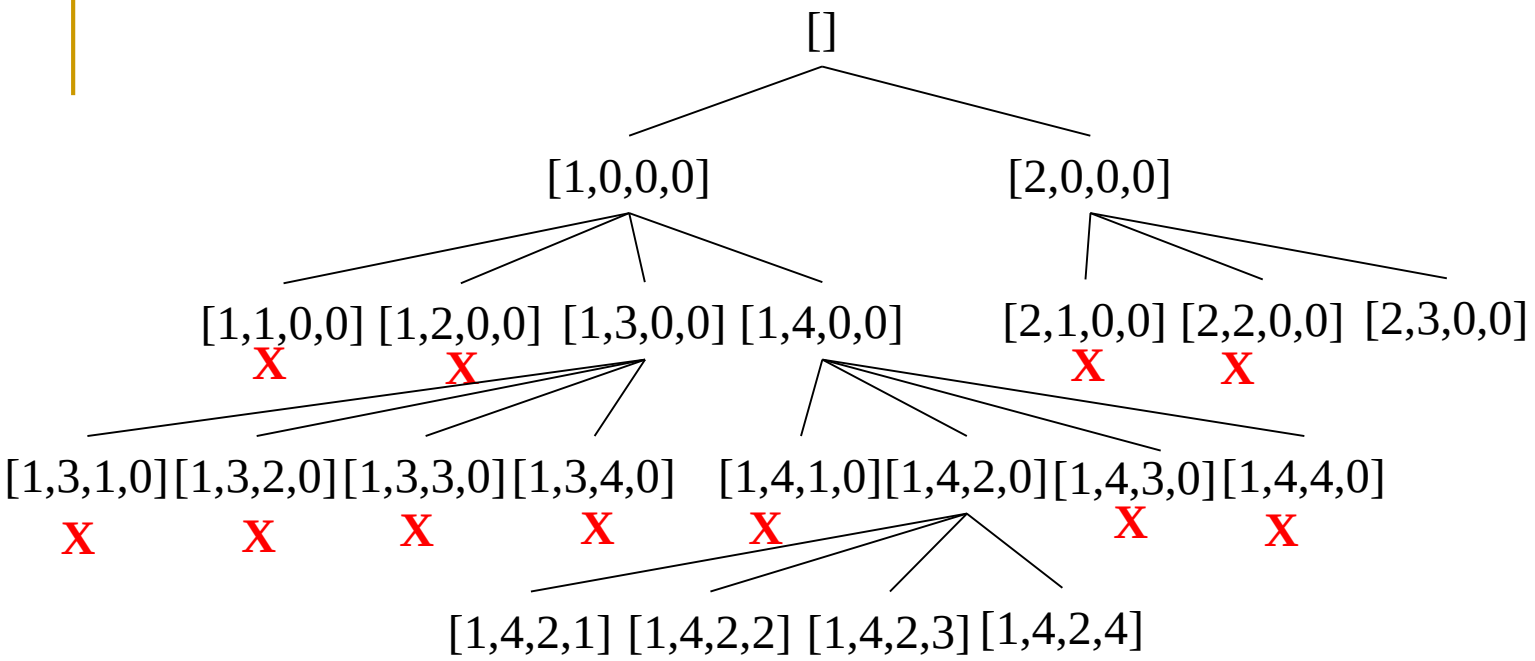

R
R

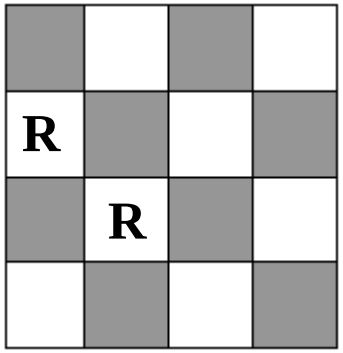

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
X
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

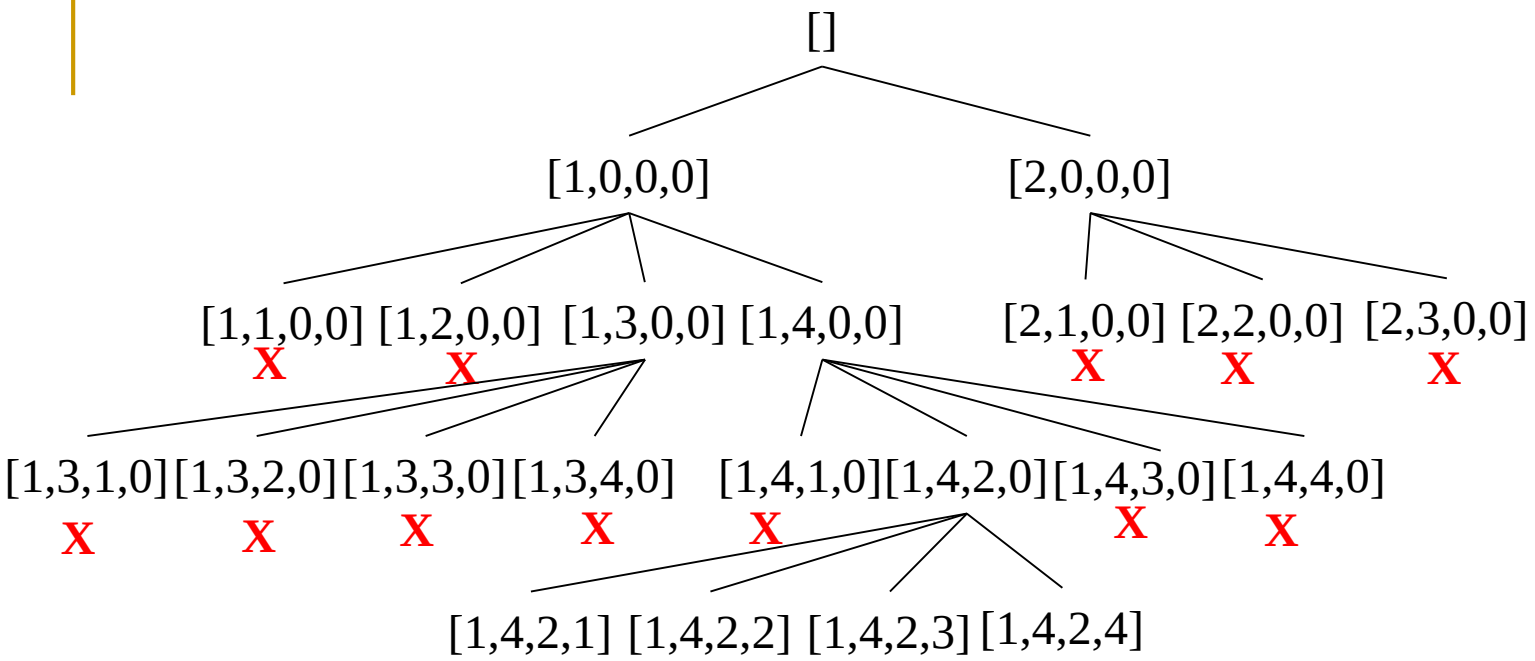

R
R

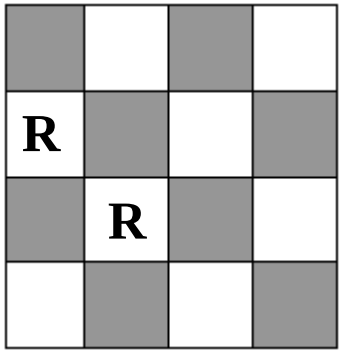

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
X
X
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

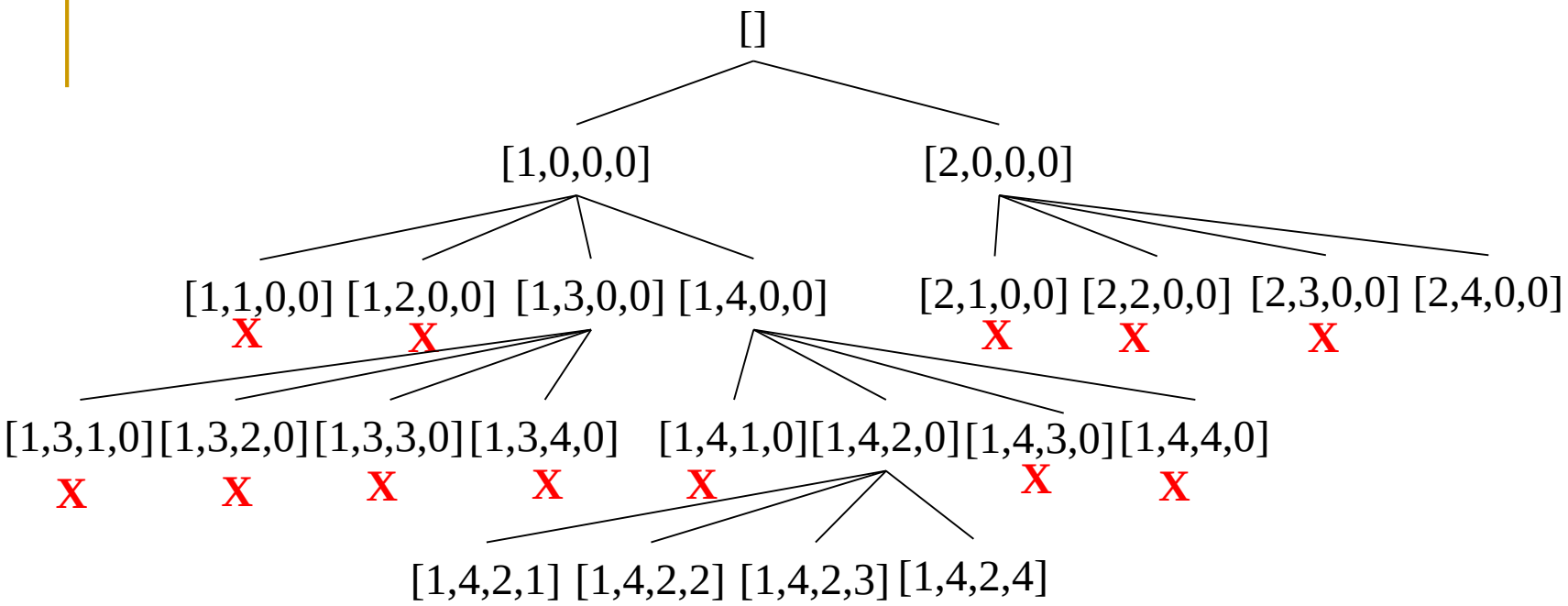

R
R

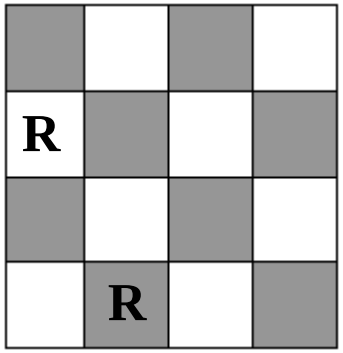

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
[2,3,0,0]
X
X
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

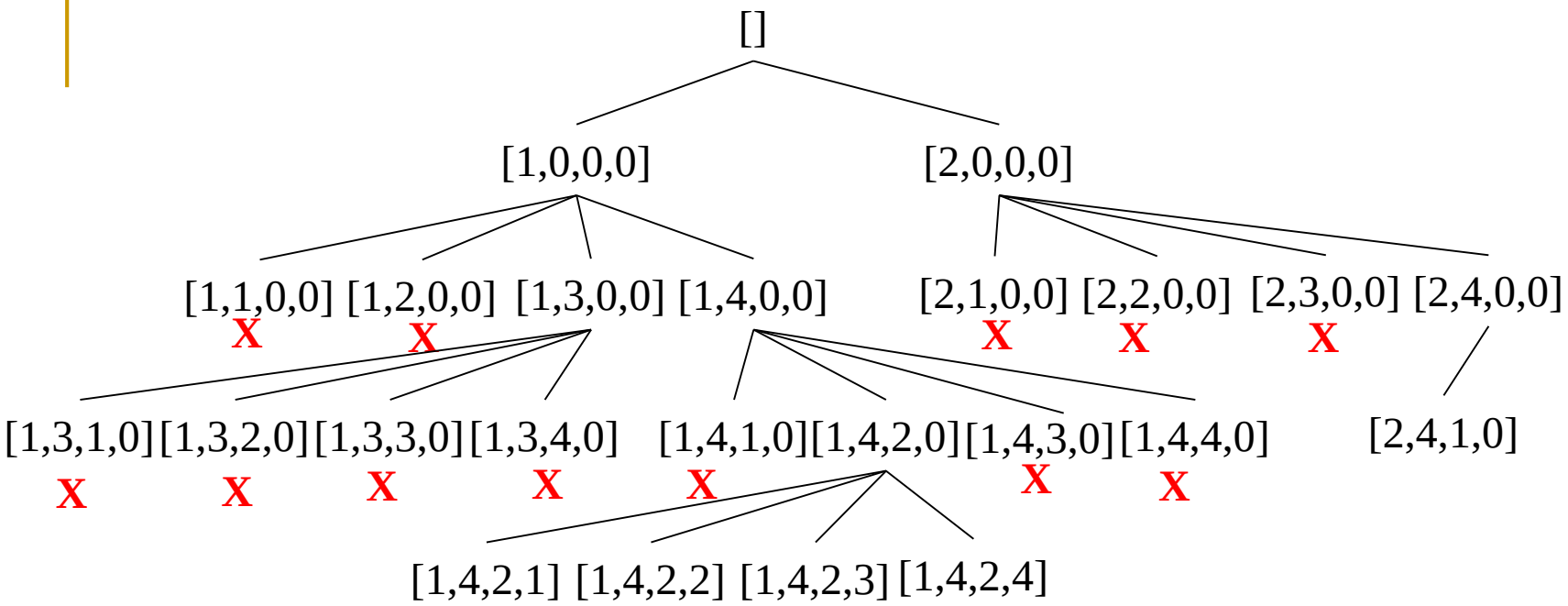

R
R

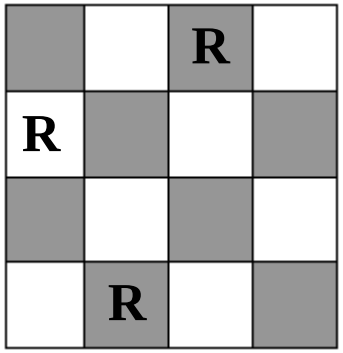

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
[2,3,0,0]
X
X
X
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

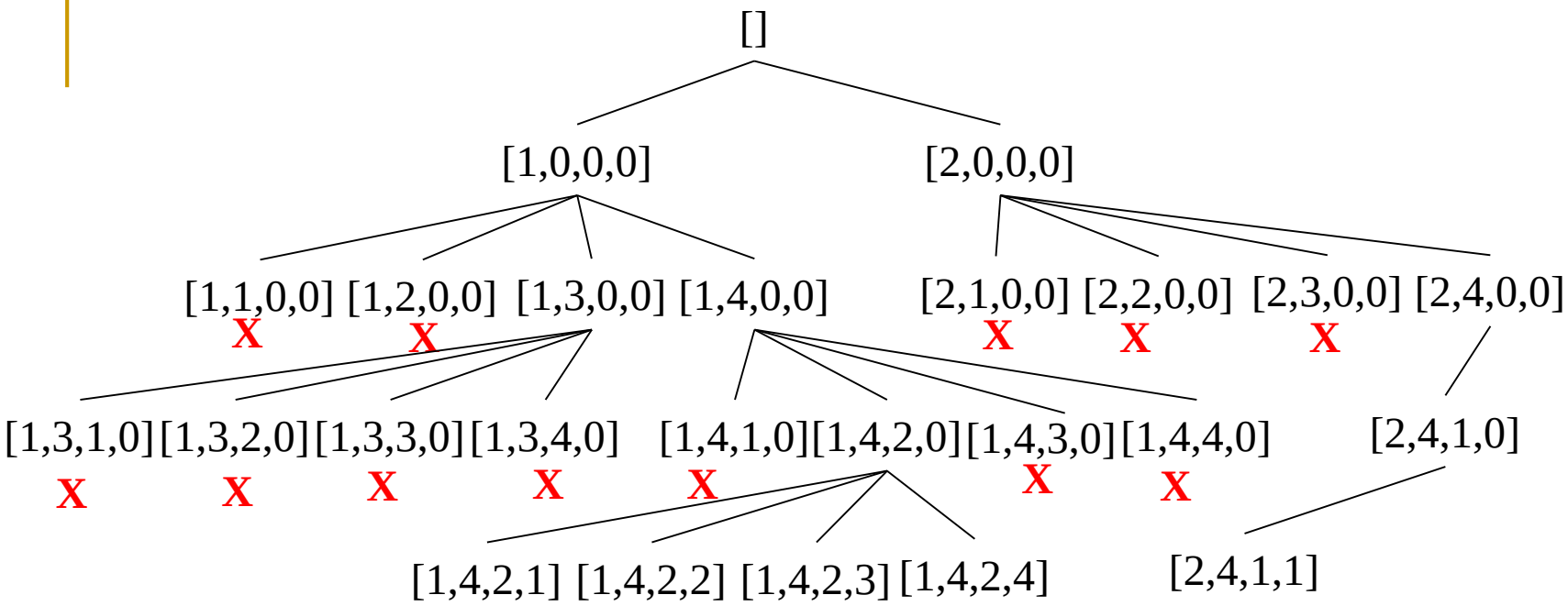

R
R

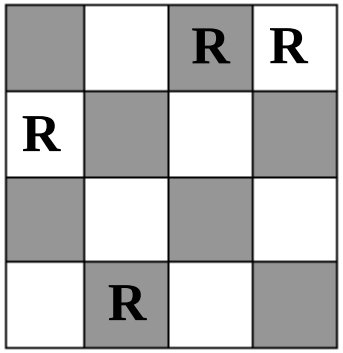

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
[2,3,0,0]
[2,4,0,0]
X
X
X
X
X
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
X
X
X
X
X
X
X
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

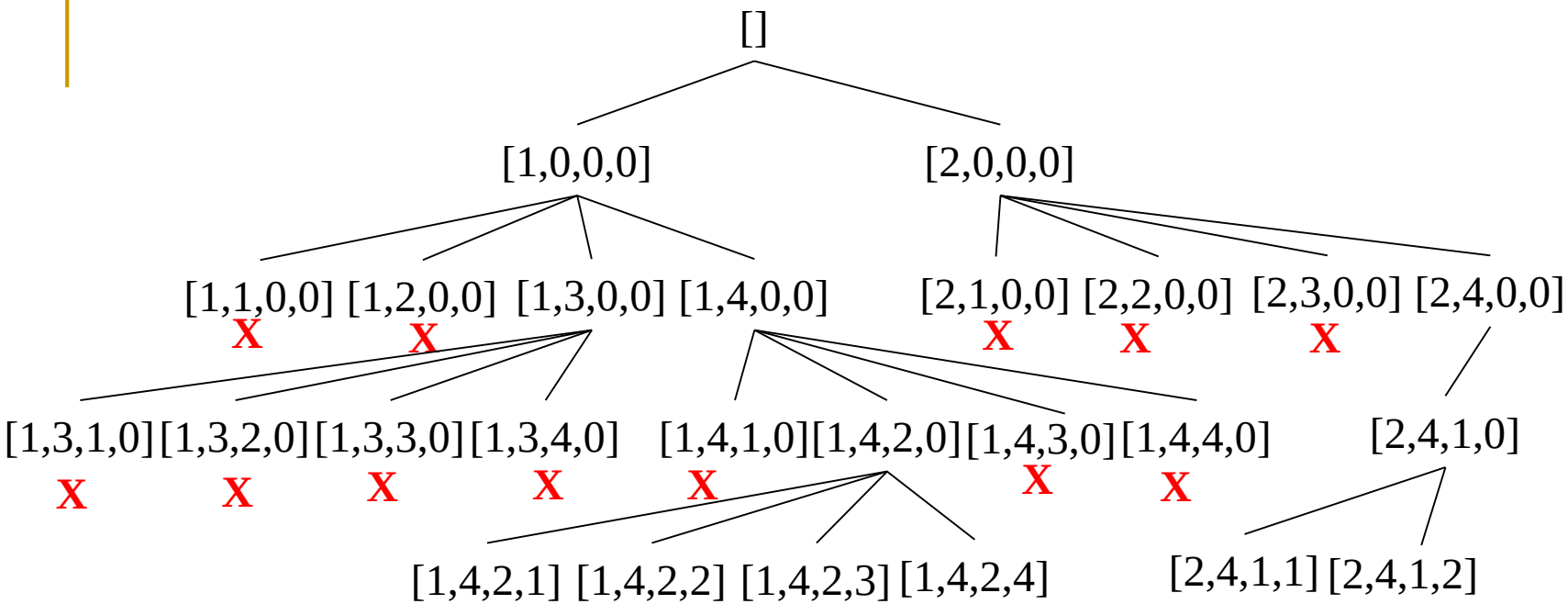

R
R

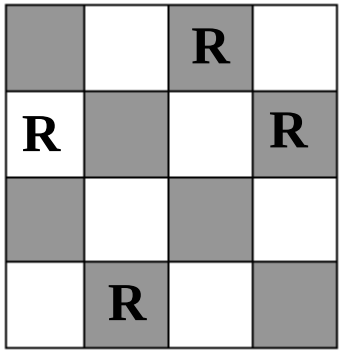

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
[2,3,0,0]
[2,4,0,0]
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
[2,4,1,0]
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]

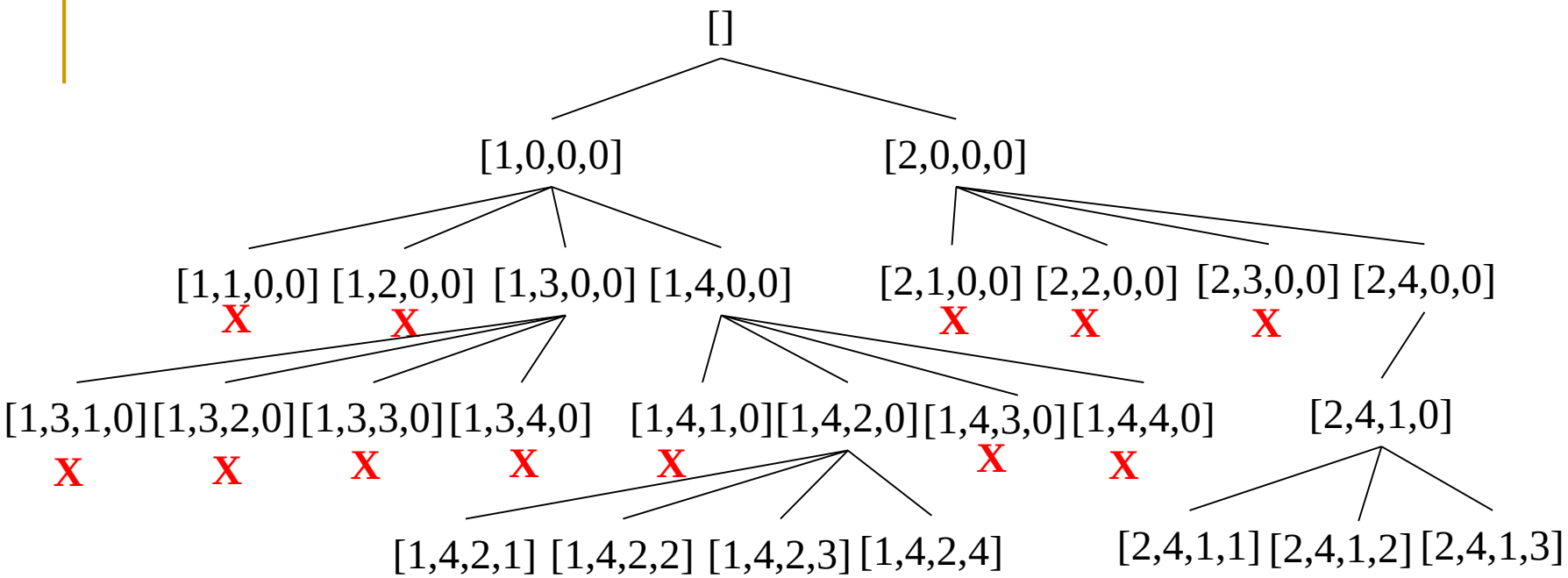

R
R
R

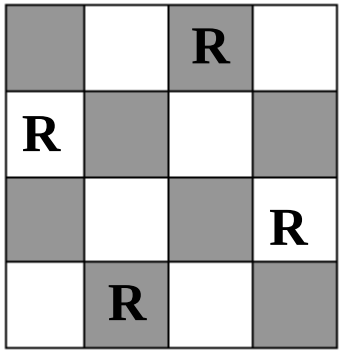

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
[2,3,0,0]
[2,4,0,0]
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
[2,4,1,0]
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]
[2,4,1,1]

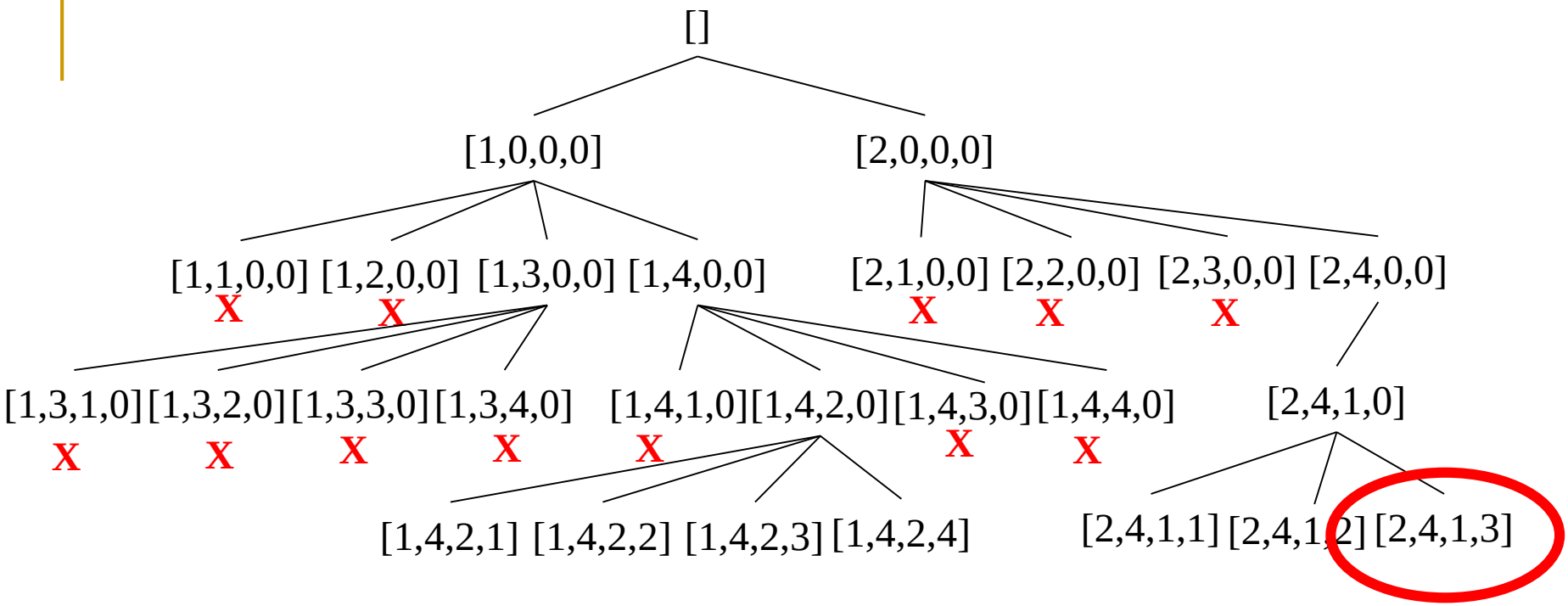

R
R
R
R

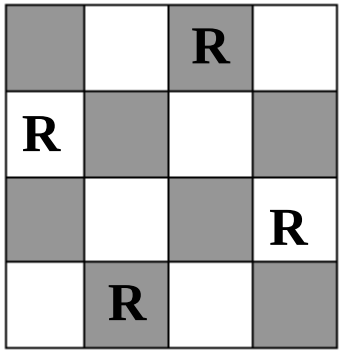

[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
[2,3,0,0]
[2,4,0,0]
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
[2,4,1,0]
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]
[2,4,1,1]
[2,4,1,2]

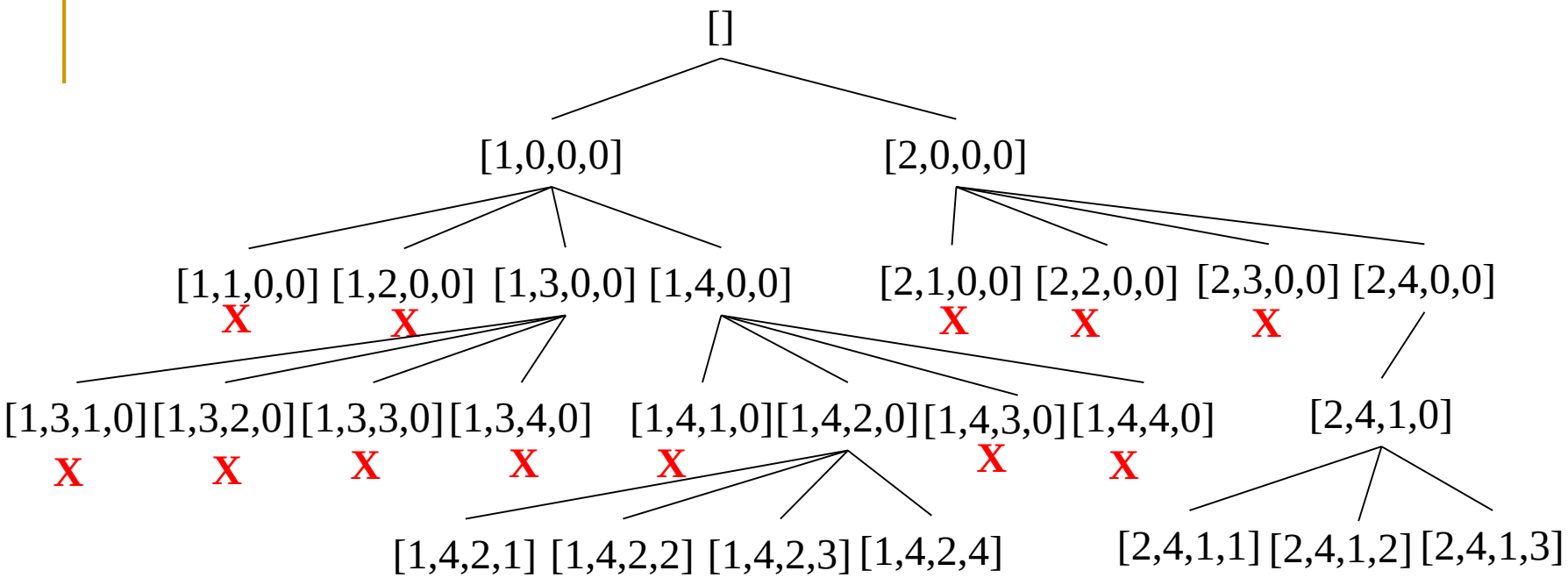

R
R
R
R


[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
[2,3,0,0]
[2,4,0,0]
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
[2,4,1,0]
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]
[2,4,1,1]
[2,4,1,2]
[2,4,1,3]

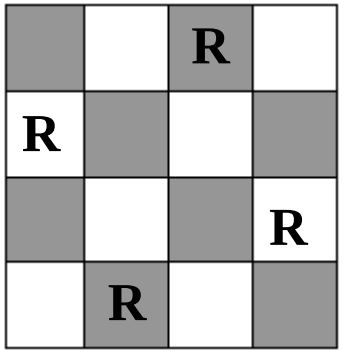

R
R
R
R


[]
[1,0,0,0]
[2,0,0,0]
[1,1,0,0]
[1,2,0,0]
[1,3,0,0]
[1,4,0,0]
[2,1,0,0]
[2,2,0,0]
[2,3,0,0]
[2,4,0,0]
X
[1,3,1,0]
[1,3,2,0]
[1,3,3,0]
[1,3,4,0]
[1,4,1,0]
[1,4,2,0]
[1,4,3,0]
[1,4,4,0]
[2,4,1,0]
[1,4,2,1]
[1,4,2,2]
[1,4,2,3]
[1,4,2,4]
[2,4,1,1]
[2,4,1,2]
[2,4,1,3]


R
R
R
R



Solução: [2,4,1,3]



Backtracking – Algoritmo Genérico

Algoritmo Backtrack(x):
Entrada: uma instância x de um problema difícil
Saída: uma solução para x ou "sem solução" se nenhuma existir
F ← {(X,0)}
Enquanto ! Vazio(F) **faça**
    Retire de F uma configuração (x,y) //poderia ter prioridade
    Expanda(x,y), fazendo um pequeno conjunto de escolhas adicionais
    Sejam (x1,y1), (x2,y2), …, (xk,yk) o conjunto de novas configurações
    Para cada nova configuração (xi, yi) **faça**
        Verifique consistência de (xi, yi)
        Se a verificação retorna "solução encontrada" **então**
            Retorne a solução (xi, yi)
        Se a verificação retornar "sem saída" **então**
            Descarte a configuração (xi,yi)
        Senão
            F ← F U {(xi, yi)}
FimEnquanto
Retorne "sem solução"

Goodrich, M. T., Tamasia, R. Projeto de Algoritmos: Fundamentos, Análise e Exemplos da Internet

Backtracking

- É eficiente para problemas de decisão, mas não foi planejado para problemas de otimização
- Ele pára quando encontra uma solução
- Podemos estende-lo para trabalhar com um problema de otimização
- Dessa modificação obtemos o padrão de algoritmo chamado de *branch-and-bound*

Bibliografia

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; (2002). Algoritmos –Teoria e Prática. Tradução da 2ª edição americana. Rio de Janeiro. Editora Campus
- TAMASSIA, ROBERTO; GOODRICH, MICHAEL T. (2004). Projeto de Algoritmos -Fundamentos, Análise e Exemplos da Internet
- ZIVIANI, N. (2007). Projeto e Algoritmos com implementações em Java e C++. São Paulo. Editora Thomson