



qinzhaokun  
码龄8年

27  
原创

1346  
积分

TA的主页

私信

关注

### 最新文章

蚂蚁金服旗下网商银行招聘了

匈牙利算法

SQL体系结构

MySQL优化20条经验

MySQL 索引

### 分类专栏

## 拓扑排序的两种实现：Kahn算法和dfs算法

转载

qinzhaokun

2015-09-18 08:20:28

13628

收藏 8

展开

本文将从以下几个方面介绍拓扑排序：

- 拓扑排序的定义和前置条件
- 和离散数学中偏序/全序概念的联系
- 典型实现算法
  - *Kahn*算法
  - 基于*DFS*的算法
- 解的唯一性问题
- 实际例子

取材自以下材料：

[http://en.wikipedia.org/wiki/Topological\\_sorting](http://en.wikipedia.org/wiki/Topological_sorting)

[http://en.wikipedia.org/wiki/Hamiltonian\\_path](http://en.wikipedia.org/wiki/Hamiltonian_path)

### 定义和前置条件：

定义：将有向图中的顶点以线性方式进行排序。即对于任何连接自顶点 $u$ 到顶点 $v$ 的有向边 $uv$ ，在最后的排序结果中，顶点 $u$ 总是在顶点 $v$ 的前面。

如果这个概念还略显抽象的话，那么不妨考虑一个非常非常经典的例子——选课。我想任何看过数据结构相关书籍的同学都知道它吧。假设我非常想学习一门机器学习的课程，但是在修这么课程之前，我们必须先学习一些基础课程，比如计算机科学概论，C语言程序设计，数据结构，算法等等。那么这个制定选修课程顺序的过程，实际上就是一个拓扑排序的过程，每门课程相当于有向图中的一个顶点，而连接顶点之间的有向边就是课程学习的先后关系。只不过这个过程不是那么复杂，自然而然就在我们的头脑中完成了。将这个学习过程以算法的形式描述出来的结果，就是拓扑排序。



已赞 15



评论 7



分享



收藏 8



手机看

文章举报

收起全文 ^

那么是不是所有的有向图都能够被拓扑排序呢？显然不是。继续考虑上面的例子，如果告诉你在选修计算机科学概论这门课之前需要你先学习机器学习，你是不是会被弄糊涂？在这种情况下，就无法进行拓扑排序，因为它中间存在互相依赖的关系，从而无法确定谁先谁后。在有向图中，这种情况被描述为存在环路。因此，一个有向图能被拓扑排序的充要条件就是它是一个有向无环图(DAG: *Directed Acyclic Graph*)。

### 偏序/全序关系：

偏序和全序实际上是离散数学中的概念。

这里不打算说太多形式化的定义，形式化的定义教科书上或者上面给的链接中就说的很详细。

还是以上面选课的例子来描述这两个概念。假设我们在学习完了算法这门课后，可以选修机器学习或者计算机图形学。这个或者表示，学习机器学习和计算机图形学这两门课之间没有特定的先后顺序。因此，在我们所有可以选择的课程中，任意两门课程之间的关系要么是确定的(即拥有先后关系)，要么是不确定的(即没有先后关系)，绝对不存在互相矛盾的关系(即环路)。以上就是偏序的意义，抽象而言，有向图中两个顶点之间不存在环路，至于连通与否，是无关紧要的。所以，有向无环图必然是满足偏序关系的。

理解了偏序的概念，那么全序就好办了。所谓全序，就是在偏序的基础之上，有向无环图中的任意一对顶点还需要有明确的关系(反映在图中，就是单向连通的关系，注意不能双向连通，那就成环了)。可见，全序就是偏序的一种特殊情况。回到我们的选课例子中，如果机器学习需要在学习



归档

2018

3月1篇

2016

6月1篇

5月4篇

4月1篇

3月6篇

2015

12月2篇

11月3篇

10月6篇

9月9篇

热门文章

spark安装与使用（入门）13804

拓扑排序的两种实现：Kahn算法和dfs算法13614

Linux下安装Hadoop6576

logstash安装和使用4244

Kafka原理及应用3753

最新评论

拓扑排序的两种实现：Kahn算法和...  
hanguangchuan：谢谢作者，写得真棒

拓扑排序的两种实现：Kahn算法和...  
hanguangchuan：‘只要当前顶点还存在边指向其它任何顶点，它就会递归调用dfs方法，而不会 ...

拓扑排序的两种实现：Kahn算法和...  
qq\_28959087：写的真棒

拓扑排序的两种实现：Kahn算法和...  
weixin\_42419701：很受用，谢谢

算法学习--动态规划  
weixin\_43584808：这两种方法都能存储子问题解决方案。在第一个版本中，记忆化存储只在查找 ...

了计算机图形学之后才能学习(可能学的是图形学领域相关的机器学习算法.....)，那么它们之间也就存在了确定的先后顺序，原本的偏序关系就变成了全序关系。

实际上，很多地方都存在偏序和全序的概念。

比如对若干互不相等的整数进行排序，最后总是能够得到唯一的排序结果(从小到大，下同)。这个结论应该不会有人表示疑问吧:)但是如果我们以偏序/全序的角度来考虑一下这个再自然不过的问题，可能就会有别的体会了。

那么如何用偏序/全序来解释排序结果的唯一性呢？

我们知道不同整数之间的大小关系是确定的，即 $i$ 总是小于 $4$ 的，不会有人说 $i$ 大于或者等于 $4$ 吧。这就是说，这个序列是满足全序关系的。而对于拥有全序关系的结构(如拥有不同整数的数组)，在其线性化(排序)之后的结果必然是唯一的。对于排序的算法，我们评价指标之一是看该排序算法是否稳定，即值相同的元素的排序结果是否和出现的顺序一致。比如，我们说快速排序是不稳定的，这是因为最后的快排结果中相同元素的出现顺序和排序前不一致了。如果用偏序的概念可以这样解释这一现象：**相同值的元素之间的关系是无法确定的**。因此它们在最终的结果中的出现顺序可以是任意的。而对于诸如插入排序这种稳定性排序，它们对于值相同的元素，还有一个潜在的比较方式，即比较它们的出现顺序，出现靠前的元素大于出现后出现的元素。因此通过这一潜在的比较，将偏序关系转换为了全序关系，从而保证了结果的唯一性。

拓展到拓扑排序中，结果具有唯一性的条件也是其所有顶点之间都具有全序关系。如果没有这一层全序关系，那么拓扑排序的结果也就不是唯一的了。在后面会谈到，如果拓扑排序的结果唯一，那么该拓扑排序的结果同时也代表了一条哈密顿路径。

### 典型实现算法：

**Kahn**算法：

摘一段维基百科上关于**Kahn**算法的伪码描述：

$L \leftarrow$  Empty list that will contain the sorted elements

$S \leftarrow$  Set of all nodes with no incoming edges

**while**  $S$  is non-empty **do**

    remove a node  $n$  from  $S$

    insert  $n$  into  $L$

**foreach** node  $m$  with an edge  $e$  from  $n$  to  $m$  **do**

        remove edge  $e$  from thegraph

**if**  $m$  has no other incoming edges **then**

            insert  $m$  into  $S$

**if** graph has edges **then**

    return error (graph has at least onecycle)

**else**

    return  $L$  (a topologically sortedorder)

不难看出该算法的实现十分直观，关键在于需要维护一个入度为 $0$ 的顶点的集合：

每次从该集合中取出(没有特殊的取出规则，随机取出也行，使用队列/栈也行，下同)一个顶点，将该顶点放入保存结果的 $List$ 中。

紧接着循环遍历由该顶点引出的所有边，从图中移除这条边，同时获取该边的另外一个顶点，如果该顶点的入度在减去本条边之后为 $0$ ，那么也将这个顶点放到入度为 $0$ 的集合中。然后继续从集合中取出一个顶点.....

当集合为空之后，检查图中是否还存在任何边，如果存在的话，说明图中至少存在一条环路。不存在的话则返回结果 $List$ ，此 $List$ 中的顺序就是对图进行拓扑排序的结果。

实现代码：

QQ客服            kefu@csdn.net  
客服论坛            400-660-0108

工作时间 8:30-22:00

关于我们 | 招聘 | 广告服务 | 网站地图

京ICP备19004658号 经营性网站备案信息

 公安备案号 11010502030143

京网文〔2020〕1039-165号

©1999-2020 北京创新乐知网络技术有限公司

网络110报警服务

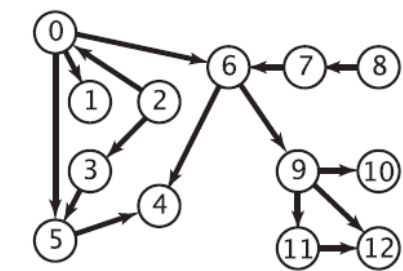
北京互联网违法和不良信息举报中心

中国互联网举报中心 家长监护

版权与免责声明 版权申诉

```
[java]    
01. public class KahnTopological  
02. {  
03.     private List<Integer> result; // 用来存储结果集  
04.     private Queue<Integer> setOfZeroIndegree; // 用来存储入度为0的顶点  
05.     private int[] indegrees; // 记录每个顶点当前的入度  
06.     private int edges;  
07.     private Digraph di;  
08.  
09.     public KahnTopological(Digraph di)  
10.     {  
11.         this.di = di;  
12.         this.edges = di.getE();  
13.         this.inderees = new int[di.getV()];  
14.         this.result = new ArrayList<Integer>();  
15.         this.setOfZeroIndegree = new LinkedList<Integer>();  
16.  
17.         // 对入度为0的集合进行初始化  
18.         Iterable<Integer>[] adjs = di.getAdj();  
19.         for(int i = 0; i < adjs.length; i++)  
20.         {  
21.             // 对每一条边 v -> w  
22.             for(int w : adjs[i])  
23.             {  
24.                 indegrees[w]++;  
25.             }  
26.         }  
27.  
28.         for(int i = 0; i < indegrees.length; i++)  
29.         {  
30.             if(0 == indegrees[i])  
31.             {  
32.                 setOfZeroIndegree.enqueue(i);  
33.             }  
34.         }  
35.         process();  
36.     }  
37.  
38.     private void process()  
39.     {  
40.         while(!setOfZeroIndegree.isEmpty())  
41.         {  
42.             int v = setOfZeroIndegree.dequeue();  
43.  
44.             // 将当前顶点添加到结果集中  
45.             result.add(v);  
46.  
47.             // 遍历由v引出的所有边  
48.             for(int w : di.adj(v))  
49.             {  
50.                 // 将该边从图中移除，通过减少边的数量来表示  
51.                 edges--;  
52.                 if(0 == --indegrees[w]) // 如果入度为0，那么加入入度为0的集合  
53.                 {  
54.                     setOfZeroIndegree.enqueue(w);  
55.                 }  
56.             }  
57.         }  
58.         // 如果此时图中还存在边，那么说明图中含有环路  
59.         if(0 != edges)  
60.         {  
61.             throw new IllegalArgumentException("Has Cycle !");  
62.         }  
63.     }  
64. }
```

```
62.     }
63. }
64.
65. public Iterable<Integer> getResult()
66. {
67.     return result;
68. }
69. }
```



对上图进行拓扑排序的结果：  
2->8->0->3->7->1->5->6->9->4->11->10->12

复杂度分析：  
初始化入度为0的集合需要遍历整张图，检查每个节点和每条边，因此复杂度为 $O(E+V)$ ；  
然后对该集合进行操作，又需要遍历整张图中的，每条边，复杂度也为 $O(E+V)$ ；  
因此Kahn算法的复杂度即为 $O(E+V)$ 。

基于DFS的拓扑排序：

除了使用上面直观的Kahn算法之外，还能够借助深度优先遍历来实现拓扑排序。这个时候需要使用到栈结构来记录拓扑排序的结果。

同样摘录一段维基百科上的伪码：

L ← Empty list that will contain the sorted nodes

S ← Set of all nodes with no outgoing edges

```
for each node n in S do
    visit(n)
function visit(node n)
    if n has not been visited yet then
        mark n as visited
        for each node m with an edge from n to m do
            visit(m)
        add n to L
```

DFS的实现更加简单直观，使用递归实现。利用DFS实现拓扑排序，实际上只需要添加一行代码，即上面伪码中的最后一行：add n to L。

需要注意的是，将顶点添加到结果List中的时机是在visit方法即将退出之时。

这个算法的实现非常简单，但是要理解的话就相对复杂一点。

关键在于为什么在visit方法的最后将该顶点添加到一个集合中，就能保证这个集合就是拓扑排序的结果呢？

因为添加顶点到集合中的时机是在dfs方法即将退出之时，而dfs方法本身是个递归方法，只要当前顶点还存在边指向其它任何顶点，它就会递归调

用dfs方法，而不会退出。因此，退出dfs方法，意味着当前顶点没有指向其它顶点的边了，即当前顶点是一条路径上的最后一个顶点。

下面简单证明一下它的正确性：

考虑任意的边v->w，当调用dfs(v)的时候，有如下三种情况：

1. dfs(w)还没有被调用，即w还没有被mark，此时会调用dfs(w)，然后当dfs(w)返回之后，dfs(v)才会返回
2. dfs(w)已经被调用并返回了，即w已经被mark
3. dfs(w)已经被调用但是在此时调用dfs(v)的时候还未返回

需要注意的是，以上第三种情况在拓扑排序的场景下是不可能发生的，因为如果情况3是合法的话，就表示存在一条由w到v的路径。而现在我们的前提条件是由v到w有一条边，这就导致我们的图中存在环路，从而该图就不是一个有向无环图(DAG)，而我们已经知道，非有向无环图是不能被拓扑排序的。

那么考虑前两种情况，无论是情况1还是情况2，w都会先于v被添加到结果列表中。所以边v->w总是由结果集中后出现的顶点指向先出现的顶点。为了让结果更自然一些，可以使用栈来作为存储最终结果的数据结构，从而能够保证边v->w总是由结果集中先出现的顶点指向后出现的顶点。

实现代码：

```
[java]
01. public class DirectedDepthFirstOrder
02. {
03.     // visited数组，DFS实现需要用到
04.     private boolean[] visited;
05.     // 使用栈来保存最后的结果
06.     private Stack<Integer> reversePost;
07.
08.     /**
09.      * Topological Sorting Constructor
10.      */
11.     public DirectedDepthFirstOrder(Digraph di, boolean detectCycle)
12.     {
13.         // 这里的DirectedDepthFirstCycleDetection是一个用于检测有向图中是否存在环路的类
14.         DirectedDepthFirstCycleDetection detect = new DirectedDepthFirstCycleDetection(
15.             di);
16.
17.         if (detectCycle && detect.hasCycle())
18.             throw new IllegalArgumentException("Has cycle");
19.
20.         this.visited = new boolean[di.getV()];
21.         this.reversePost = new Stack<Integer>();
22.
23.         for (int i = 0; i < di.getV(); i++)
24.         {
25.             if (!visited[i])
26.             {
27.                 dfs(di, i);
28.             }
29.         }
30.     }
31.
32.     private void dfs(Digraph di, int v)
33.     {
34.         visited[v] = true;
35.
36.         for (int w : di.adj(v))
37.         {
38.             if (!visited[w])
39.             {
```

```
40.         dfs(di, w);
41.     }
42. }
43.
44. // 在即将退出dfs方法的时候，将当前顶点添加到结果集中
45. reversePost.push(v);
46. }
47.
48. public Iterable<Integer> getReversePost()
49. {
50.     return reversePost;
51. }
52. }
```

复杂度分析：  
复杂度同DFS一致，即 $O(E+V)$ 。具体而言，首先需要保证图是有向无环图，判断图是DAG可以使用基于DFS的算法，复杂度为 $O(E+V)$ ，而后面的拓扑排序也是依赖于DFS，复杂度为 $O(E+V)$

还是对上文中的那张有向图进行拓扑排序，只不过这次使用的是基于DFS的算法，结果是：  
8->7->2->3->0->6->9->10->11->12->1->5->4

**两种实现算法的总结：**  
这两种算法分别使用链表和栈来表示结果集。  
对于基于DFS的算法，加入结果集的条件是：顶点的出度为0。这个条件和Kahn算法中入度为0的顶点集合似乎有着异曲同工之妙，这两种算法的思想犹如一枚硬币的两面，看似矛盾，实则不然。一个是从入度的角度来构造结果集，另一个则是从出度的角度来构造。

实现上的一些不同之处：  
Kahn算法不需要检测图为DAG，如果图为DAG，那么在出度为0的集合为空之后，图中还存在没有被移除的边，这就说明了图中存在环路。而基于DFS的算法需要首先确定图为DAG，当然也能够做出适当调整，让环路的检测和拓扑排序同时进行，毕竟环路检测也能够基于DFS的基础上进行。二者的复杂度均为 $O(V+E)$ 。

环路检测和拓扑排序同时进行的实现：

```
[java]
01. public class DirectedDepthFirstTopoWithCircleDetection
02. {
03.     private boolean[] visited;
04.     // 用于记录dfs方法的调用栈，用于环路检测
05.     private boolean[] onStack;
06.     // 用于当环路存在时构造之
07.     private int[] edgeTo;
08.     private Stack<Integer> reversePost;
09.     private Stack<Integer> cycle;
10.
11.     /**
12.      * Topological Sorting Constructor
13.      */
14.     public DirectedDepthFirstTopoWithCircleDetection(Digraph di)
15.     {
16.         this.visited = new boolean[di.getV()];
17.         this.onStack = new boolean[di.getV()];
18.         this.edgeTo = new int[di.getV()];
19.         this.reversePost = new Stack<Integer>();
```

```
20.
21.     for (int i = 0; i < di.getV(); i++)
22.     {
23.         if (!visited[i])
24.         {
25.             dfs(di, i);
26.         }
27.     }
28. }
29.
30. private void dfs(Digraph di, int v)
31. {
32.     visited[v] = true;
33.     // 在调用dfs方法时，将当前顶点记录到调用栈中
34.     onStack[v] = true;
35.
36.     for (int w : di.adj(v))
37.     {
38.         if (hasCycle())
39.         {
40.             return;
41.         }
42.         if (!visited[w])
43.         {
44.             edgeTo[w] = v;
45.             dfs(di, w);
46.         }
47.         else if (onStack[w])
48.         {
49.             // 当w已经被访问，同时w也存在于调用栈中时，即存在环路
50.             cycle = new Stack<Integer>();
51.             cycle.push(w);
52.             for(int start = v; start != w; start = edgeTo[start])
53.             {
54.                 cycle.push(v);
55.             }
56.             cycle.push(w);
57.         }
58.     }
59.
60.     // 在即将退出dfs方法时，将顶点添加到拓扑排序结果集中，同时从调用栈中退出
61.     reversePost.push(v);
62.     onStack[v] = false;
63. }
64.
65. private boolean hasCycle()
66. {
67.     return (null != cycle);
68. }
69.
70. public Iterable<Integer> getReversePost()
71. {
72.     if (!hasCycle())
73.     {
74.         return reversePost;
75.     }
76.     else
77.     {
78.         throw new IllegalArgumentException("Has Cycle: " + getCycle());
79.     }
80. }
81.
82. public Iterable<Integer> getCycle()
83. {
```

```
84.         return cycle;
85.     }
86. }
```

## 拓扑排序解的唯一性：

### 哈密顿路径：

哈密顿路径是指一条能够对图中所有顶点正好访问一次的路径。本文中只会解释一些哈密顿路径和拓扑排序的关系，至于哈密顿路径的具体定义以及应用，可以参见本文开篇给出的链接。

前面说过，当一个 $DAG$ 中的任何两个顶点之间都存在可以确定的先后关系时，对该 $DAG$ 进行拓扑排序的解是唯一的。这是因为它们形成了全序的关系，而对存在全序关系的结构进行线性化之后的结果必然是唯一的(比如对一批整数使用稳定的排序算法进行排序的结果必然就是唯一的)。

需要注意的是，非 $DAG$ 也是能够含有哈密顿路径的，为了利用拓扑排序来实现判断，所以这里讨论的主要是判断 $DAG$ 中是否含有哈密顿路径的算法，因此下文中的图指代的都是 $DAG$ 。

那么知道了哈密顿路径和拓扑排序的关系，我们如何快速检测一张图是否存在哈密顿路径呢？

根据前面的讨论，是否存在哈密顿路径的关键，就是确定图中的顶点是否存在全序的关系，而全序的关键，就是任意一对顶点之间都是能够确定先后关系的。因此，我们能够设计一个算法，用来遍历顶点集中的每一对顶点，然后检查它们之间是否存在先后关系，如果所有的顶点对有先后关系，那么该图的顶点集就存在全序关系，即图中存在哈密顿路径。

但是很显然，这样的算法十分低效。对于大规模的顶点集，是无法应用这种解决方案的。通常一个低效的解决办法，十有八九是因为没有抓住现有问题的一些特征而导致的。因此我们回过头来再看看这个问题，有什么特征使我们没有利用的。还是举对整数进行排序的例子：

比如现在有 $3, 2, 1$ 三个整数，我们要对它们进行排序，按照之前的思想，我们分别对 $(1,2), (2,3), (1,3)$ 进行比较，这样需要三次比较，但是我们很清楚， $1$ 和 $3$ 的那次比较实际上是多余的。我们为什么知道这次比较是多余的呢？我认为，是我们下意识的利用了整数比较满足传递性的这一规则。但是计算机是无法下意识的使用传递性的，因此只能通过其它的方式来告诉计算机，有一些比较是不必要的。所以，也就有了相对插入排序，选择排序更加高效的排序算法，比如归并排序，快速排序等，将 $n^2$ 的算法加速到了 $n\log n$ 。或者是利用了问题的特点，采取了更加独特的解决方案，比如基数排序等。

扯远了一点，回到正题。现在我们没有利用到的就是全序关系中传递性这一规则。如何利用它呢，最简单的想法往往就是最实用的，我们还是选择排序，排序后对每对相邻元素进行检测不就间接利用了传递性这一规则嘛？所以，我们先使用拓扑排序对图中的顶点进行排序。排序后，对每对相邻顶点进行检测，看看是否存在先后关系，如果每对相邻顶点都存在着一致的先后关系(在有向图中，这种先后关系以有向边的形式体现，即查看相邻顶点对之间是否存在有向边)。那么就可以确定该图中存在哈密顿路径了，反之则不存在。

实现代码：

```
[java]
01.  /**
02.   * Hamilton Path Detection for DAG
03.   */
04.  public class DAGHamiltonPath
05.  {
06.      private boolean hamiltonPathPresent;
07.      private Digraph di;
08.      private KahnTopological kts;
09.
10.      // 这里使用Kahn算法进行拓扑排序
11.      public DAGHamiltonPath(Digraph di, KahnTopological kts)
```



```
12. {
13.     this.di = di;
14.     this.kts = kts;
15.
16.     process();
17. }
18.
19. private void process()
20. {
21.     Integer[] topoResult = kts.getResultAsArray();
22.
23.     // 依次检查每一对相邻顶点，如果二者之间没有路径，则不存在哈密顿路径
24.     for(int i = 0; i < topoResult.length - 1; i++)
25.     {
26.         if(!hasPath(topoResult[i], topoResult[i + 1]))
27.         {
28.             hamiltonPathPresent = false;
29.             return;
30.         }
31.     }
32.     hamiltonPathPresent = true;
33. }
34.
35. private boolean hasPath(int start, int end)
36. {
37.     for(int w : di.adj(start))
38.     {
39.         if(w == end)
40.         {
41.             return true;
42.         }
43.     }
44.     return false;
45. }
46.
47. public boolean hasHamiltonPath()
48. {
49.     return hamiltonPathPresent;
50. }
51. }
```

实际例子：

TestNG中循环依赖的检测：

[http://blog.csdn.net/dm\\_vincent/article/details/7631916](http://blog.csdn.net/dm_vincent/article/details/7631916)



qinzhaokun



原创文章 27 获赞 70 访问量 10万+

关注

私信



hanguangchuan: 谢谢作者，写得真棒



hanguangchuan:



“只要当前顶点还存在边指向其它任何顶点，它就会递归调用dfs方法，而不会退出。”是不是应该改为“只要当前顶点还存在边指向其它任何顶点（除去已访问的顶点），它就会递归调用dfs方法，而不会退出。”

 jet_qi: 写的真棒 1年前	
 weixin_42419701: 很受用，谢谢 1年前	
 NairoJ: 写的太好了，十分感谢 1年前	
 胆识与智慧: 重要的是算法，不是语言 1年前	
 梦不灭: 代码能用C/C++实现吗 2年前	
<div>【算法设计与数据结构】拓扑排序算法的实现——Kahn算法及基于dfs的算法</div> <div>拓扑排序的定义和原理等我不再赘述，各种教材和网络上都有详细解释，今天我主要谈一谈两</div>	jiange_zh的博客 3988
<div>Kahn算法-拓扑排序</div> <div>/*时间：2015/11/22内容：拓扑排序的实现（可用于判断图中是否有环路）概述：基于Kahn算法</div>	I'm noob.... 1388
<div>史上最全的IDEA快捷键总结</div> <div>现在Idea成了主流开发工具，这篇博客对其使用的快捷键做了总结，希望对大家的开发工作有</div>	扬帆向海的博客 15万+
<div>拓扑排序(Kahn算法和基于DFS求解法)</div> <div>拓扑排序是对有向无环图(DAG)进行排序，从而找到一个序列。该序列满足对于任意一对不同的</div>	yo_bc的博客 2129
<div>拓扑排序模板（Kahn算法和DFS实现）</div> <div>拓扑排序思想：每次取出入度为0的顶点删掉，并删掉和该点有关的边，需要维护一个入度</div>	baodream的博客 620
<div>10 个最难回答的 Java 问题</div> <div>1.为什么等待和通知是在 Object 类而不是 Thread 中声明的？一个棘手的 Java 问题，如果</div>	aaa13268的博客 5万+
<div>“程序员数学不行，干啥都不行！”高级开发：90%都是瞎努力！</div> <div>点击上方“Python大本营”，选择“置顶公众号”Python大本营 IT人的职业提升平台之前有很多读</div>	Python大本营的博客 2万+
<div>为什么猝死的都是程序员，基本上不见产品经理猝死呢？</div> <div>相信大家时不时听到程序员猝死的消息，但是基本上听不到产品经理猝死的消息，这是为什么</div>	曹银飞的专栏 25万+
<div>拓扑排序【kahn算法及dfs的拓扑排序】</div> <div>家谱树有个人的家族很大，辈分关系很混乱，请你帮整理一下这种关系。给出每个人的孩子的</div>	fl_334正经的码棚 182
<div>• 拓扑排序【kahn算法及dfs的拓扑排序】 - fl_334正经的..._CSDN博客</div>	11-2
<div>• 拓扑排序-Kahn算法 - 哇-WA 的博客 - CSDN博客</div>	10-16
<div>500行代码，教你用python写个微信飞机大战</div> <div>这几天在重温微信小游戏的飞机大战，玩着玩着就在思考人生了，这飞机大战怎么就可以做的</div>	Python专栏 8万+
<div>• 拓扑排序(topological sorting)时间复杂度 - This is b..._CSDN博客</div>	11-14

<div><div></div><div>•</div></div> <div>拓扑排序——用C++中STL实现 - 疯狂的指针的博客 - CSDN博客</div>	11-21
<div><div></div><div>毕业5年，我问遍了身边的大佬，总结了他们的学习方法 我问了身边10个大佬，总结了他们的学习方法，原来成功都是有迹可循的。</div></div>	<div>敖丙</div> <div>27万+</div>
<div><div></div><div>Synchronized关键字深析（小白慎入，深入jvm源码，两万字长文） 从jvm层面解析synchronized，看完绝对可以超越绝大多数人</div></div>	<div>Java新生代</div> <div>2万+</div>
<div><div></div><div>•</div></div> <div>例题6-15 拓扑排序 DFS实现和STL实现 uva10305 - CSDN博客</div>	7-16
<div><div></div><div>•</div></div> <div>Uva 10305 Ordering Tasks (用dfs 实现拓扑排序) - PK..._CSDN博客</div>	11-18
<div><div></div><div>dfs拓扑排序原理详解-----还不明白请来砍我 我们不创造算法，我们只是算法的搬运工个人笔记之核心点：@1：dfs在实现时不仅访问了顶</div></div>	<div>qq_34384524的博客</div> <div>2246</div>
<div><div></div><div>20道你必须要背会的微服务面试题，面试一定会被问到 这篇博客总结了面试中最常见的微服务面试题，相信对你有所帮助。</div></div>	<div>扬帆向海的博客</div> <div>8万+</div>
<div><div></div><div>•</div></div> <div>Ordering Tasks UVA - 10305---拓扑排序(dfs) - Nicola..._CSDN博客</div>	1-4
<div><div></div><div>•</div></div> <div>拓扑排序(算法 非可执行程序) - weixin_34270865的博客 - CSDN博客</div>	1-10
<div><div></div><div>大学四年，因为知道这些开发工具，我成为别人眼中的大神 亲测全部都很好用，自己开发都离不开的软件，如果你是学生可以看看，提前熟悉起来。...</div></div>	<div>敖丙</div> <div>6万+</div>
<div><div></div><div>•</div></div> <div>拓扑排序dfs - qq_40688707的博客 - CSDN博客</div>	10-19
<div><div></div><div>拓扑排序 (topological sorting) AOV网络 在有向图中，用顶点表示活动，用有向边表示活动Vi必须先于活动Vj进行。这种</div></div>	<div>一只码畜</div> <div>1074</div>
<div><div></div><div>分别使用Kahn和DFS实现拓扑排序 1，先了解什么是偏序？偏序就是图中存在无先后顺序的顶点对。全序即找不到这样的顶点</div></div>	<div>记录每一个小阶段的学习心得，持之以恒！</div> <div>671</div>
<div><div></div><div>拓扑排序的原理及其实现</div></div>	<div>不忘初心，好好沉淀</div> <div>19万+</div>
<div><div></div><div>B 站上有哪些很好的学习资源？ 哇说起B站，在小九眼里就是宝藏般的存在，放年假宅在家时一天刷6、7个小时不在话下，更别</div></div>	<div>九章算法的博客</div> <div>19万+</div>
<div><div></div><div>将一个接口响应时间从2s优化到 200ms以内的一个案例 一、背景在开发联调阶段发现一个接口的响应时间特别长，经常超时，囧...本文讲讲是如何定位</div></div>	<div>明明如月的专栏</div> <div>1万+</div>
<div><div></div><div>《MySQL 性能优化》之理解 MySQL 体系结构 本文介绍 MySQL 的体系结构，包括物理结构、逻辑结构以及插件式存储引擎。</div></div>	<div>Tony.Dong的专栏</div> <div>5万+</div>
<div><div></div><div>花了20分钟，给女朋友们写了一个web版群聊程序 参考博客[1]https://www.byteslounge.com/tutorials/java-ee-html5-websocket-example</div></div>	<div></div> <div>34万+</div>
<div><div></div><div>Uva 10305 Ordering Tasks （用dfs 实现拓扑排序） 原題：点击右边的--&amp;gt;原題題意：给你一个n，表示有n个任务，给你m组两个数，表示两个</div></div>	<div>PK_PK的博客</div> <div>207</div>

python爬取百部电影数据，我分析出了一个残酷的真相 2019年就这么匆匆过去了，就在前几天国家电影局发布了2019年中国电影市场数据，数据显示	Leo的博客 5万+
拓扑排序 实现算法之二dfs实现 关键路径的实现基础 如果想深入学习拓扑排序与关键路径（最长路径）的关系，建议学习博	aiwo1376301646的博客 135
曾经优秀的人，怎么就突然不优秀了。 职场上有很多辛酸事，很多合伙人出局的故事，很多技术骨干被裁员的故事。说来模板都类	caoz的梦呓 7万+
图基本算法 拓扑排序（基于dfs） 拓扑排序，是对有向无回路图进行排序，以期找到一个线性序列，这个线性序列在生活正	weixin_33753845的博客 28
超全Python图像处理讲解（多图预警） 文章目录Pillow模块讲解一、Image模块1.1、打开图片和显示图片1.2、创建一个简单的图	ZackSock的博客 4万+
“金三银四”，敢不敢“试”？ 临近3月份，到了“金三银四”换工作的高峰期，往年可能会3、4月份，今年特殊，多方渠道了解	铭毅天下（公众号同名） 1万+
和黑客斗争的 6 天！ 互联网公司工作，很难避免不和黑客们打交道，我呆过的两家互联网公司，几乎每月每天每分	纯洁的微笑 10万+
[数据结构]Graph之拓扑排序BFS&DFS实现 什么是拓扑排序在这里就不说了。直接讲讲拓扑排序的DFS和BFS实现算法。一、DFS实现拓扑	yuchenchenyi的博客 1499
讲一个程序员如何副业月赚三万的真实故事 loonggg读完需要3分钟速读仅需 1 分钟大家好，我是你们的校长。我之前讲过，这年头，只要	非著名程序员 3万+
如何优雅的替换掉代码中的ifelse TODO	薛定谔的雄猫的博客 1万+
图基本算法 拓扑排序（基于邻接表的dfs实现） 拓扑排序，是对有向无回路图（顶点活动网络AOV网）进行排序，以期找到一个线性序列，这	Tham 在思索中前行！ 2674
谁说程序员不懂浪漫——我的C语言结婚请柬（附源码） 前言：但行好事，莫问前程——《增广贤文》从上学起开始学C++，后面也做过H5，现在	启舰 2万+
Ordering Tasks UVA - 10305----拓扑排序（dfs） 题目描述约翰有许多工作要做。不幸的是，这个任务并不是独立的，如果其他任务已经执行一	Nicolas的博客 67
总结了 150 余个神奇网站，你不来瞅瞅吗？ 原博客再更新，可能就没了，之后将持续更新本篇博客。	爪白白的个人博客 3万+
学Python后到底能干什么？网友：我太难了 感觉全世界营销文都在推Python，但是找不到工作的话，又有哪个机构会站出来给我推荐工	CSDN资讯 2万+
Auto.JS实现抖音，刷宝等刷视频app,自动点赞，自动滑屏，自动切换视频 Auto.JS实现抖音，刷宝等刷视频app,自动点赞，自动滑屏，自动切换视频代码如下auto();var	qq_40618664的博客 2万+
大学四年自学走来，这些私藏的实用工具/学习网站我贡献出来了 大学四年，看课本是不可能一直看课本的了，对于学习，特别是自学，善于搜索网上的一些资	帅地 78万+

学Python后到底能干什么？网友：我太难了 感觉全世界营销文都在推Python，但是找不到工作的话，又有哪个机构会站出来给我推荐工	CSDN学院 3万+
在中国程序员是青春饭吗？ 今年，我也32了，为了不给大家误导，咨询了猎头、圈内好友，以及年过35岁的几位老程序	启舰 47万+
Java校招入职华为，半年后我跑路了 何来我，一个双非本科弟弟，有幸在 19 届的秋招中得到前东家华为（以下简称 hw）的赏识，	Java成神之路 25万+
在三线城市工作爽吗？ 我是一名程序员，从正值青春年华的 24 岁回到三线城市洛阳工作，至今已经 6 年有余。一不	沉默王二 17万+
这些插件太强了，Chrome 必装！尤其程序员！ 推荐 10 款我自己珍藏的 Chrome 浏览器插件	沉默王二 9万+
@程序员：GitHub这个项目快薅羊毛 今天下午在朋友圈看到很多人都在发github的羊毛，一时没明白是怎么回事。后来上百度搜索	dotNet全栈开发 7万+
用python打开女同学的电脑摄像头，并把图像传回qq邮箱 前言:如何悄悄的打开朋友的摄像头，看看她最近过的怎么样，嘿嘿！这次让我带你们来实现这	python_LC_nohtyp的博客 3774
技术大佬：我去，你写的 switch 语句也太老土了吧 昨天早上通过远程的方式 review 了两名新来同事的代码，大部分代码都写得很漂亮，严谨的同	沉默王二 10万+
Linux面试题（2020最新版） 文章目录Linux 概述什么是LinuxUnix和Linux有什么区别？什么是 Linux 内核？Linux的基本组件	ThinkWon的博客 9万+
刚回应！删库报复！一行代码蒸发数10亿！ 年后复工大戏，又增加一出：删库跑路！此举直接给公司带来数10亿的市值蒸发损失，并引发	CSDN学院 1万+
副业收入是我做程序媛的3倍，工作外的B面人生是怎样的？ 提到“程序员”，多数人脑海里首先想到的大约是：为人木讷、薪水超高、工作枯燥.....然而，当	九章算法的博客 9万+
MySQL数据库面试题（2020最新版） 文章目录数据库基础知识为什么要使用数据库什么是SQL？什么是MySQL?数据库三大范式是什	ThinkWon的博客 14万+
如果你是老板，你会不会踢了这样的员工？ 有个好朋友ZS，是技术总监，昨天问我：“有一个老下属，跟了我很多年，做事勤勤恳恳，主动	shenjian58的博客 7万+
CSDN惊天BUG：用户直接可以阅读设置了可见性权限的文章 CSDN惊天BUG：用户直接可以阅读设置了可见性权限的文章，讲技术的大平台，不应该啊！你	AT阿宝哥的博客 3022
离职半年了，老东家又发 offer，回不回？ 有小伙伴问松哥这个问题，他在上海某公司，在离职了几个月后，前公司的领导联系到他，希	江南一点雨的专栏 5万+
男生更看重女生的身材脸蛋，还是思想？ 往往，我们看不进去大段大段的逻辑。深刻的哲理，往往短而精悍，一阵见血。问：产品经理	shenjian58的博客 2万+
当HR压你价，说你只值7K，你该怎么回答？ 当HR压你价，说你只值7K时，你可以流畅地回答，记住，是流畅，不能犹豫。礼貌地	qianlia的博客 8127

面试：第十六章：Java中级开发（16k） HashMap底层实现原理，红黑树，B+树，B树的结构原理 Spring的AOP和IOC是什么？它们常	做人还是低调点 3万+
面试阿里p7，被按在地上摩擦，鬼知道我经历了什么？ 面试阿里p7被问到的问题(当时我只知道第一个)：@Conditional是做什么的?@Conditional多个	路人甲Java 5万+
终于懂了TCP和UDP协议区别 终于懂了TCP和UDP协议区别	郑晖的博客 1万+
Python爬虫，高清美图我全都要（彼岸桌面壁纸） 爬取彼岸桌面网站较为简单，用到了requests、lxml、Beautiful Soup4	Zhangguohao666的博客 2万+
编码的未来是“无代码” 作者 Owen Williams译者   明明如月，责编   夕颜出品  CSDN（ID:CSDNnews）二十年前，学	CSDN资讯 3952
面试了一个 31 岁程序员，让我有所触动，30岁以上的程序员该何去何从？ 最近面试了一个31岁8年经验的程序猿，让我有点感慨，大龄程序猿该何去何从。...	程序猿学社的博客 17万+
大三实习生，字节跳动面经分享，已拿Offer 说实话，自己的算法，我一个不会，太难了吧	敖丙 6万+
程序员垃圾简历长什么样？ 已经连续五年参加大厂校招、社招的技术面试工作，简历看的不下于万份这篇文章会用实例告	启舰 11万+
我对视频号的思考和挑战 视频号被灰度到很久了，刚开始为了体验发了两个视频，过了一段时间发现流量还不错，看来	程序新视界 2865
Java岗开发3年，公司临时抽查 <b>算法</b> ，离职后这几题我记一辈子 前几天我们公司做了一件蠢事，非常非常愚蠢的事情。我原以为从学校出来之后，除了找工作	EnjoyEDU的博客 3万+
博主在阿里笔试中拿了0分，竟是因为分不清楚 Java 输入类 nextLine 与 next 两个... 前言以前做算法题，都是实现一个方法，需要的参数会在方法参数中直接给出，而且需要的返	Geffin的博客 7万+
Mysql中的三类锁，你知道吗？ 导读正所谓有人(锁)的地方就有江湖(事务)，人在江湖飘，怎能一无所知？今天不聊江湖，来细	码猿技术专栏 2307
面试官：你连SSO都不懂，就别来面试了 大厂竟然要我SSO，卧槽。	3y 2万+
终于，月薪过5万了！ 来看几个问题想不想月薪超过5万？想不想进入公司架构组？想不想成为项目组的负责人？想不	路人甲Java 4万+
我说我懂多线程，面试官立马给我发了offer 不小心拿了几个offer，有点烦	3y 2万+
自从喜欢上了B站这12个UP主，我越来越觉得自己是个废柴了！ 不怕告诉你，我自从喜欢上了这12个UP主，哔哩哔哩成为了我手机上最耗电的软件，几乎每天	编码之外的技术博客 5万+
代码注释如此沙雕，会玩还是你们程序员！ 某站后端代码被“开源”，同时刷遍全网的，还有代码里的那些神注释。我们这才知道，原来程序	九章算法的博客 9352

	Java	C语言	Python	C++	C#	Visual Basic .NET	JavaScript	PHP	SQL	Go语言	R语言	Assembly
	language	Swift	Ruby	MATLAB	PL/SQL	Perl	Visual Basic	Objective-C	Delphi/Object Pascal	Unity3D		
©2019 CSDN 皮肤主题: 大白 设计师: CSDN官方博客												