Problem Solving and Search in Artificial Intelligence

Uninformed Search Strategies

Nysret Musliu
Database and Artificial Intelligence Group
Institut für Informationssysteme, TU-Wien

Introduction

- Many classic algorithms are designed to search spaces for an optimum solution
- Broadly, classic methods fall into two classes:
  - Algorithms that only evaluate complete solutions
    - Exhaustive search, local search, ...
  - Algorithms that require the evaluation of partially constructed or approximate solutions
    - Branch and bound strategies, ...

Exhaustive Search

- Checks every solution in the search space until the best global solution has been found
- Can be used only for small instances of problems
- Exhaustive (enumerative) algorithms are simple
- Search space can be reduced by backtracking search
- Some optimization methods, e.g., branch and bound and A^* are based on an exhaustive search

Exhaustive Search

- How can we generate a sequence of every possible solution to the problem?
  - The order in which the solution are generated is irrelevant
  - Depends from selected representation

Enumerating the SAT

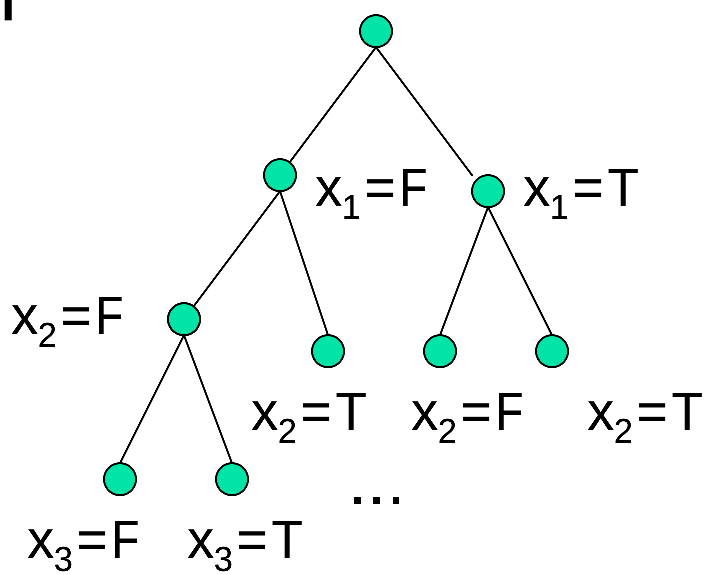
- We have to generate every possible binary string of length n
- All solutions correspond to whole numbers in a one-to-one mapping
- Generate all non-negative integers from 0 to 2^n-1 and convert each of these integers into the matching binary string of length n

0000 0

0001 1	0100 4	0111 7	1010 10	1101 13
0010 2	0101 5	1000 8	1011 11	1110 14
0011 3	0110 6	1001 9	1010 12	1111 15

Enumerating the SAT

- Tree representations of search space for SAT



Enumerating TSP

- How to generate all possible permutations?
- If some cities are not connected, some permutation might not be feasible

Generation of Permutations

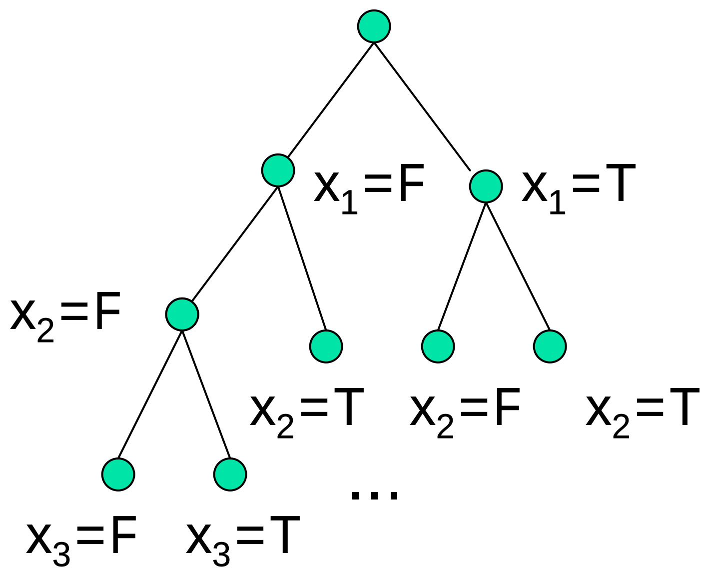
```
Procedure gen1-permutation(i)
Begin
   k=k+1
   P[i]=k
   if k=n then
        for q=1 to n do
               print P[q]
   for q=1 to n do
        if P[q]=0 then gen1-permutation(q)
   k=k-1
   P[i]=0
end
```

Uninformed search strategies

- Breadth first search
- Depth first search
- Depth limited search
- Iterative deepening search

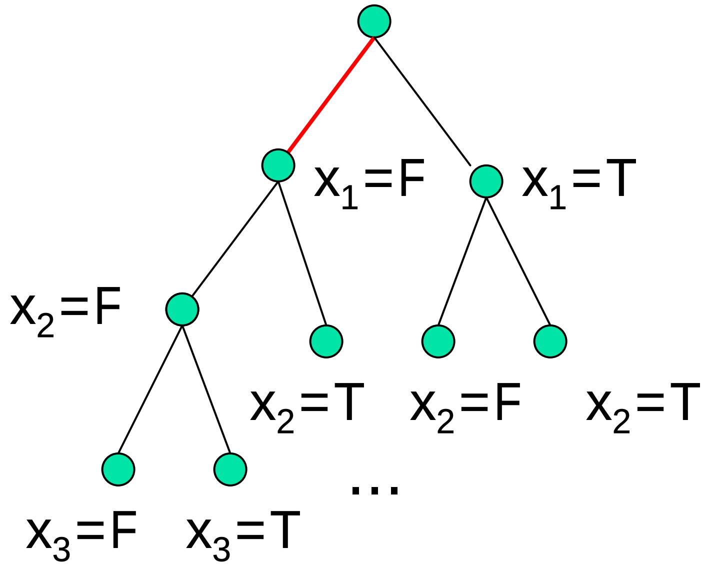
Breadth First Search

- The shallowest unexpanded node is first expanded



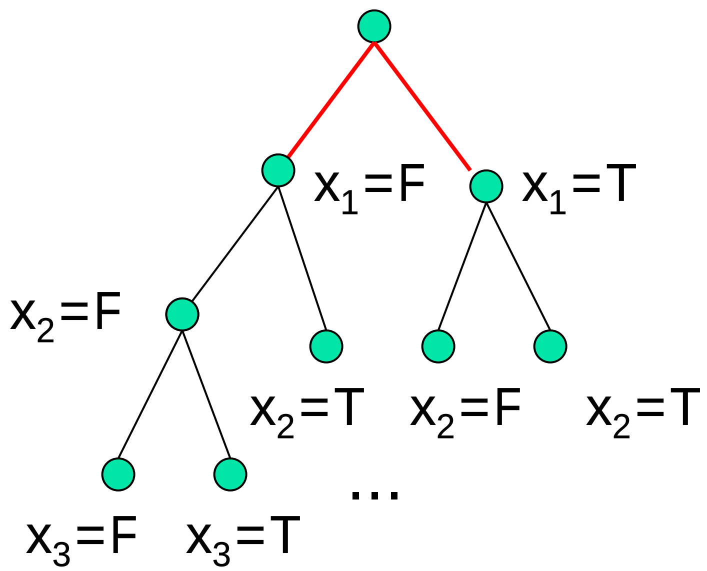
Breadth First Search

- The shallowest unexpanded node is first expanded



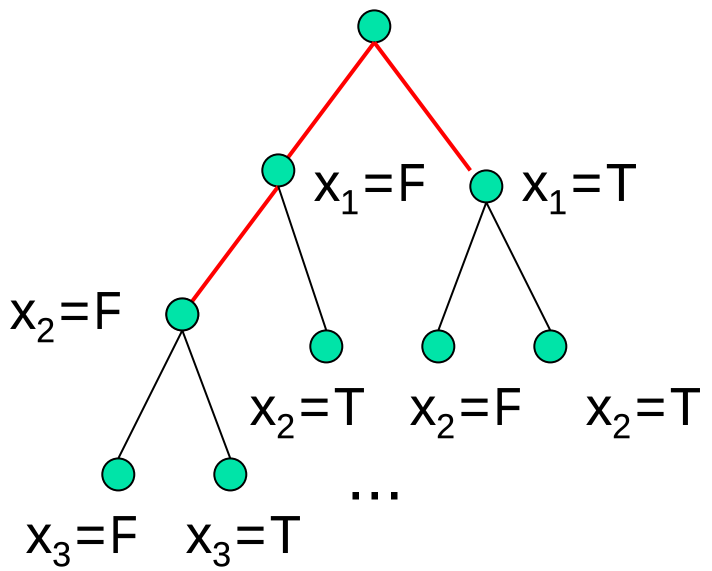
Breadth First Search

- The shallowest unexpanded node is first expanded

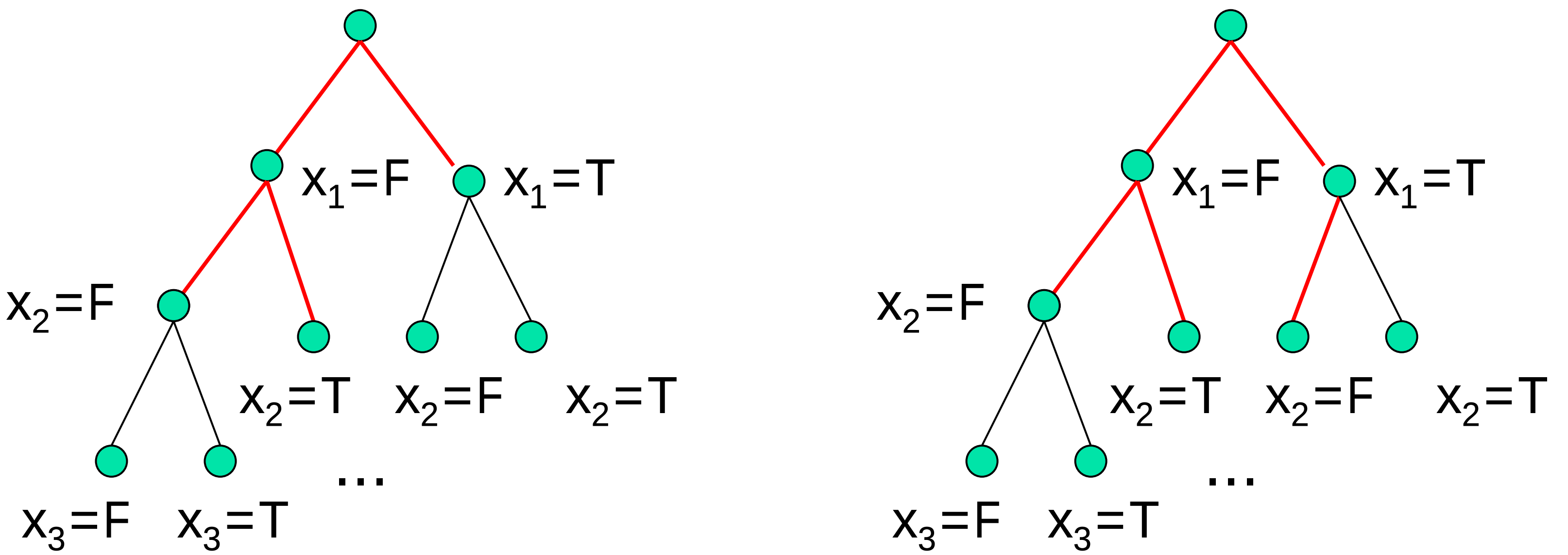


Breadth First Search

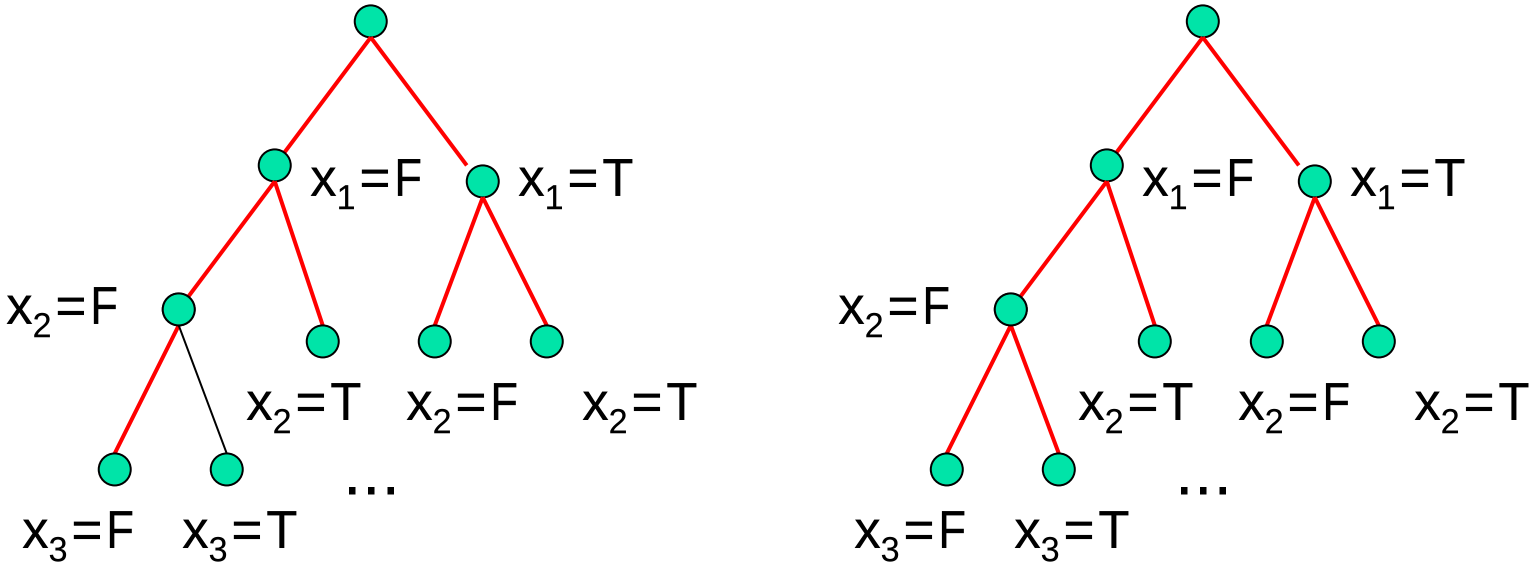
- The shallowest unexpanded node is first expanded



Breadth First Search



Breadth First Search



Depth First Search

Procedure depth-first(v)

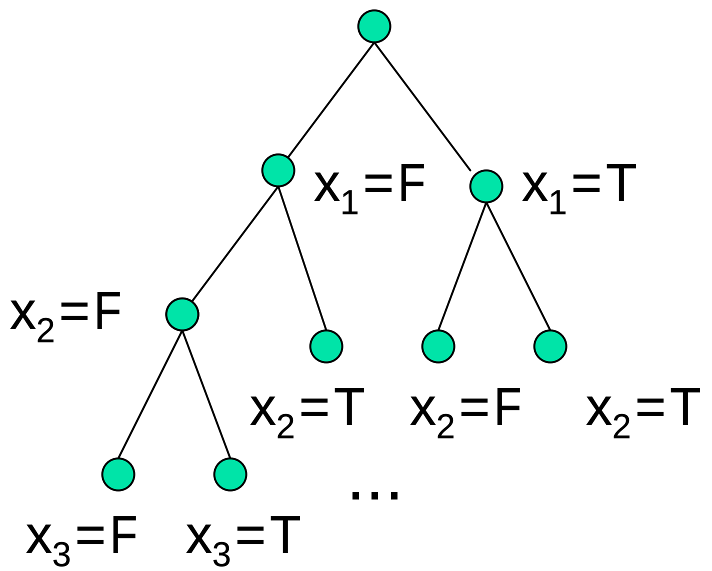
Begin

visit v

for each child w of v **do**

depth-first(w)

end



Depth First Search

Procedure depth-first(v)

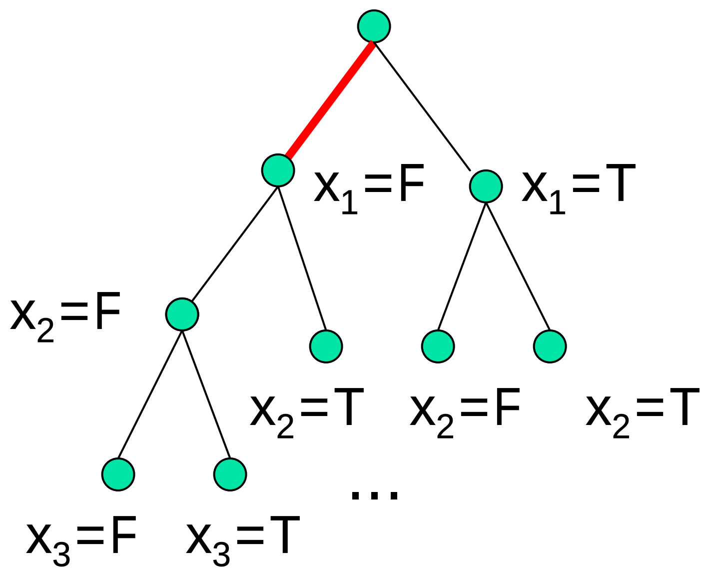
Begin

visit v

for each child w of v **do**

depth-first(w)

end



Depth First Search

Procedure depth-first(v)

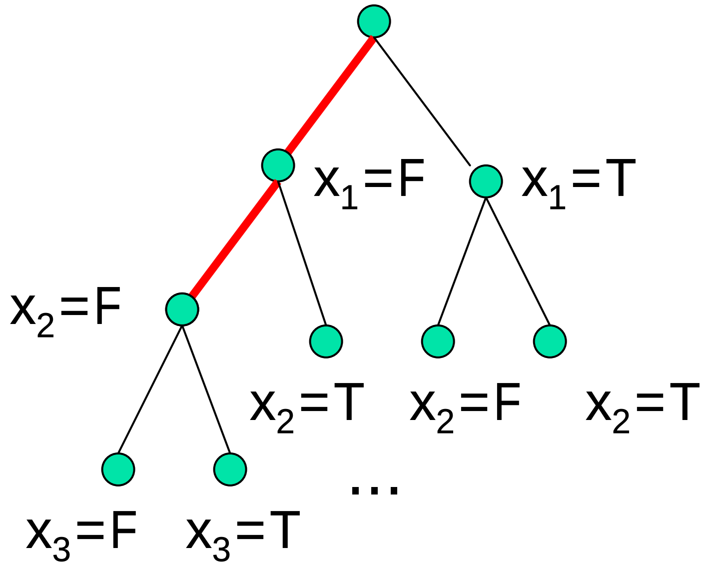
Begin

visit v

for each child w of v **do**

depth-first(w)

end



Depth First Search

Procedure depth-first(v)

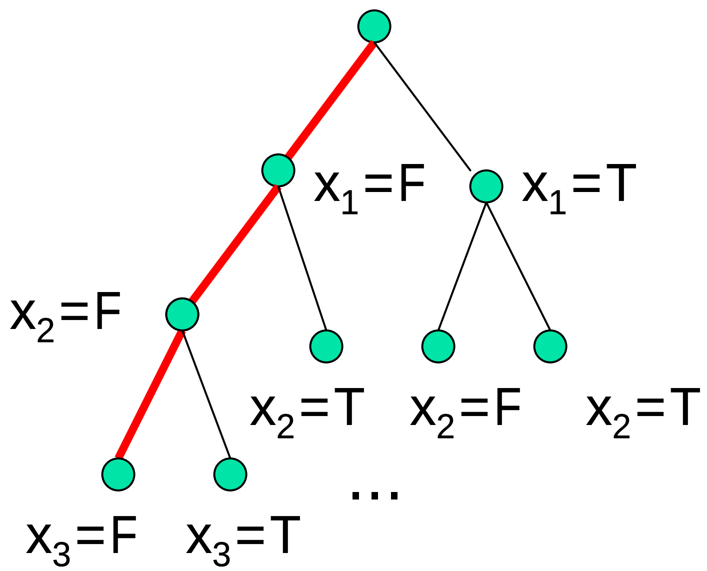
Begin

visit v

for each child w of v **do**

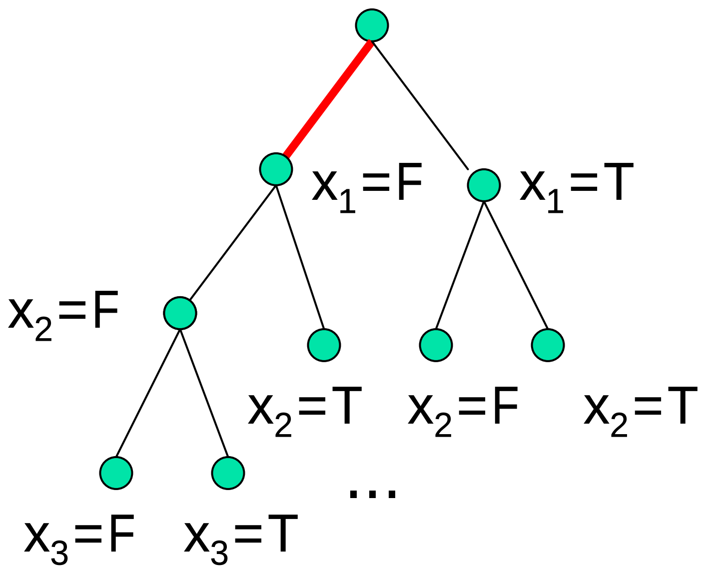
depth-first(w)

end



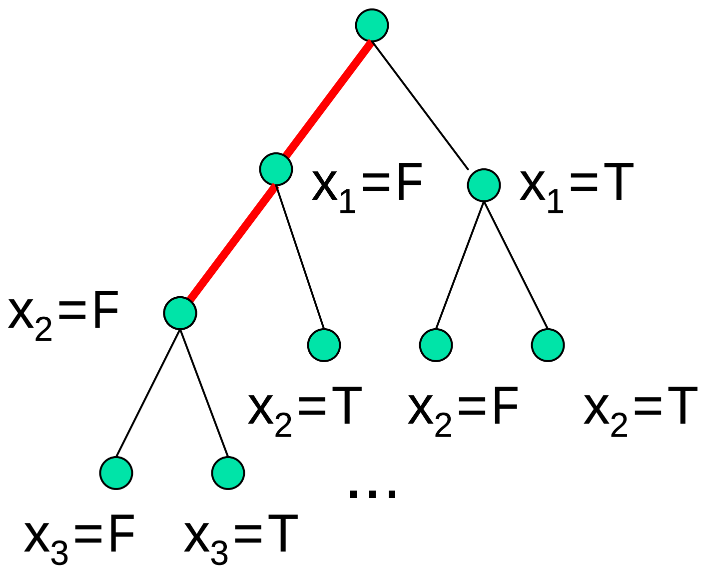
Backtracking

- Suppose the SAT formula contains a clause: ...$(x_1 \vee x_2)$...



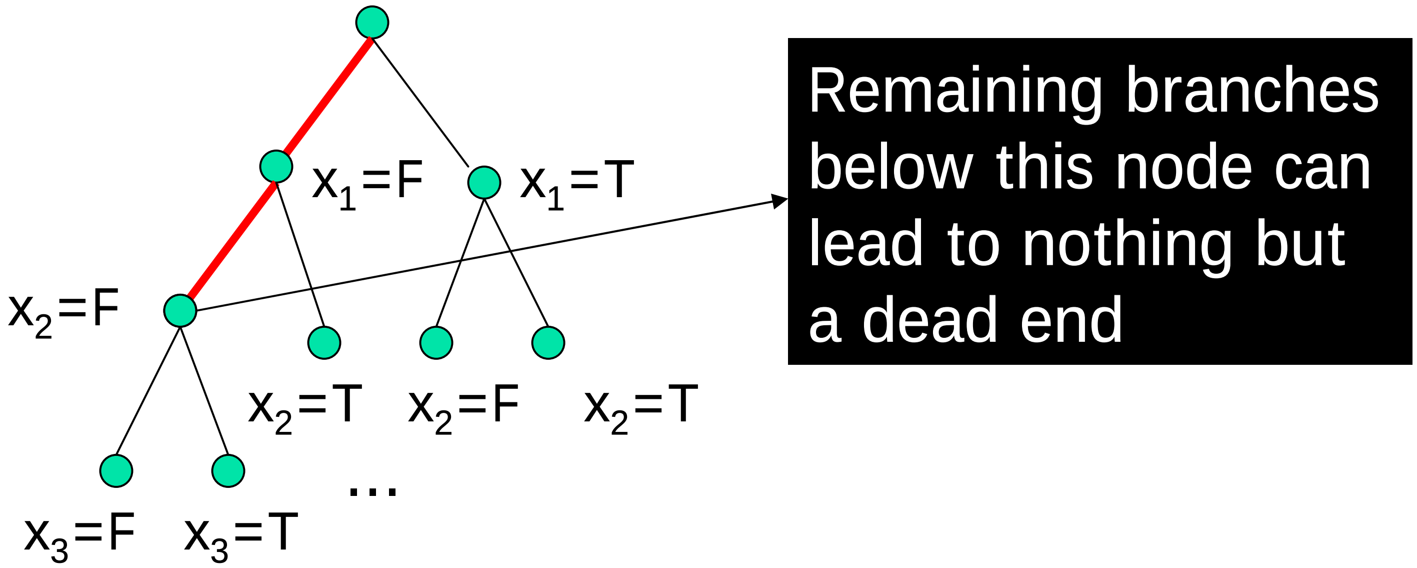
Backtracking

- Suppose the SAT formula contains a clause: ...$(x_1 \vee x_2)$...



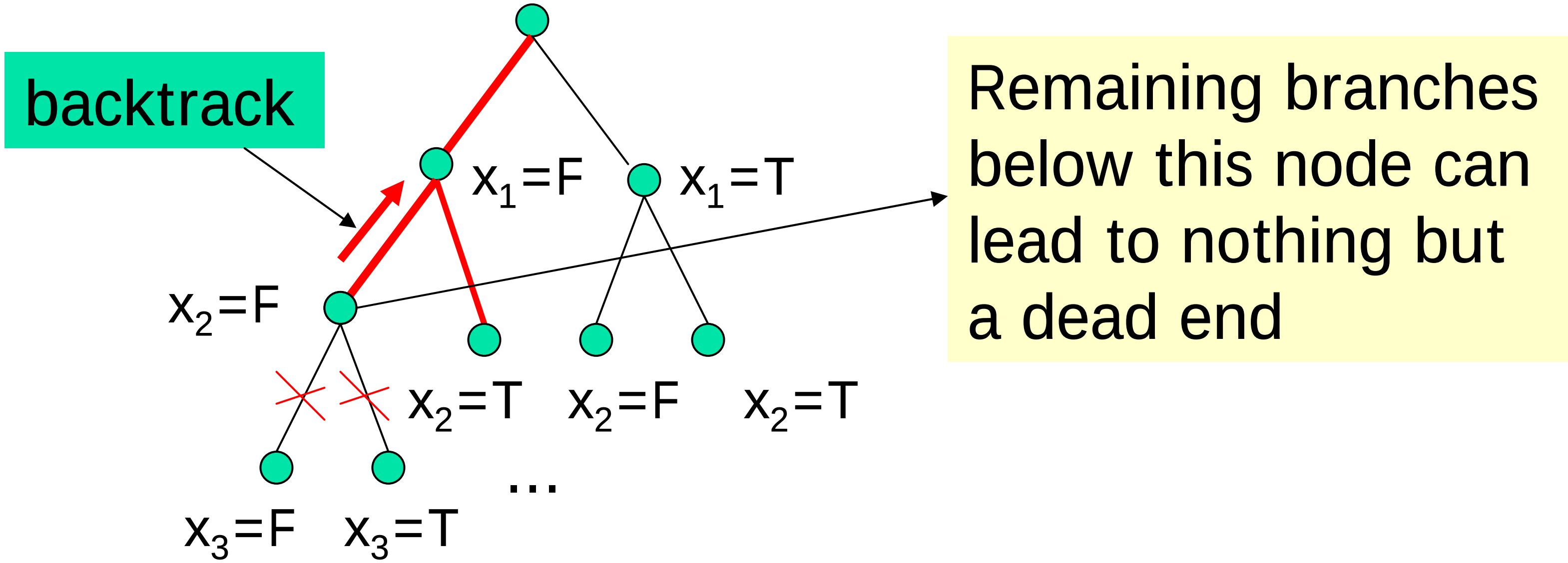
Backtracking

- Suppose the SAT formula contains a clause: ...(x_1v x_2)...



Backtracking

- Suppose the SAT formula contains a clause: ...$(x_1 \vee x_2)$...



Search algorithms performance

- Completeness
  - Does the algorithm finds a solution if that exists
- Optimality
  - Does the algorithm finds an optimal solution
- Time Complexity
  - Number of expanded nodes
- Space Complexity
  - Memory needed to perform the search

Search algorithms performance

b : branching factor (maximum number of successors)

d : the depth of the shallowest goal node in the search tree

m : maximum depth of the tree

Breadth first search performance

- Complete if b is finite
- Optimal if the cost of each step is 1
- Time:

$$1 + b + b^2 + \ldots + b^d = (b^{d+1}-1)/(b-1) = O(b^d)$$

- Space:

$O(b^d)$

Depth first search performance

- Complete if the search tree if finite
- Not optimal
- Time:

  $1 + b + b^2 + \ldots + b^m = O(b^m)$
- Space:

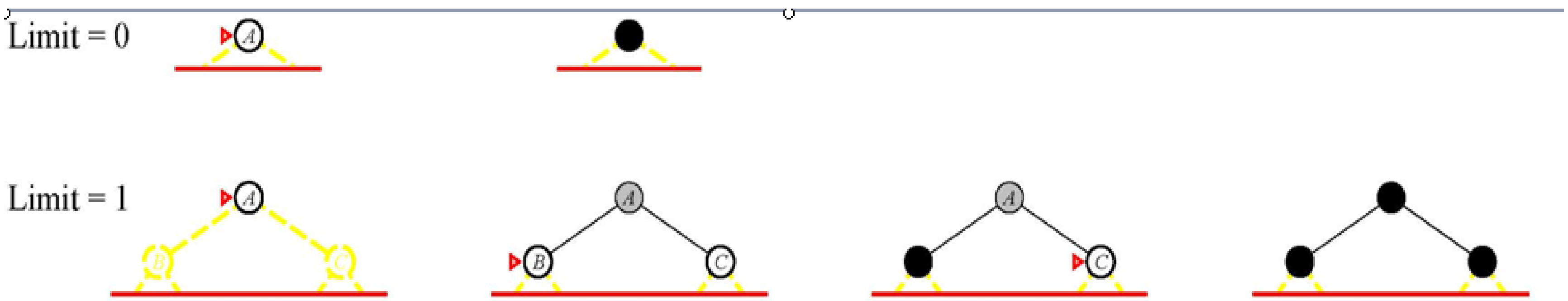
  $O(bm)$

Depth-limited search

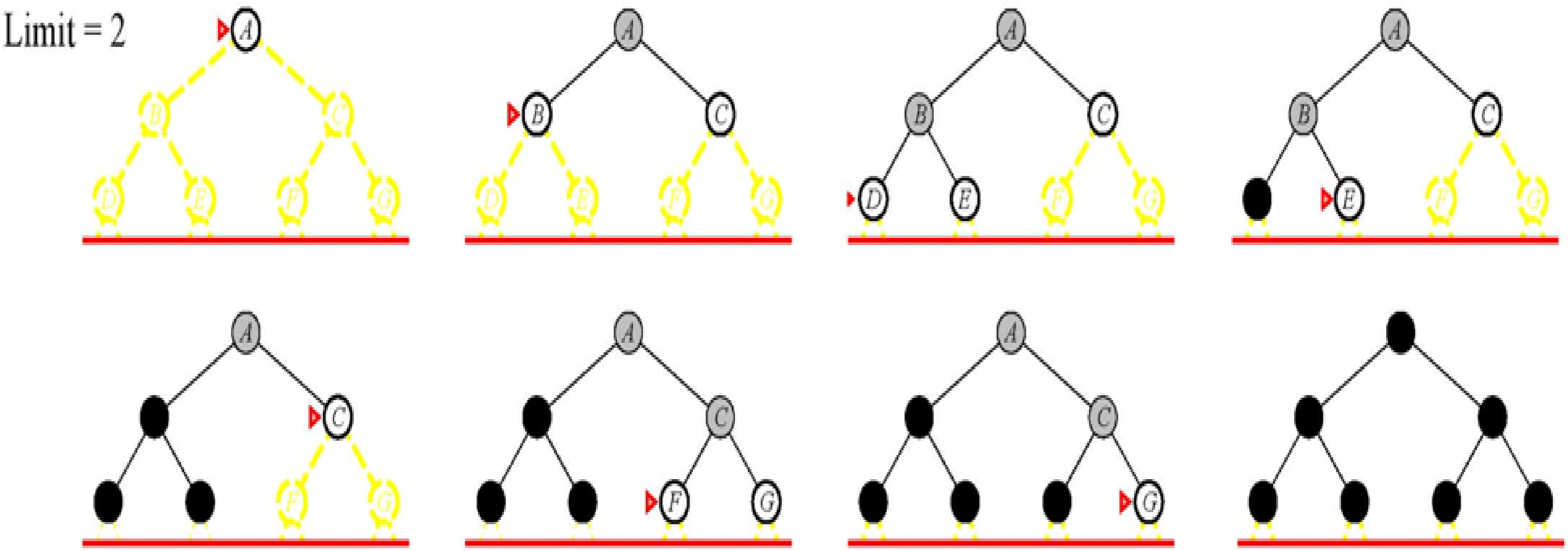
- Depth first search with depth limit L
  - Nodes at depth L are not expanded
- Eliminates problem with infinite path
- How to select L?
- Possible failures:
  - No solution
  - Cutoff – no solution within the depth limit

Iterative deepening depth first search

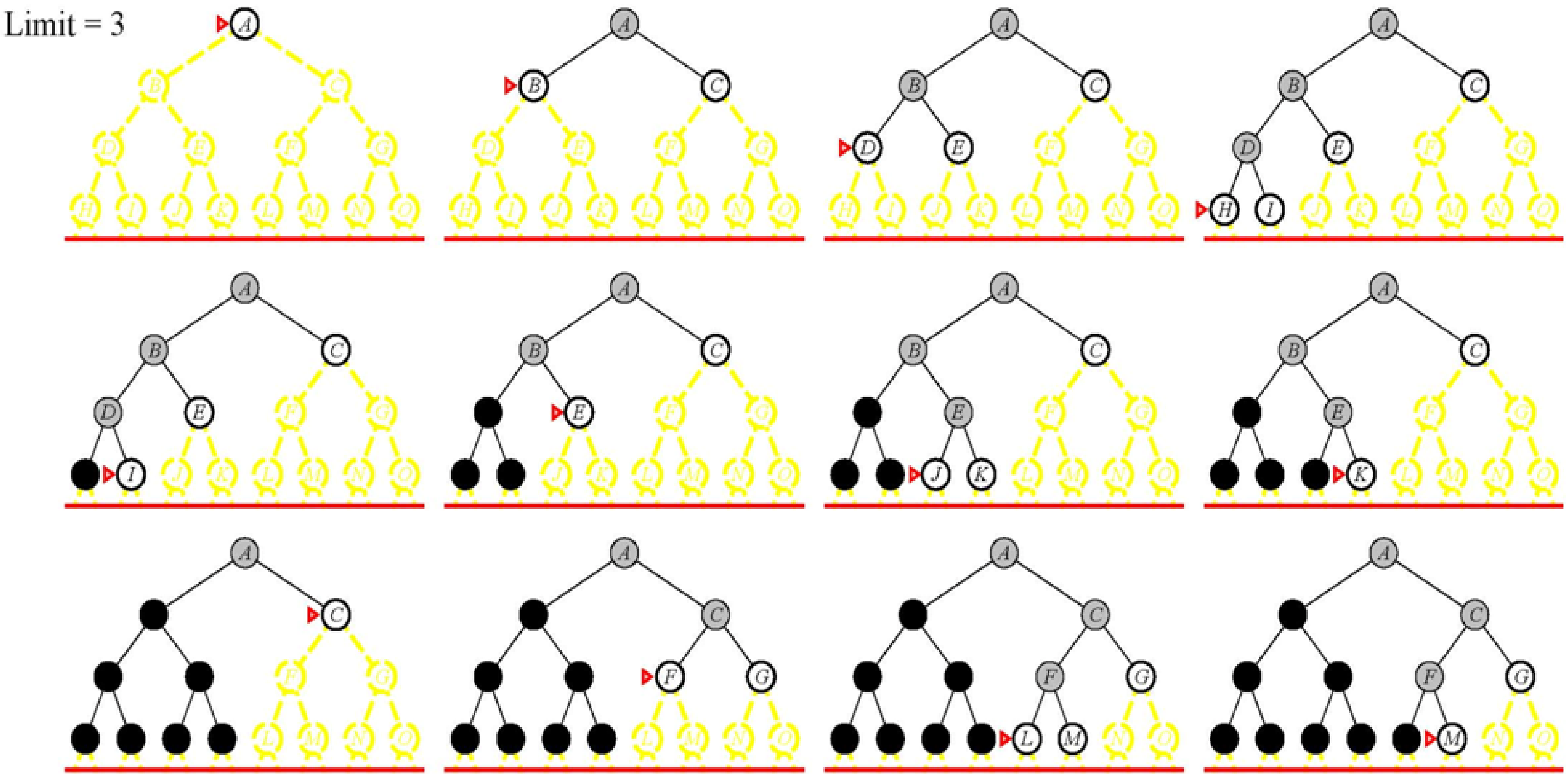
- Repeat Depth-limited search with L=1,2,3,…



Iterative deepening depth first search



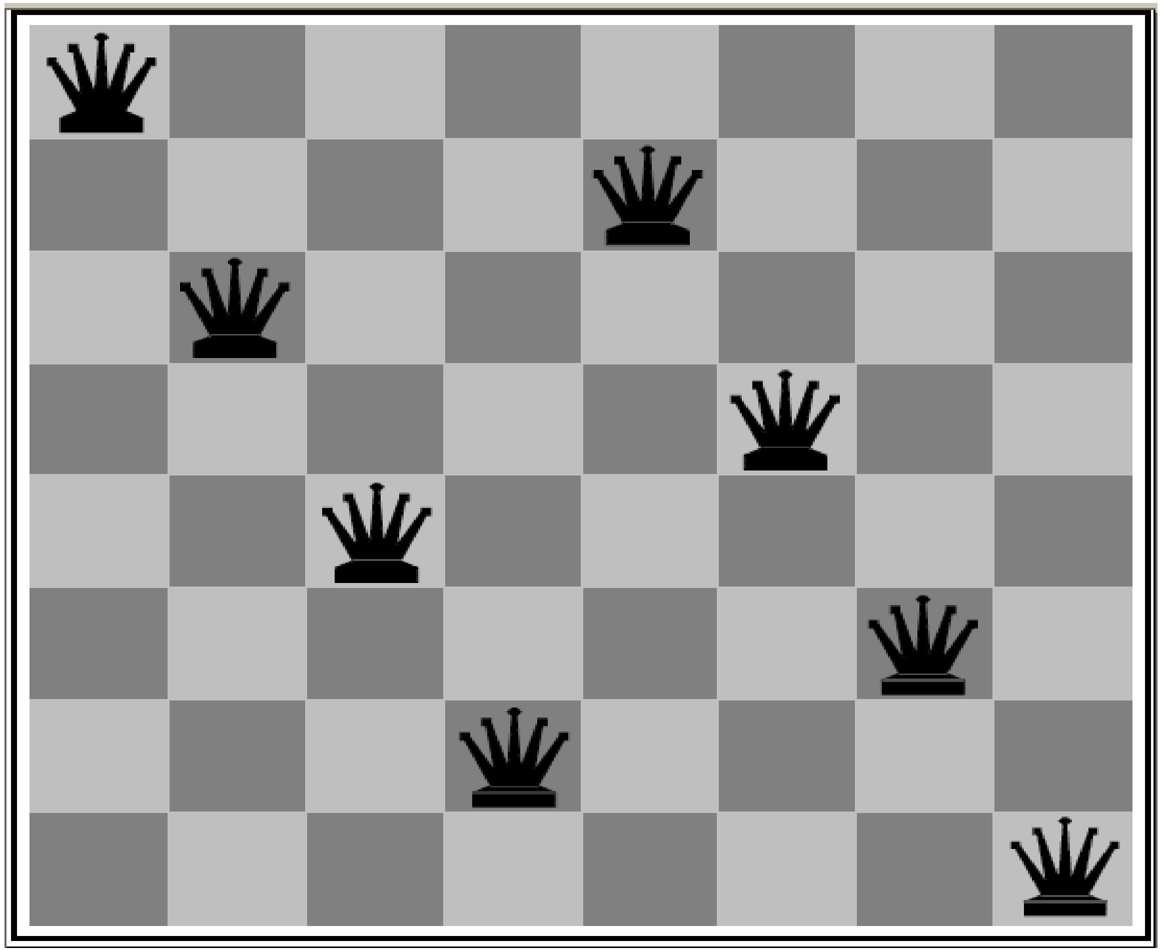
Iterative deepening depth first search



Properties of Iterative deepening search

- Complete
- Time
  - $O(b^d)$: $(d+1)(1) + db + (d-1)b^2 + \ldots + (1)\,b^d$
- Space:
  - $O(bd)$
- Optimal if step cost is 1

Construct Search Tree for 8-Queens Problem?



Construct search tree for 8-Puzzle?

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Exam question: *18.10.2007*

(Artificial Intelligence, Russell and Norvig): The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of the river, along with a boat that can hold one or two people. Find a way to get everyone to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968)*.* (18 Points)

- Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.
- Solve this problem by using depth first search and breadth first search. (Check for the repeated states).

DEMO...

- Rotating Workforce Scheduling

	Mo	Tu	We	Th	Fr	Sa	Su
A	D	D	D			D	D
B	D	D	D	D			
C	A	A	A	A			A
D	A	A	A	A	A		
E	D	D	A	A	A		
F	A	A	N	N	N		
G	N	N	N	N	N		
H		N	N	N	N	N	
I			D	D	D	A	A
J				D	D	N	N
K	N				A	A	N
L	N	N			D	D	D

Illustration of depth first search and backtracking ...

Literature

- Artificial Intelligence: A Modern Approach (Stuart Russell and Peter Norvig)
  - Chapter 3

- How to Solve It: Modern Heuristics (Z. Michalewicz and D. B. Fogel.)
  - Chapter 3