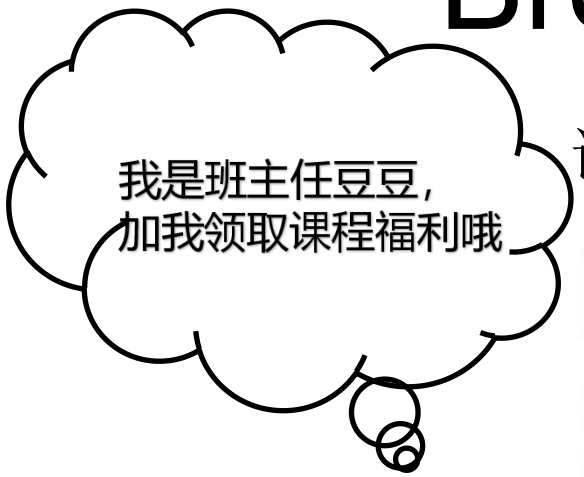
性价比之王的宽度优先搜索
Breadth First Search



课程版本 v6.1

主讲 侯卫东

加班主任，进班级答疑群
快速获取面试资料/课程福利

关注公众号，了解大厂资讯

版权声明

九章的所有课程均受法律保护，不允许录像与传播录像

一经发现，将被追究法律责任和赔偿经济损失

先修内容

请在随课教程中先修如下内容：

http://www.jiuzhang.com/tutorial/algorithm/14

- 什么是队列，如何自己实现一个队列
- 什么是 Interface，LinkedList 和 Queue 之间的关系是什么？
- 什么是拓扑排序 Topological Sorting
- 如何定义一个图的数据结构

大纲 Outline

宽度优先搜索算法是一个性价比极高的算法
如果你只有很短的时间准备面试（比如一周），那么你应该把时间花在宽度优先搜索上

- 什么时候使用 BFS
- 二叉树上的 BFS
- 图上的 BFS
- 矩阵上的 BFS
- 拓扑排序 Topological Sorting

什么时候应该使用BFS？

图的遍历 Traversal in Graph

- 层级遍历 Level Order Traversal
- 由点及面 Connected Component
- 拓扑排序 Topological Sorting

最短路径 Shortest Path in Simple Graph

- 仅限简单图求最短路径
- 即，图中每条边长度都是1，或者边长都相等

非递归的方式找所有方案 Iteration solution for all possible results

- 这一点我们将在后面 DFS 的课上提到

如果题目问最短路径

除了BFS还可能是什么算法？

如果问最长路径呢？

题型与算法的对应关系

最短路径

简单图 → BFS

复杂图 → Dijkstra, SPFA, Floyd（一般面试不考这个）

最长路径

图可以分层 → Dynamic Programming

分层：比如走到第i一定会经过第 i-1 层（棋盘格子图是典型的例子）

不可以分层 → DFS

二叉树上的宽度优先搜索

BFS in Binary Tree

Binary Tree Level Order Traversal

http://www.lintcode.com/problem/binary-tree-level-order-traversal/

http://www.jiuzhang.com/solutions/binary-tree-level-order-traversal/

图的遍历（层级遍历）

注：树是图的一种特殊形态，树属于图

使用队列作为主要的数据结构 Queue

思考：用栈（Stack）是否可行？为什么行 or 为什么不行？

是否需要实现分层？

需要分层的算法比不需要分层的算法多一个循环

Java / C++:

size=queue.size()

如果直接 for (int i = 0; i < queue.size(); i++) 会怎么样？

Binary Tree Serialization

问：什么是序列化？

什么是序列化？

将“**内存**”中结构化的数据变成“**字符串**”的过程

序列化：object to string

反序列化：string to object

什么时候需要序列化？

1. 将内存中的数据持久化存储时

内存中重要的数据不能只是呆在内存里，这样断电就没有了，所需需要用一种方式写入硬盘，在需要的时候，能否再从硬盘中读出来在内存中重新创建

2. 网络传输时

机器与机器之间交换数据的时候，不可能互相去读对方的内存。只能讲数据变成字符流数据（字符串）后通过网络传输过去。接受的一方再将字符串解析后到内存中。

常用的一些序列化手段：

- XML
- Json
- Thrift (by Facebook)
- ProtoBuf (by Google)

序列化算法

一些序列化的例子：

- 比如一个数组，里面都是整数，我们可以简单的序列化为”[1,2,3]”
- 一个整数链表，我们可以序列化为，”1->2->3”
- 一个哈希表(HashMap)，我们可以序列化为，”{\”key\”: \”value\”}”

序列化算法设计时需要考虑的因素：

- **压缩率**。对于网络传输和磁盘存储而言，当然希望更节省。
  - 如 Thrift, ProtoBuf 都是为了更快的传输数据和节省存储空间而设计的。
- **可读性**。我们希望开发人员，能够通过序列化后的数据直接看懂原始数据是什么。
  - 如 Json，LintCode 的输入数据

二叉树如何序列化？

你可以使用任何你想要用的方法进行序列化，只要保证能够解析回来即可。

LintCode 采用的是 BFS 的方式对二叉树数据进行序列化，这样的好处是，你可以更为容易的自己画出整棵二叉树。

算法描述：

http://www.lintcode.com/en/help/binary-tree-representation/

- 按照BFS顺序

题目及解答：

http://www.lintcode.com/en/problem/binary-tree-serialization/

http://www.jiuzhang.com/solutions/binary-tree-serialization/

相关问题 Related Problems

Binary Tree Level Order Traversal II （从下往上）

http://www.lintcode.com/en/problem/binary-tree-level-order-traversal-ii/

http://www.jiuzhang.com/solutions/binary-tree-level-order-traversal-ii/

Binary Tree Zigzag Order Traversal（奇数层从左往右，偶数层从右往左）

http://www.lintcode.com/en/problem/binary-tree-zigzag-level-order-traversal/

http://www.jiuzhang.com/solutions/binary-tree-zigzag-level-order-traversal/

Convert Binary Tree to Linked Lists by Depth （每一层建立一个单向链表）

http://www.lintcode.com/en/problem/convert-binary-tree-to-linked-lists-by-depth/

http://www.jiuzhang.com/solutions/convert-binary-tree-to-linked-lists-by-depth/

图上的宽度优先搜索

BFS in Graph

问：和树上有什么区别？

哈希表

图中存在环

存在环意味着，同一个节点可能重复进入队列

Java: *HashMap* / *HashSet*

C++: *unordered_map* / *unordered_set*

Python: *dict* / *set*

Clone Graph (F)

http://www.lintcode.com/problem/clone-graph/

http://www.jiuzhang.com/solutions/clone-graph/

BFS遍历图，得到点

然后复制点和边

BFS 的时间复杂度

O(N + M)

其中 N 为点数，M 为边数

Word Ladder

http://www.lintcode.com/problem/word-ladder/

http://www.jiuzhang.com/solution/word-ladder/

最典型的BFS问题 —— **隐式图** (Implicit Graph) 最短路径

休息5分钟

take a break

矩阵中的宽度优先搜索

BFS in Matrix

矩阵 vs 图

图 Graph

N个点，M条边

M最大是 O(N^2) 的级别

图上BFS时间复杂度 = O(N + M)

- 说是O(M)问题也不大，因为M一般都比N大

所以最坏情况可能是 O(N^2)

矩阵 Matrix

R行C列

R*C个点，R*C*2 条边（每个点上下左右4条边，每条边被2个点共享）。

矩阵中BFS时间复杂度 = O(R * C)

Number of Islands

http://www.lintcode.com/problem/number-of-islands/

http://www.jiuzhang.com/solutions/number-of-islands/

图的BFS遍历

每一次从一块陆地遍历获得一个岛

坐标变换数组

```
int[] deltaX = {1,0,0,-1};
int[] deltaY = {0,1,-1,0};
```

问：写出八个方向的坐标变换数组？

更多 Union Find 有关的问题

将在《九章算法强化班》中讲解

并查集 Union Find

Knight Shortest Path

http://www.lintcode.com/problem/knight-shortest-path/

http://www.jiuzhang.com/solutions/knight-shortest-path/

简单图最短路径

follow up: speed up?（见随课教程）

拓扑排序 Topological Sorting

几乎每个公司都有一道拓扑排序的面试题！

BFS or DFS?

独孤九剑——破剑式

能够用 BFS 解决的问题，一定**不要用** DFS 去做！

因为用 Recursion 实现的 DFS 可能造成 StackOverflow!

(Iteration 的 DFS 一来你不会写，二来面试官也看不懂)

拓扑排序 Topological Sorting

入度（In-degree）：

有向图（Directed Graph）中指向当前节点的点的个数（或指向当前节点的边的条数）

算法描述：

1. 统计每个点的入度
2. 将每个入度为 0 的点放入队列（Queue）中作为起始节点
3. 不断从队列中拿出一个点，去掉这个点的所有连边（指向其他点的边），其他点的相应的入度 - 1
4. 一旦发现新的入度为 0 的点，丢回队列中

拓扑排序并不是传统的排序算法

一个图可能存在多个拓扑序（Topological Order），也可能不存在任何拓扑序

Topological Sorting

http://www.lintcode.com/problem/topological-sorting/

http://www.jiuzhang.com/solutions/topological-sorting/

该问题保证一定存在至少一个拓扑序

拓扑排序的四种不同问法

求任意1个拓扑序（Topological Order）

问是否存在拓扑序（是否可以被拓扑排序）

求所有的拓扑序

求是否存在且仅存在一个拓扑序

拓扑排序的四种不同问法

求任意1个拓扑序（Topological Order）

问是否存在拓扑序（是否可以被拓扑排序）

求所有的拓扑序 **DFS**

求是否存在且仅存在一个拓扑序 **Queue中最多同时只有1个节点**

Course Schedule I && II (GAFZ)

http://www.lintcode.com/problem/course-schedule/

http://www.lintcode.com/problem/course-schedule-ii/

换了个皮，分别问有没有拓扑序和给出一个拓扑序

Alien Dictionary (GFTSAP)

http://www.lintcode.com/problem/alien-dictionary/

http://www.jiuzhang.com/solution/alien-dictionary/

相似问题：http://www.lintcode.com/problem/sequence-reconstruction/

考点1：如何构建图

考点2：如何存储图

考点3：如何拓扑排序

相关问题

- **图上的BFS**
- 判断一个图是否是一棵树
- http://www.lintcode.com/problem/graph-valid-tree/
- 搜索图中最近值为target的点
- http://www.lintcode.com/problem/search-graph-nodes/
- 无向图联通块
- http://www.lintcode.com/problem/connected-component-in-undirected-graph/
- 序列重构（判断是否只有一个拓扑排序）
- http://www.lintcode.com/problem/sequence-reconstruction/
- **矩阵上的BFS**
- 僵尸多少天吃掉所有人
- http://www.lintcode.com/problem/zombie-in-matrix/
- 建邮局问题 Build Post Office II
- http://www.lintcode.com/problem/build-post-office-ii/

总结 Conclusion

- 能用 BFS 的一定不要用 DFS（除非面试官特别要求）
- BFS 的两个使用条件
  - 图的遍历（由点及面，层级遍历）
  - 简单图最短路径
- 是否需要层级遍历
  - size = queue.size()
- 拓扑排序必须掌握！
- 坐标变换数组
  - deltaX, deltaY
  - inBound

课后补充内容

请在随课教程中自学如下内容：

http://www.jiuzhang.com/tutorial/algorithm/367

另外两种宽度优先搜索算法的实现方式

双向宽度优先搜索算法（Bidirecitonal BFS）