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

知识点回顾

- 二分查找时的写法，求mid不溢出：
  A. mid = (l + r) / 2　B. mid = l + (r - l) / 2

- BST的中根(Inorder)遍历，节点访问顺序是什么样的?

- BST上的操作和哪一种算法比较相似？

- Binary Tree Vertical Order Traversal 这题中我们使用什么算法来一层层的遍历？

概要

- BFS类问题（2题）
- DFS类问题（4题）

BFS类问题

Surrounded Regions

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

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

Surrounded Regions

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

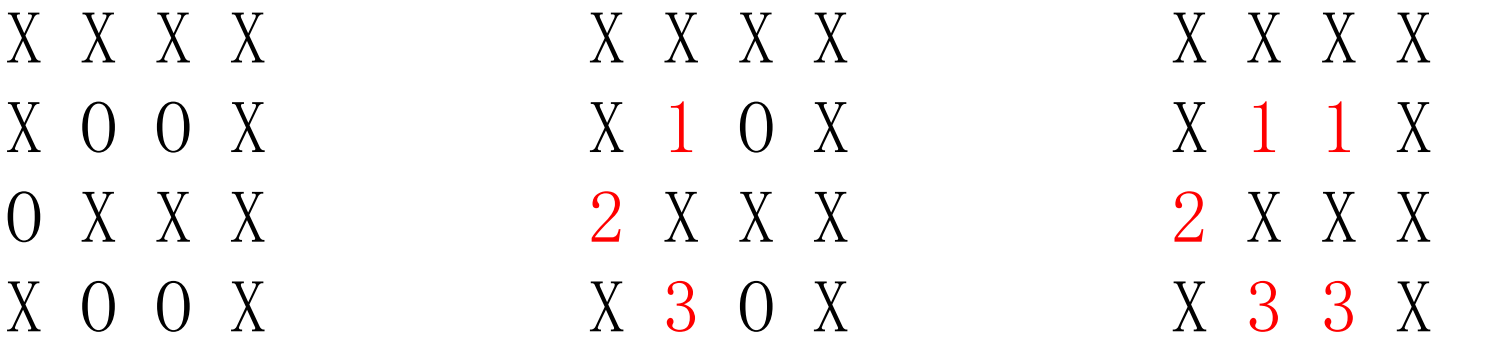
2. 还有走迷宫时，也可以一层层的找，先距离为1的，再距离为2的
   - 额外的好处：当你找到出口时，同时也知道了起点与出口的最短距离（就是找的时候的第几层）

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



原盆地　　开始注水的位置　　水流流开后的样子

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 O O X
O X X X
X O O 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 O O X
2 X X X
X 3 O 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 O O X
2 X X X
X 3 3 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
  - 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
O X X X
X O O X

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
O X X X
X O O X

Surrounded Regions

- 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

Surrounded Regions

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

Surrounded Regions

◆ 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

Nearest Exit

http://www.lintcode.com/zh-cn/problem/nearest-exit/

http://www.jiuzhang.com/solutions/nearest-exit/

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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
```

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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

Nearest Exit

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

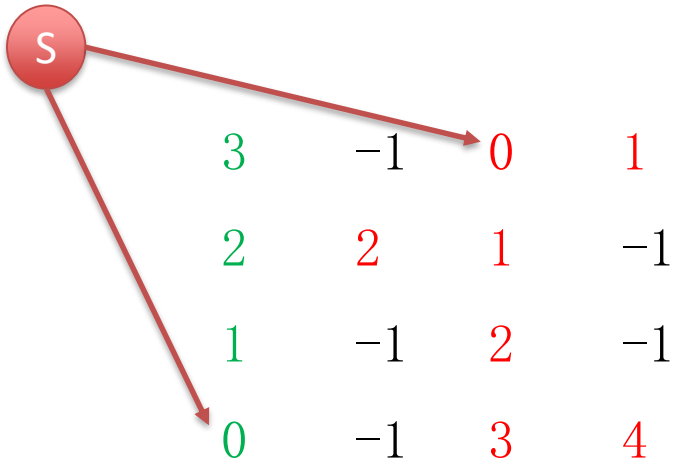
```
-1 :  Wall
 0 :  Gate
INF:  Empty room
```

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

Nearest Exit

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

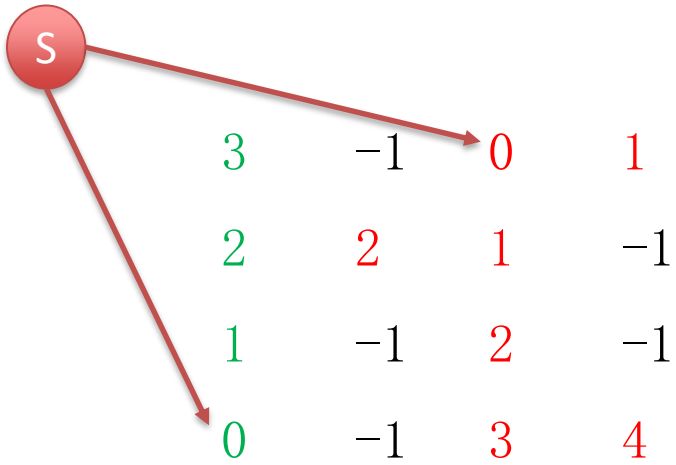



```
-1 :  Wall
 0 :  Gate
INF:  Empty room
```

Nearest Exit

思路：

- 增加了超级源后，其实相当于从超级源的单源最短路

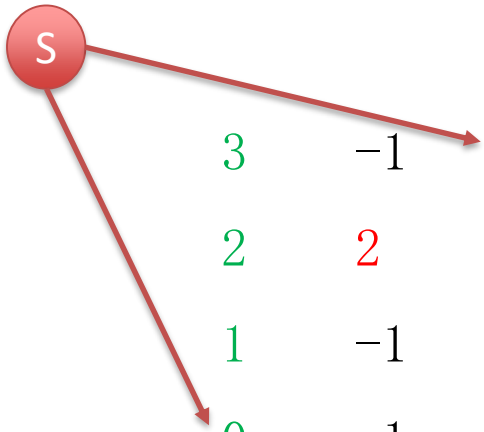


-1 : Wall
0 : Gate
INF: Empty room

Nearest Exit

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）



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

-1 : Wall
0 : Gate
INF: Empty room

Nearest Exit

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

Nearest Exit

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

Nearest Exit

- Company Tags：Google Facebook

考点：

- 最短路问题中的一些常用转化
- BFS求边长=1的图的最短路（如此题的棋盘图）

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

Nearest Exit

◆ 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类问题总结

- 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类问题总结

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

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/

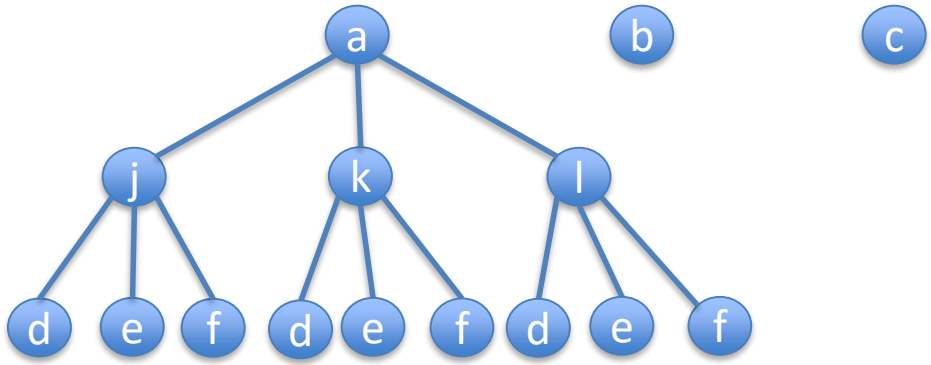
Letter Combinations of a Phone Number

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



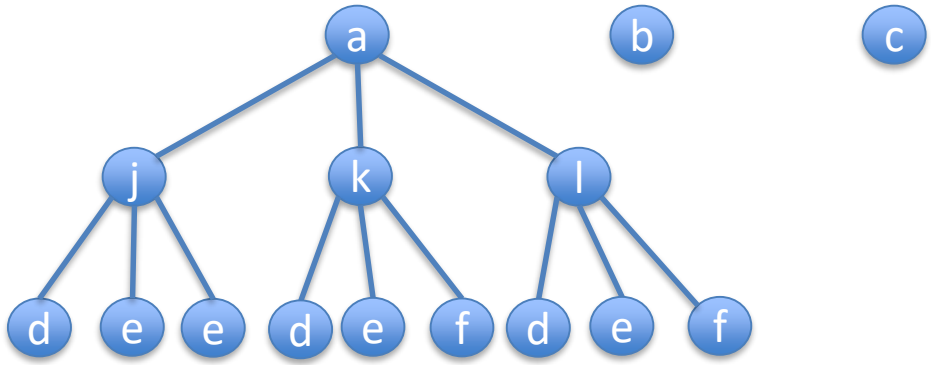
Letter Combinations of a Phone Number

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



Letter Combinations of a Phone Number

- 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：可以放在成员变量中，在DFS时滚动，比如这题的str（建议根据具体需要来，一般来说用 1 的方法放在参数中会更方便）

Letter Combinations of a Phone Number

- Company Tags: Amazon Google Facebook

考点：

- 基础DFS的快速实现
- 电面题

Letter Combinations of a Phone Number

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）

Factorization

http://www.lintcode.com/zh-cn/problem/factorization/

http://www.jiuzhang.com/solutions/factorization/

Factorization

思路：

- DFS练手题
  - 枚举每一个位置的因子
  - 最后一个因子remain 不用枚举，可以通过 remain=num / factor1/ factor2 …..factorN 算出来

- 这样保证因子顺序从小到大？
  - 记录下lastFactor（简写lastF） 上一阶段枚举的因子，本阶段从lastF开始枚举
  - factor 小于 remain
  - 即factor1 <= factor2 <= factor3 ….. factorN <= remain

Factorization

- 用什么方法记录状态
  - LastF remain采用方法1 放到DFS函数的参数中
  - item采用方法3 全局数组+滚动来记录状态
  - （当然也可以用1号方法把item放到参数中 改动5行）

Factorization

- Company Tag: LinkedIn

考点：

- 基础DFS的快速实现

Factorization

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）

Add Operators

http://www.lintcode.com/zh-cn/problem/add-operators/

http://www.jiuzhang.com/solutions/add-operators/

Add Operators

思路：

- 简化版，只有加减没有乘号怎么做？
  - 枚举数字后再枚举符号
  - sum作为中间状态记录下前面的和

- 样例数据模拟
  - 3456237490
  - 3456 -23 -74 +90

Add Operators

- 只有加减的时候比较好处理，那么怎么处理乘号呢？
  - 再记录一个状态，最后的连乘的那个数 lastFactor
    - 34-56*23*74　90　枚举完34562374时，lastFactor = -56 * 23 * 74
    - 34-562-374　90　枚举完34562374时，lastFactor = -374
  - 乘号时：
    - sum更新方法：sum = sum – lastFactor + lastFactor * 当前枚举的数
    - lastFactor更新方法：lastFactor = lastFactor * 当前枚举的数
  - 加号/减号时：
    - sum更新方法：sum = sum +/- 当前枚举的数
    - lastFactor更新方法：lastFactor = 当前枚举的数（加号时为正，减号时为负）

Add Operators

- 有哪些情况需要特殊判断？
  - 第一个数字前不能有符号　　（+3456 -23 -74 +90 错误）
  - 数字不能有前导0　　（ 2543+034　　错误）

- 这一题我的写法中是怎么记录状态的？
  - pos str sum lastFactor 采用方法1放在DFS函数参数中
  - num target 不变的值采用方法2放到全局变量中

Add Operators

- Company Tags: Google Facebook

考点：

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

Add Operators

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）

Word Squares

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

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

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

Word Squares

- 所以怎办？
  - 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

Word Squares

- 这一题有哪些冗余？
- 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
  - 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

Word Squares

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

- 见代码

Word Squares

- 怎么记录状态的？
  - l 第几层用的方法1放在DFS的函数参数中
  - wordLen 不改变的采用方法2 放在全局变量中
  - squares 采用方法3放在全局变量中，并在DFS时滚动
  - pre 前缀不好记录，在DFS过程中临时算的

Word Squares

- Company Tags: Google

- 考点：
- DFS过程中的剪枝
- Hash Trie树的应用

Word Squares

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类问题总结

DFS类问题题目：

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

本节课知识点总结

- 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

- Nearest Exit
  - ◆ 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）

本节课知识点总结

- 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

- Factorization
- Add Operators
- Word Squares
  - DFS剪枝的方法

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