

搜索类题目如何高效实现



扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: [ninechapter](#)

知乎专栏: <http://zhuanlan.zhihu.com/jiuzhang>

微博: <http://www.weibo.com/ninechapter>

官网: www.jiuzhang.com

- 把BST规范的画出来是什么样子？
 - 向下投影出的序列是从____到____的顺序 （填空，一空一字）
- 按照DFS的定义宏观理解，比较适合哪种类型的DFS？（单选）
 - a. 分治型DFS b. 枚举型DFS
- BST上的操作和哪一种算法比较相似？

- DFS类问题（3题）
- BFS类问题（2题）
- 快速点题（6题）

DFS类问题

Letter Combinations of a Phone Number

<http://www.lintcode.com/zh-cn/problem/letter-combinations-of-a-phone-number/>

<http://www.jiuzhang.com/solutions/letter-combinations-of-a-phone-number/>

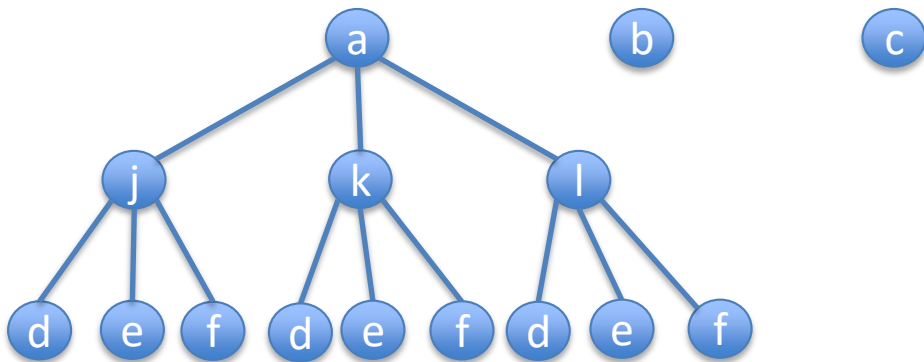
Letter Combinations of a Phone Number

思路:

- 基础枚举型DFS（按方法1 执行过程理解）
 - 输入数字串长度为n，做n个阶段的选择，DFS n层
 - 每一阶段枚举该位的数字对应的一个字母
 - 直到所有情况都枚举完

输入 2 5 3

2: a b c
5: j k l
3: d e f



Letter Combinations of a Phone Number

思路:

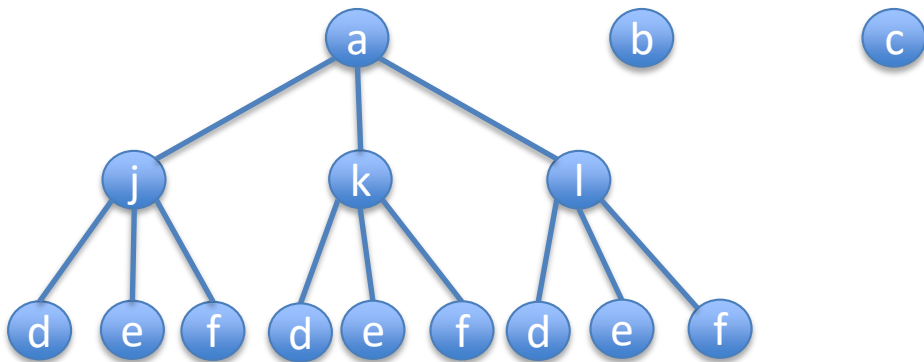
- 怎样写DFS比较好想，比较快？
 - 先写扩展，再写退出情况

输入 2 5 3

2: a b c

5: j k l

3: d e f



- DFS中间状态怎么记录？比如现在在第几层，现在得到的字符串是什么？
 1. 最简单、不易出错：全部放在DFS函数的参数中 coding time
 2. 中间不改变的东西：可以放在成员变量中，比如这一题的l, digits, phone
 3. 改变的记录状态的数组：可以放在函数参数或成员变量中，在DFS时手动模拟栈操作（回溯），比如这题的str

Letter Combinations of a Phone Number



- Company Tags: Amazon Google Facebook

考点:

- 基础DFS的快速实现
- 电面题

能力维度：

2. 代码基础功力
3. 基础数据结构/算法
5. 细节处理（**corner case**）

Word Squares

<http://www.lintcode.com/zh-cn/problem/word-squares/>

<http://www.jiuzhang.com/solutions/word-squares/>

思路:

- 裸写DFS很简单:
 - 枚举每一行选哪个单词
 - 枚举完后check是否满足条件
 - 但是时间上过不了 最坏 1000^5

- 所以怎么办？
 - DFS+剪枝
- DFS中，什么是剪枝？
 - 剪枝就是去掉搜索过程中的冗余
- 什么是冗余？
 - 就是搜到某个情况下，明显往后继续搜是搜不出结果的

- 冗余一：
 - 第一个词填了**ball**后，第二个词必须以**a**开头
 - 第二个词填了**area**后，第三个词必须以**le**开头
 - 以其他开头的就没必要搜下去了

b	a	l	l
a			
l			
l			

- 冗余一：
 - 怎么实现？
 - 用Hash or Trie树记录下以某个前缀开头的有哪些单词
 - 比如以l开头的有lead lady，以le开头的有lead，以lea开头的有lead
 - 每次只用从特定开头的单词中继续往后搜

b	a	l	l
a			
l			
l			

lead lady	lead	lead	lead
l	le	lea	lead

- 冗余二：
 - 第一个词填了ball
 - 第二个词想填area的话
 - 字典中必须有以le la开头的单词，否则没有的话就不能填area

b	a	l	l
a	r	e	a
l	e		
l	a		

- 冗余二：
 - 怎么实现？
 - 直接利用冗余一中的Hash or Trie树

b	a	l	l
a	r	e	a
l	e		
l	a		

- 见代码

- Company Tags: Google
- 考点:
- DFS过程中的剪枝
- Hash Trie树的应用

能力维度：

2. 代码基础功力
3. 基础数据结构/算法
4. 逻辑思维/算法优化能力
5. 细节处理（**corner case**）

- <http://www.jiuzhang.com/course/9/questionnaire>

Expression Add Operators

<http://www.lintcode.com/problem/expression-add-operators/>

<http://www.jiuzhang.com/solution/expression-add-operators/>

思路:

- 简化版，只有加减没有乘号怎么做？
 - 枚举数字后再枚举数字前的符号（第一个数字前不枚举符号）
 - `sum`作为中间状态记录下前面的和
- 样例数据模拟

3 4 5 6 2 3 7 4 9 0

`sum=0`

思路:

- 简化版，只有加减没有乘号怎么做？
 - 枚举数字后再枚举数字前的符号（第一个数字前不枚举符号）
 - `sum`作为中间状态记录下前面的和
- 样例数据模拟

3 4 5 6 2 3 7 4 9 0

3456

sum=3456

思路:

- 简化版，只有加减没有乘号怎么做？
 - 枚举数字后再枚举数字前的符号（第一个数字前不枚举符号）
 - `sum`作为中间状态记录下前面的和
- 样例数据模拟

3 4 5 6 2 3 7 4 9 0

3456-23

sum=sum-23=3433

思路:

- 简化版，只有加减没有乘号怎么做？
 - 枚举数字后再枚举数字前的符号（第一个数字前不枚举符号）
 - `sum`作为中间状态记录下前面的和
- 样例数据模拟

3 4 5 6 2 3 7 4 9 0

3456-23-7

sum=sum-7=3426

思路:

- 简化版，只有加减没有乘号怎么做？
 - 枚举数字后再枚举数字前的符号（第一个数字前不枚举符号）
 - `sum`作为中间状态记录下前面的和
- 样例数据模拟

3 4 5 6 2 3 7 4 9 0

3456-23-7+490

sum=sum+490=3916

- 只有加减的时候比较好处理，那有乘号的时候怎么处理呢？
 - 枚举乘号时，sum怎么更新？

3 4 5 6 2 3 7 4 9 0

3456-23*7

枚举7之前:

sum=3456-23=3433

sum=?

- 只有加减的时候比较好处理，那有乘号的时候怎么处理呢？
 - 所以怎么办？
 - 增加一个状态lastFactor (lastF)记录下如果枚举乘号的话，当前的数应该和谁乘，也就是最后一个因子是谁

3 4 5 6 2 3 7 4 9 0

3456-23*7

sum=sum-
lastF+lastF*7=3295

- 只有加减的时候比较好处理，那有乘号的时候怎么处理呢？
 - 所以怎么办？
 - 增加一个状态`lastFactor (lastF)`记录下如果枚举乘号的话，当前的数应该和谁乘，也就是最后一个因子是谁

3 4 5 6 2 3 7 4 9 0

`lastF=0`

`sum=0`

- 只有加减的时候比较好处理，那有乘号的时候怎么处理呢？
 - 所以怎么办？
 - 增加一个状态`lastFactor (lastF)`记录下如果枚举乘号的话，当前的数应该和谁乘，也就是最后一个因子是谁

3 4 5 6 2 3 7 4 9 0

3456

lastF=3456

sum=3456

- 只有加减的时候比较好处理，那有乘号的时候怎么处理呢？
 - 所以怎么办？
 - 增加一个状态lastFactor (lastF)记录下如果枚举乘号的话，当前的数应该和谁乘，也就是最后一个因子是谁

3 4 5 6 2 3 7 4 9 0

3456-23

lastF=-23

sum=sum-23=3433

- 只有加减的时候比较好处理，那有乘号的时候怎么处理呢？
 - 所以怎么办？
 - 增加一个状态lastFactor (lastF)记录下如果枚举乘号的话，当前的数应该和谁乘，也就是最后一个因子是谁

3 4 5 6 2 3 7 4 9 0

3456-23*7

lastF=lastF*7=-161

sum=sum-
lastF+lastF*7=3295

- 只有加减的时候比较好处理，那有乘号的时候怎么处理呢？
 - 所以怎么办？
 - 增加一个状态lastFactor (lastF)记录下如果枚举乘号的话，当前的数应该和谁乘，也就是最后一个因子是谁

3 4 5 6 2 3 7 4 9 0

3456-23*7+490

lastF=490

sum=sum+490=3785

- DFS扩展情况总结：
 - 乘号时：
 - sum更新方法: $\text{sum} = \text{sum} - \text{lastF} + \text{lastF} * \text{当前枚举的数}$
 - lastF更新方法: $\text{lastF} = \text{lastF} * \text{当前枚举的数}$
 - 加号/减号时：
 - sum更新方法: $\text{sum} = \text{sum} \text{ +/- 当前枚举的数}$
 - lastF更新方法: $\text{lastF} = \text{当前枚举的数}$ （加号时为正，减号时为负）

- Company Tags: Google Facebook

考点:

- DFS枚举方式的选择
- DFS中状态记录的方法（处理乘号时的lastF）

能力维度：

2. 代码基础功力
3. 基础数据结构/算法
4. 逻辑思维/算法优化能力
5. 细节处理（corner case）

DFS类问题题目：

Palindrome Partitioning	Permutations	Permutations II
N-Queens	Combination Sum	Combination Sum II
Subsets II	Word Ladder II	

BFS类问题

Surrounded Regions

<http://www.lintcode.com/zh-cn/problem/surrounded-regions/>

<http://www.jiuzhang.com/solutions/surrounded-regions/>

分享BFS的三个比喻：

1. **BFS**就是天黑到看不见时，你手机在了地上，你会一圈一圈的找，先找离你距离为1的一个圈，然后距离为2的一个圈。。。
 - 额外的好处：当你找到手机时，同时也知道了你与手机的距离
2. 还有走迷宫时，也可以一层层的找，先距离为1的，再距离为2的
 - 额外的好处：当你找到出口时，同时也知道了起点与出口的最短距离（就是找的时候的第几层）
3. 在一个平原上有一个盆地，盆地里有一口井，井水缓慢的溢了出来，成一个圆形在盆地里扩展，**BFS**就是模拟这个扩展的过程

Surrounded Regions

思路:

- 'X' 就是平原, 'O' 就是盆地

```
X X X X
X O O X
O X X X
X O O X
```

原盆地

```
X X X X
X 1 O X
2 X X X
X 3 O X
```

开始注水的位置

```
X X X X
X 1 1 X
2 X X X
X 3 3 X
```

水流流开后的样子

- 找到所有被 'X' 围绕的区域 → 找到被平原围绕的盆地？
 - 反向思维：找到没有被平原围绕的盆地
 - 从4条边上的盆地开始注水

X	X	X	X
X	0	0	X
0	X	X	X
X	0	0	X

Surrounded Regions

- 找到所有被 'X' 围绕的区域 → 找到被平原围绕的盆地？
 - 反向思维：找到没有被平原围绕的盆地
 - 从4条边上的盆地开始注水

X	X	X	X
X	0	0	X
2	X	X	X
X	3	0	X

- 找到所有被 'X' 围绕的区域 → 找到被平原围绕的盆地？
 - 反向思维：找到没有被平原围绕的盆地
 - 从4条边上的盆地开始注水

X	X	X	X
X	0	0	X
2	X	X	X
X	3	3	X

- 找到所有被 'X' 围绕的区域 → 找到被平原围绕的盆地？
 - 反向思维：找到没有被平原围绕的盆地
 - 从4条边上的盆地开始注水
 - 除了有水的点变成 'O' 其他地方全变成 'X'

X	X	X	X
X	X	X	X
O	X	X	X
X	O	O	X

Surrounded Regions

- 怎么实现？

X	X	X	X
X	X	X	X
0	X	X	X
X	0	0	X

时间复杂度 $O(NM)$

Surrounded Regions

- 怎么实现？
- 如果我们用python实现这段代码。。。

X	X	X	X
X	X	X	X
0	X	X	X
X	0	0	X

- Company Tags: Amazon Google Facebook

考点:

- 标准的棋盘图中**BFS**问题
- 棋盘图联通部分（联通分量）标号
- 反向思维：围绕 → 没被围绕

能力维度：

2. 代码基础功力
3. 基础数据结构/算法
4. 逻辑思维/算法优化能力

◆ 小技巧总结:

- 在网格图、矩阵图、棋盘上做多个方向扩展时，用dx dy数组会让程序写起来更方便

Walls and Gates

<http://www.lintcode.com/problem/walls-and-gates/>

<http://www.jiuzhang.com/solution/walls-and-gates/>

思路:

- 问题简化: 假设只有一个门
 - 即求每个房间到这个门的最短路 (多源点单终点)

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	INF
INF	INF	INF	-1
INF	-1	INF	-1
-1	-1	INF	INF

思路:

- 问题简化: 假设只有一个门
 - 即求每个房间到这个门的最短路 (多源点单终点)
 - 等价于这个门到所有房间的最短路 (单源点多终点)

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	INF
INF	INF	INF	-1
INF	-1	INF	-1
-1	-1	INF	INF

思路:

- 因为是棋盘图，每个边的长度（权值）=1，所以最短路可以用BFS求出
 - 这一题谁是盆地，谁是平原，谁是注水点
 - 最短路 → BFS的第几层

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	INF
INF	INF	INF	-1
INF	-1	INF	-1
-1	-1	INF	INF

思路:

- 因为是棋盘图，每个边的长度（权值）=1，所以最短路可以用BFS求出
 - 这一题谁是盆地，谁是平原，谁是注水点
 - 最短路 → BFS的第几层

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	1
INF	INF	1	-1
INF	-1	INF	-1
-1	-1	INF	INF

思路:

- 因为是棋盘图，每个边的长度（权值）=1，所以最短路可以用BFS求出
 - 这一题谁是盆地，谁是平原，谁是注水点
 - 最短路 → BFS的第几层

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	1
INF	2	1	-1
INF	-1	2	-1
-1	-1	INF	INF

思路:

- 因为是棋盘图，每个边的长度（权值）=1，所以最短路可以用BFS求出
 - 这一题谁是盆地，谁是平原，谁是注水点
 - 最短路 → BFS的第几层

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	1
3	2	1	-1
INF	-1	2	-1
-1	-1	3	INF

思路:

- 因为是棋盘图，每个边的长度（权值）=1，所以最短路可以用BFS求出
 - 这一题谁是盆地，谁是平原，谁是注水点
 - 最短路 → BFS的第几层

-1 : Wall
0 : Gate
INF: Empty room

4	-1	0	1
3	2	1	-1
4	-1	2	-1
-1	-1	3	4

思路:

- 多源点单终点 → 单源点多终点, 最短路常用转化套路

-1 : Wall
0 : Gate
INF: Empty room

4	-1	0	1
3	2	1	-1
4	-1	2	-1
-1	-1	3	4

思路:

- 多个门，多源点多终点怎么办？

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	INF
INF	INF	INF	-1
INF	-1	INF	-1
0	-1	INF	INF

思路:

- 多个门，多源点多终点怎么办？
 - 多个源点同时注水，看谁流得快

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	INF
INF	INF	INF	-1
INF	-1	INF	-1
0	-1	INF	INF

思路:

- 多个门，多源点多终点怎么办？
 - 多个源点同时注水，看谁流得快

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	1
INF	INF	1	-1
1	-1	INF	-1
0	-1	INF	INF

思路:

- 多个门，多源点多终点怎么办？
 - 多个源点同时注水，看谁流得快

-1 : Wall
0 : Gate
INF: Empty room

INF	-1	0	1
2	2	1	-1
1	-1	2	-1
0	-1	INF	INF

思路:

- 多个门，多源点多终点怎么办？
 - 多个源点同时注水，看谁流得快

-1 : Wall
0 : Gate
INF: Empty room

3	-1	0	1
2	2	1	-1
1	-1	2	-1
0	-1	3	INF

思路:

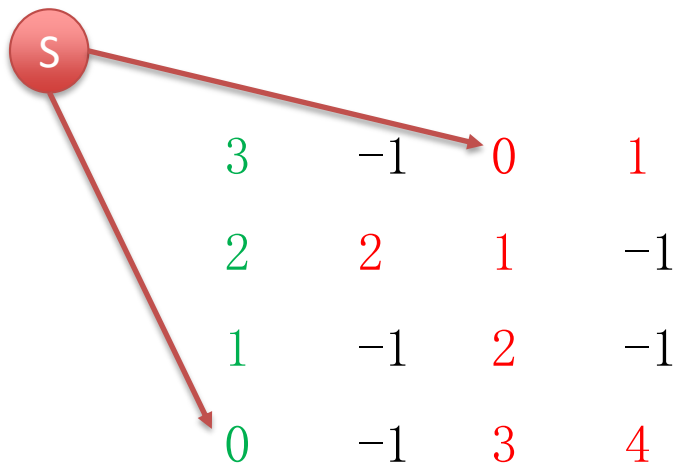
- 多个门，多源点多终点怎么办？
 - 多个源点同时注水，看谁流得快

-1 : Wall
0 : Gate
INF: Empty room

3	-1	0	1
2	2	1	-1
1	-1	2	-1
0	-1	3	4

思路:

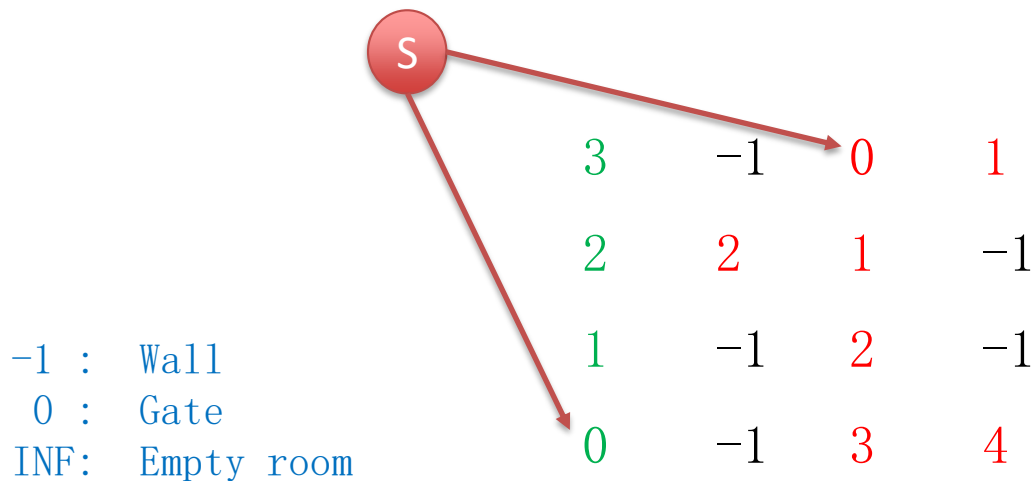
- 多源点同时注水，其实相当于增加了一个超级源点(dummy)，超级源点连接每个普通源点一条边



-1 : Wall
0 : Gate
INF: Empty room

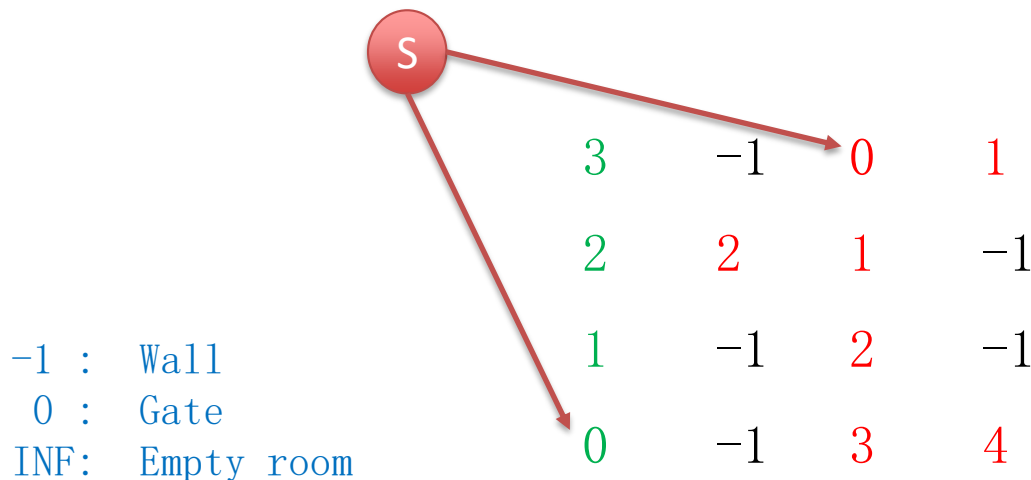
思路:

- 增加了超级源后, 其实相当于从超级源的单源最短路



思路:

- 增加了超级源后, 其实相当于从超级源的单源最短路
- 多源多终点 → 单源多终点 (增加超级源, 最短路常用转化套路)



思路:

- 细节问题: 不能达到的地方填INF怎么处理?

-1 : Wall
0 : Gate
INF: Empty room

3	-1	0	1
2	2	1	-1
1	-1	2	-1
0	-1	3	4

思路:

- 细节问题: 不能达到的地方填INF怎么处理?
 - 其实题目的输入+ **BFS** 已经帮忙处理了, **BFS**搜不到的地方就会保留题目的原始输入INF

-1 : Wall
0 : Gate
INF: Empty room

3	-1	0	1	-1
2	2	1	-1	INF
1	-1	2	-1	-1
0	-1	3	4	-1

时间复杂度 $O(NM)$

- Company Tags: Google Facebook

考点:

- 最短路问题中的一些常用转化
- 边长=1的图的最短路

能力维度：

2. 代码基础功力
3. 基础数据结构/算法
4. 逻辑思维/算法优化能力

◆ 小技巧总结:

- 多源点单终点 → 单源点多终点, 最短路常用转化套路
- 多源多终点 → 单源多终点 (增加超级源, 最短路常用转化套路)
- **BFS**可以求边长=1的图的最短路 (如此题的棋盘图)

- BFS就是如同注水的水流般一圈圈的往外搜
- BFS可以用来求连通性问题
- BFS可以求权值为1的最短路（比如棋盘上的最短路）

BFS类题目：

Clone Graph	Word Ladder	Word Ladder II
Number of Islands	Binary Tree Level Order Traversal	Search Graph Nodes
Course Schedule II	Knight Shortest Path	Zombie in Matrix
Graph Valid Tree	Binary Tree Serialization	Build Post Office II

- Letter Combinations of a Phone Number
 - 先写扩展，再写退出情况
 - 三种记录状态的方法
- Word Squares
 - DFS剪枝的方法
- Expression Add Operators

- Surrounded Regions

- BFS的三个比喻

- ◆ 小技巧总结:

- 在网格图、矩阵图、棋盘上做多个方向扩展时，用dx dy数组会让程序写起来更方便

- Walls and Gates

- ◆ 小技巧总结:

- 多源点单终点 → 单源点多终点，最短路常用转化套路

- 多源多终点 → 单源多终点 （增加超级源，最短路常用转化套路）

- BFS可以求边长=1的图的最短路（如此题的棋盘图）

快速点题

Number of Islands

- <http://lintcode.com/zh-cn/problem/number-of-islands/>
- 棋盘图上BFS统计联通分量基础题
- [Solution](#) Company Tags: Google Amazon Facebook LinkedIn

- <http://lintcode.com/zh-cn/problem/number-of-islands-ii/>
- Number of Islands 的follow up
- 题目增加了一个更改操作，每次将一个点从海洋变为岛屿，问改变后岛屿的个数
- 并查集练习题 （最小生成树小视频）
- [Solution](#) Company Tags: Facebook

- <http://lintcode.com/problem/binary-tree-vertical-order-traversal/>
- 树上的**BFS**，注意题目规定的三个访问顺序：列、行、列行相同时
- 提示：可以用**Hash**来解决访问列从小到大的顺序
- [Solution](#) Company Tags: Facebook Amazon

Binary Tree Zigzag Level Order Traversal



- <http://lintcode.com/problem/binary-tree-zigzag-level-order-traversal/>
- 树上的BFS练习题，注意zig zag的顺序
- [Solution](#) Company Tags: Google Facebook Amazon

Serialize and Deserialize Binary Tree



- <http://lintcode.com/problem/serialize-and-deserialize-binary-tree/>
 - 树上的BFS
 - 想想怎样将一棵二叉树编码和解码
 - [Solution](#)
- Company Tags: Google Facebook Amazon LinkedIn

- <http://lintcode.com/zh-cn/problem/factorization/>

- DFS练习题，注意在扩展的时候可以剪枝

- [Solution](#)

Company Tags: LinkedIn



扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: [ninechapter](#)

知乎专栏: <http://zhuankan.zhihu.com/jiuzhang>

微博: <http://www.weibo.com/ninechapter>

官网: www.jiuzhang.com