算法设计与分析

搜索策略

问题求解

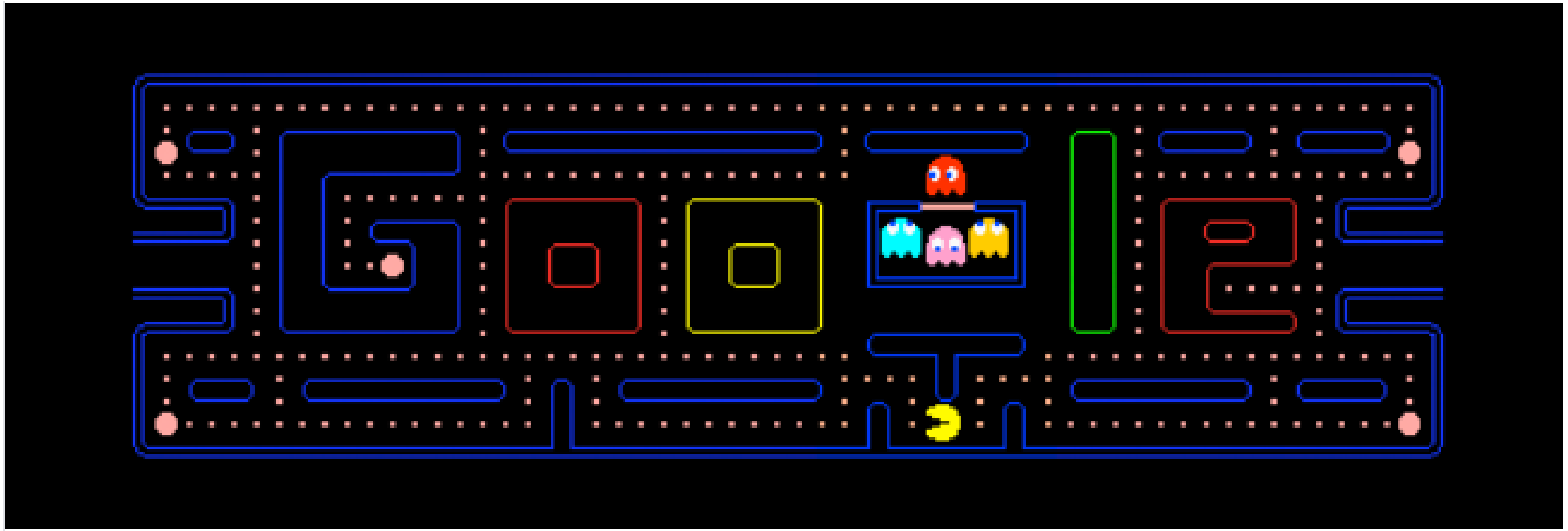
- 要进行问题求解，首先要讨论的是对问题及其解的精确定义
- 搜索，是指从问题出发寻找解的过程

一些极具意义的实际搜索问题

- VLSI layout
  - 超大规模集成电路布局问题，布置元器件、连线，要求面积、电路延迟、分布电容最小，产量最大
- Robot navigation
  - 机器人可在连续的空间上运动，而且可能的行动和状态集都是无限的

搜索问题

- PAC-MAN

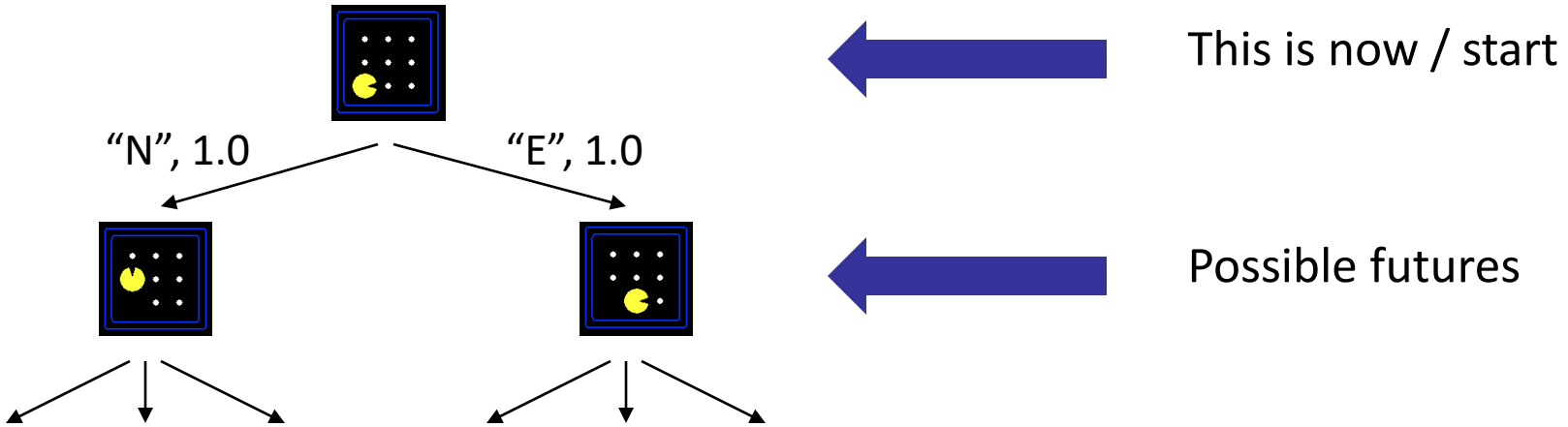


通过搜索求解

- 在对问题进行形式化之后，现在需要对问题求解
- 一个解是一个行动序列，所以搜索算法的工作就是考虑各种可能的行动序列
- 可能的行动序列从搜索树中根结点的初始状态出发
  - 连线表示行动
  - 结点对应问题的状态空间中的状态
- 第一步检测该结点是否为目标状态
- 进而考虑选择各种行动，通过扩展当前状态完成
- 选择一条路往下走，把其他的选择暂且放在一边，等以后发现第一个选择不能求出问题的解时再考虑

搜索树

- 搜索树
  - 根节点对应了初始状态
  - 子节点对应了父节点的后继
  - 节点显示状态，但对应的是到达这些状态的行动
  - 对大多数问题，实际上不会构建整个树



无信息搜索策略
（Uninformed search, also called blind search）

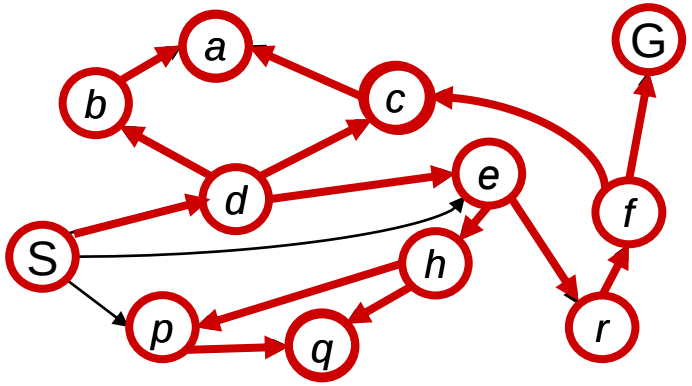
- 深度优先搜索（DFS）
- 宽度优先搜索（BFS）
- 深度受限搜索（DLS）
- 迭代加深的深度优先搜索（IDS）
- 双向搜索（Bidirectional search）

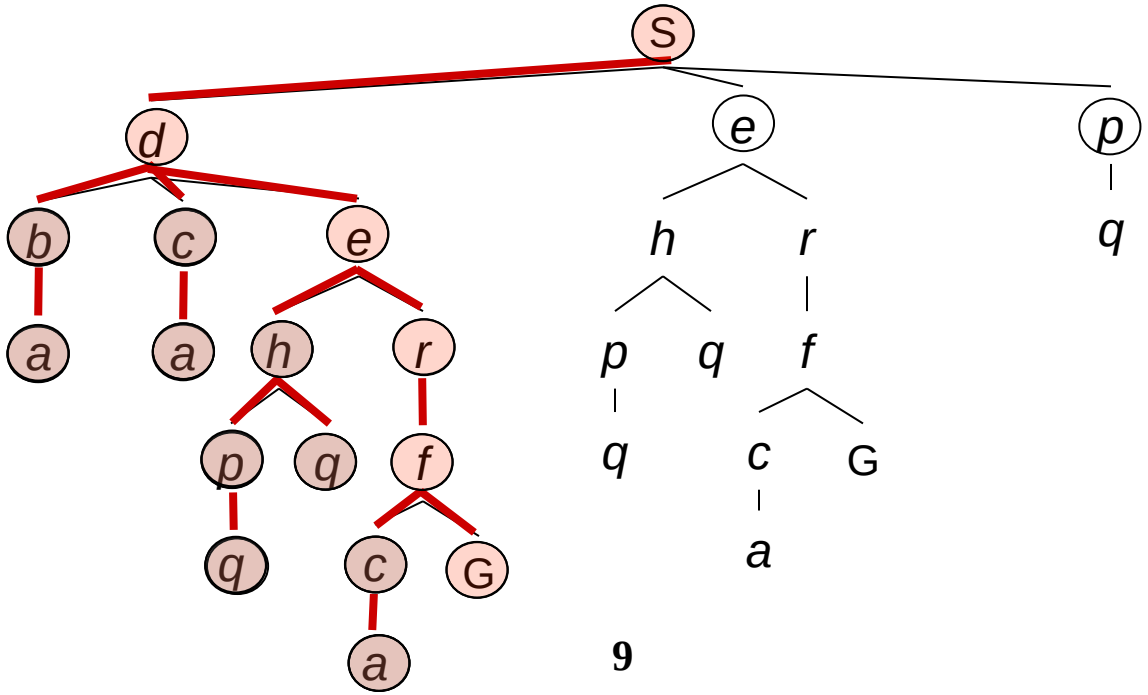
深度优先搜索（Depth-First Search, DFS）

深度优先搜索（Depth-First Search, DFS）

Strategy: expand a deepest node first

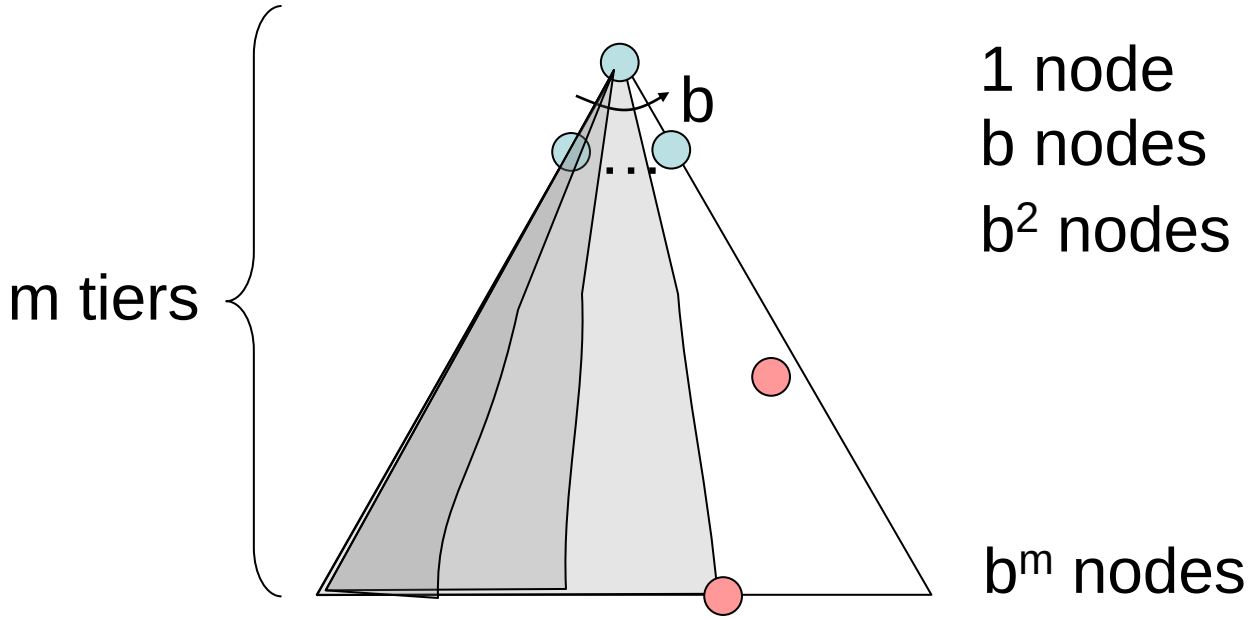
Implementation: Fringe is a LIFO stack



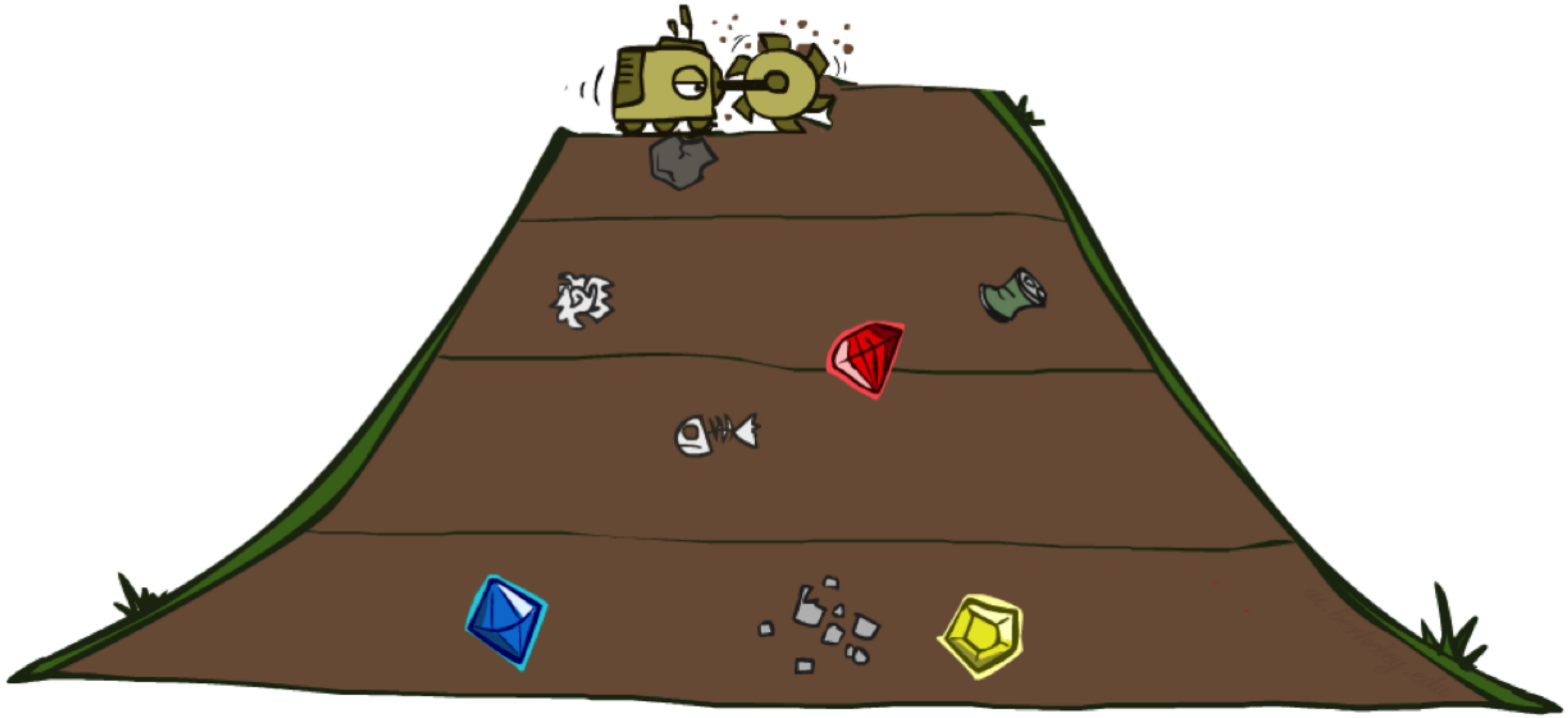


深度优先搜索（Depth-First Search, DFS）

- DFS扩展哪些结点？
  - Some left prefix of the tree.
  - Could process the whole tree!
  - If m is finite, takes time O(b^m)
- 内存需求？
  - Only has siblings on path to root, so O(bm)



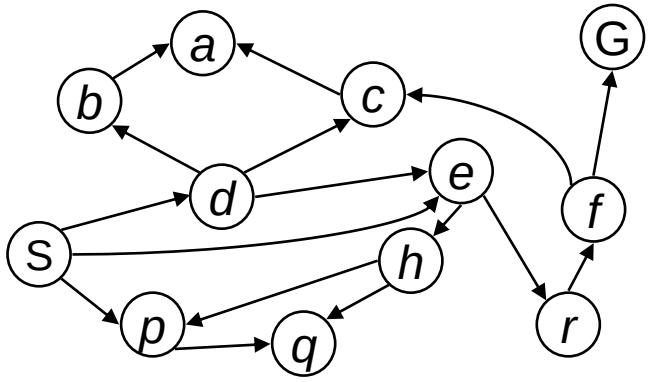
宽度优先搜索（Breadth-First Search, BFS）

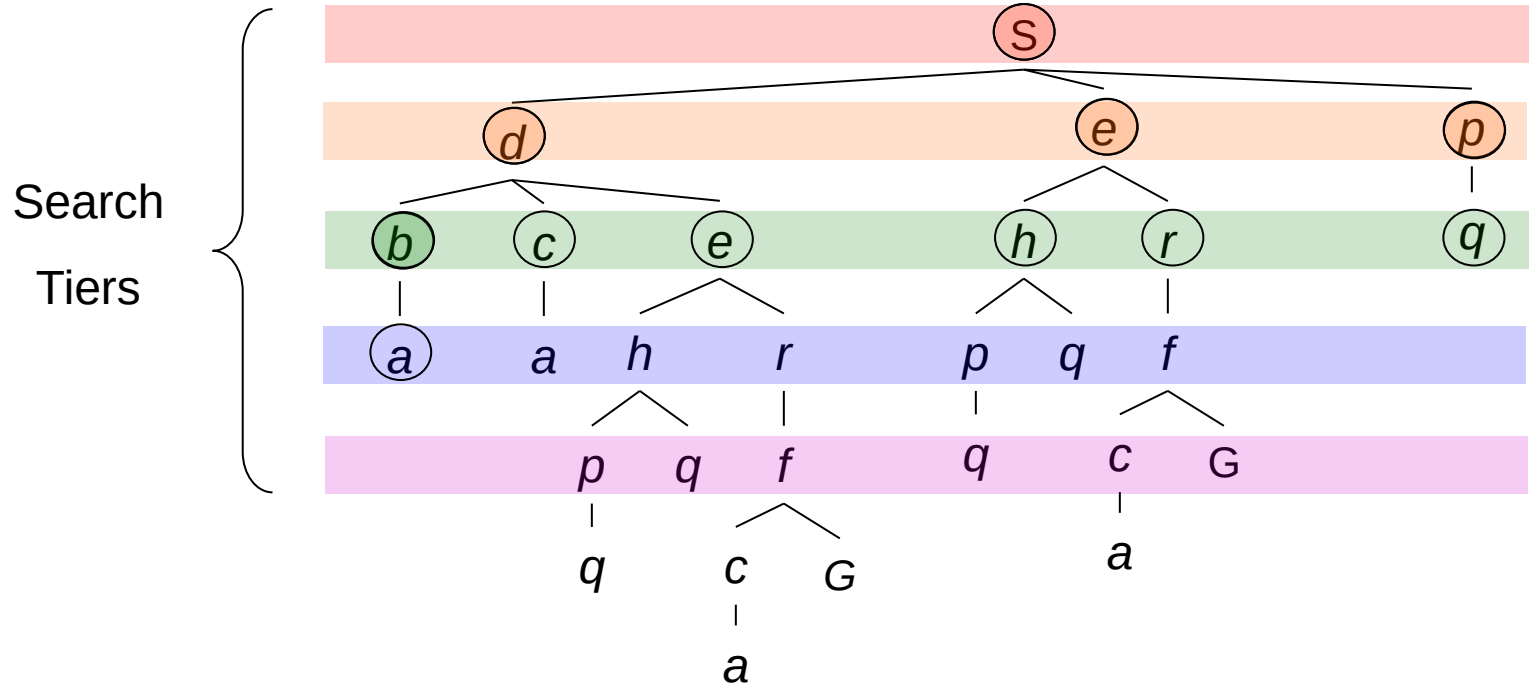


宽度优先搜索（Breadth-First Search, BFS）

Strategy: expand a shallowest node first

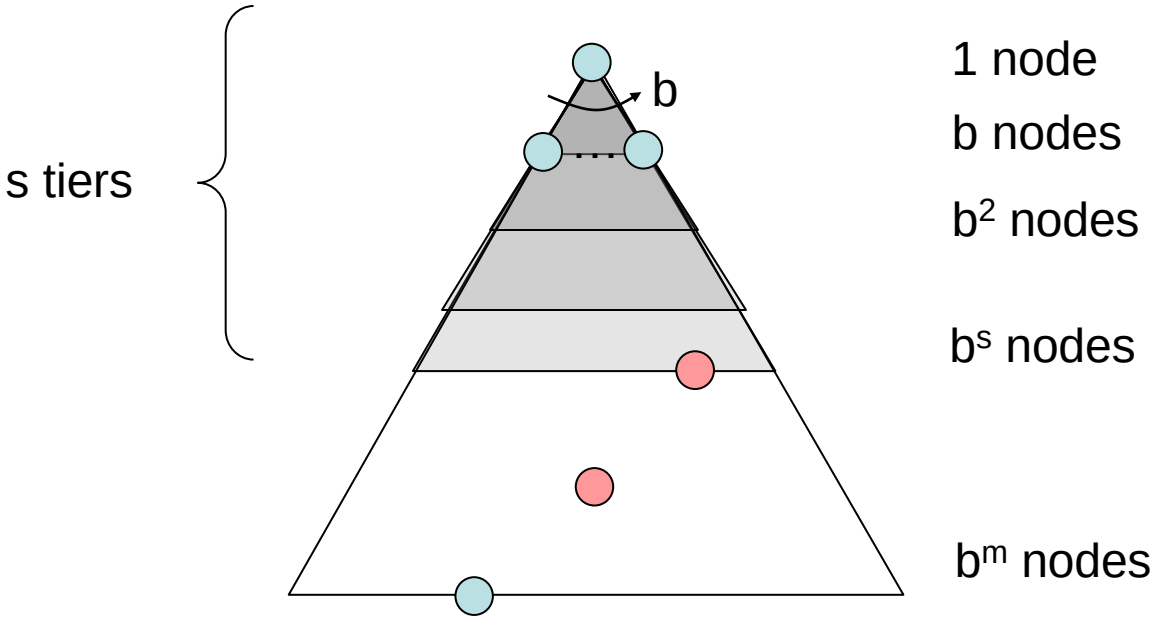
Implementation: Fringe is a FIFO queue





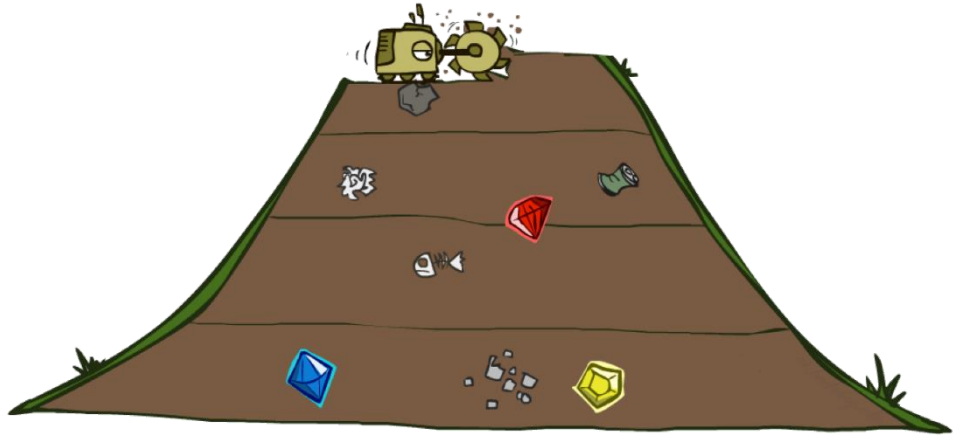
宽度优先搜索（Breadth-First Search, BFS）

- BFS扩展哪些节点?
  - Processes all nodes above shallowest solution
  - Let depth of shallowest solution be s
  - Search takes time O(b^s)



- 内存需求?
  - Has roughly the last tier, so O(b^s)

DFS vs BFS



DFS vs BFS

- 什么情况下BFS优于DFS?

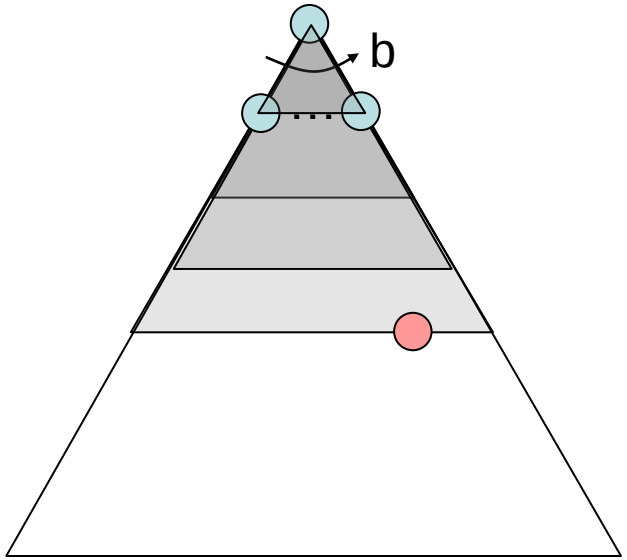
- 什么情况下DFS优于BFS?

深度受限搜索（Depth-limited search, DLS）

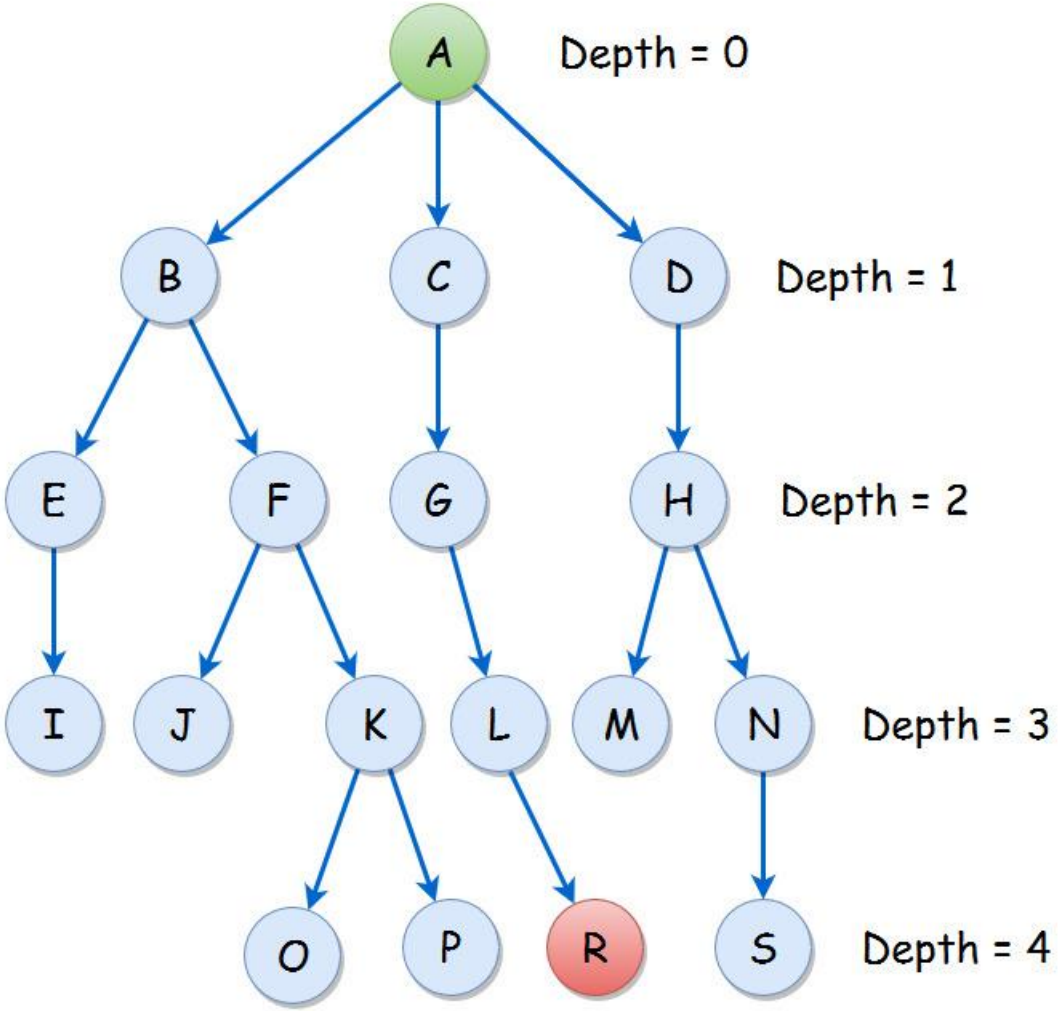
- DLS
  - 对深度优先搜索设置界限l
  - 深度为l的结点被当作没有后继对待
- DLS
  - 深度界限解决了无穷路径的问题
  - 时间复杂度是$O(bl)$
  - 空间复杂度是$O(bl)$
- 深度优先搜索可以看作是特殊的深度受限搜索，其深度$l = \infty$

迭代加深的深度优先搜索（Iterative deepening search, IDS）

- 思路：结合DFS的空间优势与BFS的时间优势
  - Run a DFS with depth limit 1. If no solution…
  - Run a DFS with depth limit 2. If no solution…
  - Run a DFS with depth limit 3. …..
  - until a goal is found
- 浪费冗余?
  - 通常绝大多数的结点都在底层，所以上层的节点生成多次影响不是很大。

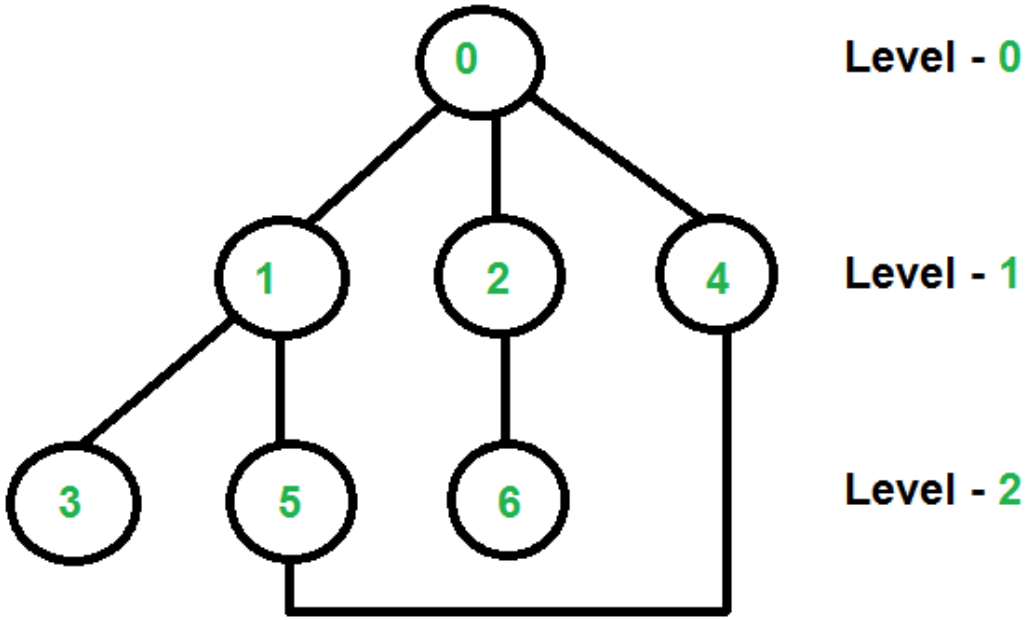


迭代加深的深度优先搜索
（Iterative deepening search, IDS）



迭代加深的深度优先搜索
（Iterative deepening search, IDS）

Depth	Iterative Deepening Depth First Search
0	0
1	0 1 2 4
2	0 1 3 5 2 6 4 5
3	0 1 3 5 4 2 6 4 5 1



迭代加深的深度优先搜索
（Iterative deepening search, IDS）

- 结合了深度优先搜索和宽度优先搜索的优点
- 它的空间需求和宽度优先搜索一样

双向搜索（Bidirectional search）

- 思路：同时运行两个搜索
  - 一个从初始状态向前搜索
  - 同时另一个从目标状态向后搜索
  - 希望它们在中间某点相遇，此时搜索中止

小结

- DFS
- BFS
- DLS
- IDS
- Bidirectional search
- 有信息的情况下怎么搜索呢？