Uninformed Search Strategies

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

Contents

What is Uninformed Search 什么是无信息搜索

- The uninformed search is also called blind search.

  无信息搜索也被称为盲目搜索。

- The term (uninformed, or blind) means that the search strategies have no additional information about states beyond that provided in the problem definition.

  该术语（无信息、盲目的）意味着该搜索策略没有超出问题定义提供的状态之外的附加信息。

- All they can do is to generate successors and distinguish a goal state from a non-goal state.

  所有能做的就是生成后继结点，并且从区分一个目标状态或一个非目标状态。

What is Uninformed Search 什么是无信息搜索

- All search strategies are distinguished by the *order* in which nodes are expanded.
  所有的搜索策略是由节点扩展的顺序加以区分。
- The search strategies: breadth-first, depth-first, and uniform-cost search.
  这些搜索策略是：宽度优先、深度优先、以及一致代价搜索。

Uninformed Search Strategy Evaluation 无信息搜索策略评价

- An uninformed search strategy is defined by picking the order of node expansion.

  一种无信息搜索策略是通过其选择节点扩展的顺序来定义的。

- The strategies can be evaluated along the following properties:

  其策略可按照如下特性来评价：

  - Completeness: Does it always find a solution if one exists?

    完备性：是否总能找到一个存在的解？

  - Time complexity: How long does it take to find a solution?

    时间复杂性：花费多长时间找到这个解？

  - Space complexity: How much memory is needed?

    空间复杂性：需要多少内存？

  - Optimality: Does it always find the optimal solution?

    最优性：是否总能找到最优的解？

Uninformed Search Strategy Evaluation 无信息搜索策略评价

- Time complexity and space complexity are measured in following terms:
  时间复杂性和空间复杂性用如下术语来度量：
  - b -- maximum branching factor of the search tree.
    搜索树的最大分支因子。
  - d -- depth of the shallowest solution.
    最浅解的深度。
  - m -- maximum depth of the search tree.
    搜索树的最大深度。

Breadth-first Search 宽度优先搜索

□ Search Strategy 搜索策略

Expand shallowest unexpanded node.

扩展最浅的未扩展节点。

□ Implementation 实现方法

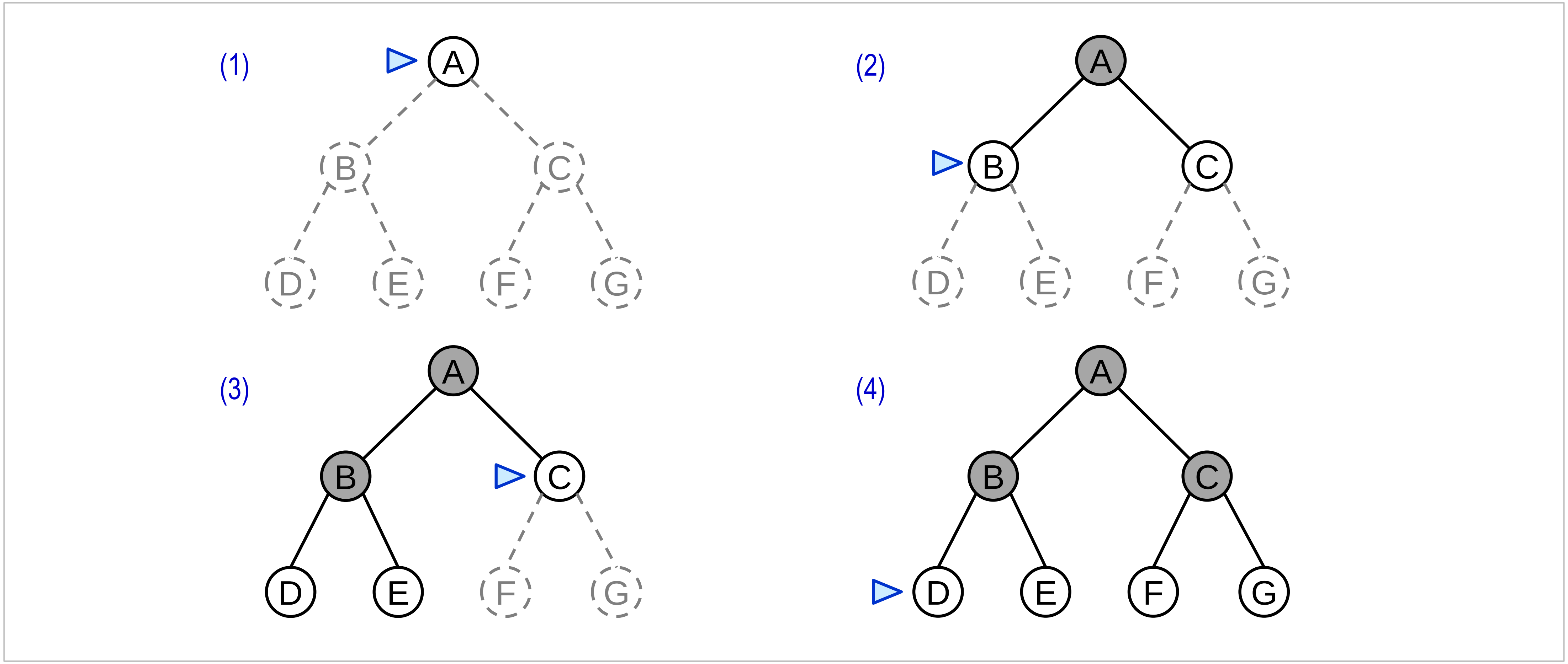
Use FIFO (First-In First-Out) queue, i.e., new successors go at end.

使用FIFO队列，即新的后继节点放在后面。

Breadth-first Search Algorithm on a Graph 图的宽度优先搜索算法

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
    node ← a node with STATE = problem.INITIAL-STATE
    PATH-TEST = 0
    frontier ← a FIFO queue with node as the only element
    explored ← an empty set
    loop do
        if EMPTY ? (frontier) then return failure
        node ← POP(frontier)        /* chooses the shallowest node in frontier */
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
                frontier ← INSERT(child, frontier)
```

Breadth-first Search on a Simple Binary Tree 简单二叉树的宽度优先搜索



At each stage the node to be expanded next is indicated by a marker.

Properties of Breadth-first Search 宽度优先搜索的性质

- **Time complexity** 时间复杂性 $b + b^2 + b^3 + \ldots + b^d = O(b^d)$

- **Space complexity** 空间复杂性 $O(b^d)$

where

- b -- the branching factor
  分枝因子
- d -- the depth of the shallowest solution
  最浅解的深度

Time and Memory Requirements 时间和内存需求

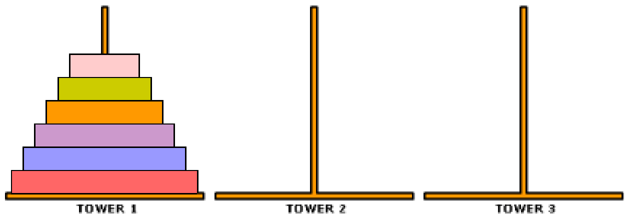
Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Assume: branching factor b = 10; 1 million nodes/second; 1000 bytes/node.

- Memory requirements are a bigger problem, execution time is still a major factor.
  内存的需求是一个很大的问题，而执行时间仍是一个主要因素。
- Breadth-first search cannot solve exponential complexity problems but small branching factor.
  宽度优先搜索不能解决指数复杂性的问题，小的分支因子除外。

Example: Tower of Hanoi 汉诺塔问题

- It is said that there is an Indian temple which contains 3 towers by 64 golden disks.
  据说一个印度寺庙里有3个塔、塔上有64个金盘。
- Priests have been moving these disks, sample rules:
  祭司一直在移动这些金盘，规则很简单：
  - Only one disk can be moved at a time.
    每次仅能移动一个金盘。
  - A disk can only be moved if it is uppermost disk.
    仅能移动最上面的那块金盘。
  - No disk may be placed on top of a smaller disk.
    大的金盘不能放在小的金盘上面。
- According to the legend, the world will end when last move.
  据传说，当最后一次移动金盘时，世界将会毁灭。



Assume:
moving 1 disk/second;
it will take $2^{64}-1$ seconds
≈ 585 billion years.

Thank you for your attention!