玩儿转图论算法

liuyubobobo

哈密尔顿回路

欧拉回路

哈密尔顿路径

欧拉路径

哈密尔顿回路和哈密尔顿路径

liuyubobobo

哈密尔顿回路

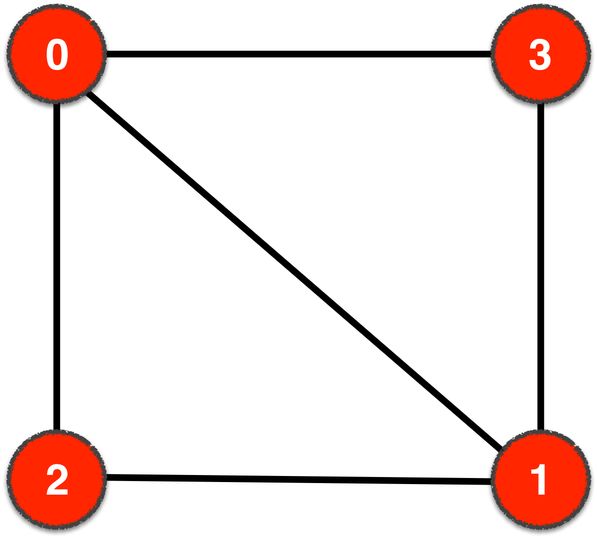
liuyubobobo

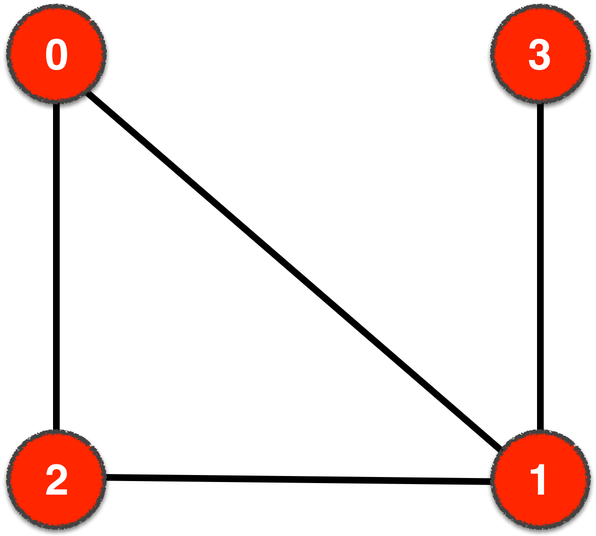
哈密尔顿回路

从一个点出发，沿着边行走，经过每个顶点恰好一次，之后再回到出发点

回到出发点

经过每个顶点恰好一次



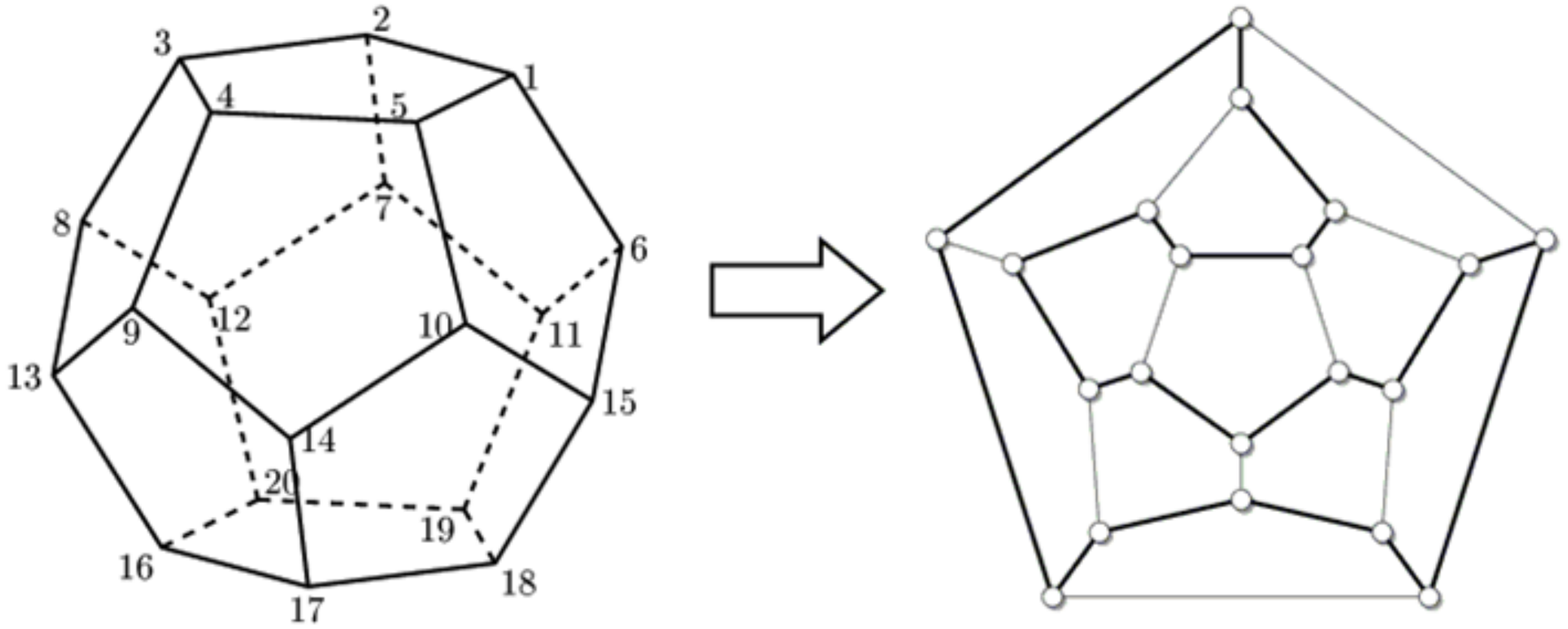


哈密尔顿回路

1859年，爱尔兰数学家哈密尔顿（Hamilton）提出下列周游世界的游戏：

在正十二面体的二十个顶点上依次标记伦敦、巴黎、莫斯科等世界著名大城市，正十二面体的棱表示连接这些城市的路线。试问能否在图中做一次旅行，从顶点到顶点，沿着边行走，经过每个城市恰好一次之后再回到出发点。这就是著名的哈密尔顿问题

哈密尔顿回路



哈密尔顿回路

1859年，爱尔兰数学家哈密尔顿（Hamilton）提出下列周游世界的游戏。

一个半世纪过去了，这个问题即一个图是否为哈密尔顿图的判定问题至今悬而未决。

数学上：找不到充分必要条件

哈密尔顿回路

旅行推销员问题（Travelling Salesman Problem, TSP）

给定一系列城市和每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路。

带权图，完全图

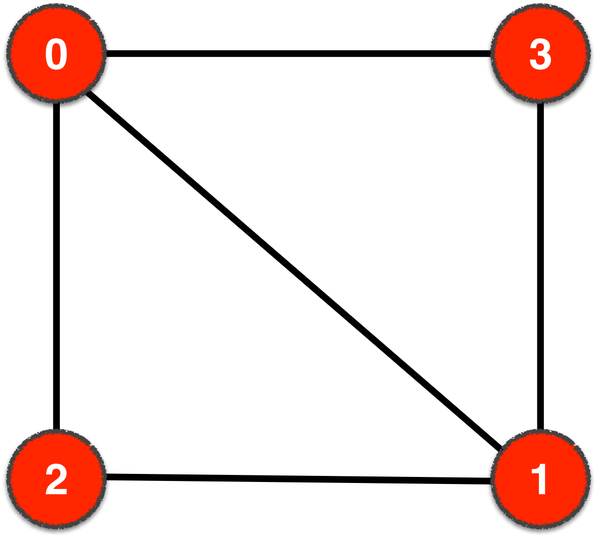
NP 难问题

求解哈密尔顿回路

liuyubobobo

哈密尔顿回路

暴力求解。



0 - 1 - 2 - 3

0 - 1 - 3 - 2

0 - 2 - 1 - 3

0 - 2 - 3 - 1

0 - 3 - 1 - 2

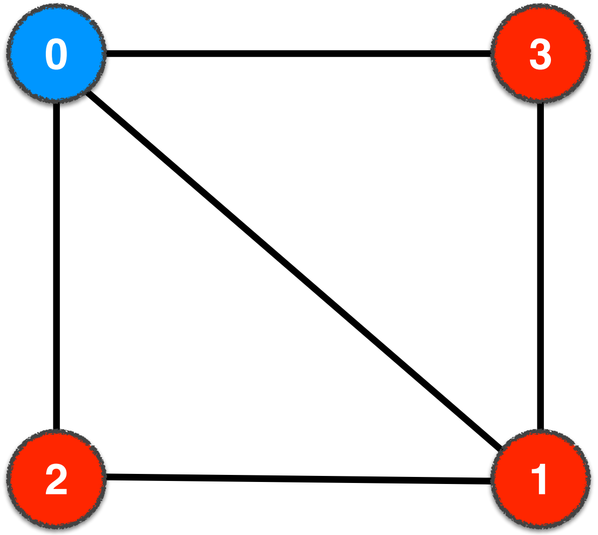
0 - 3 - 2 - 1

… …

回溯

哈密尔顿回路

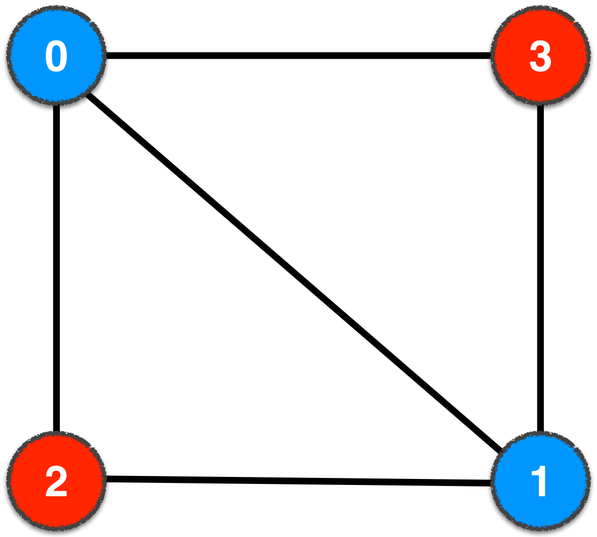
暴力求解。



0

哈密尔顿回路

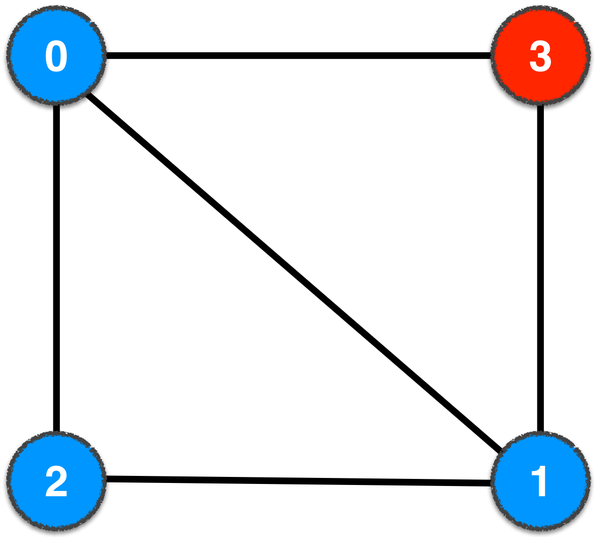
暴力求解。



0 → 1

哈密尔顿回路

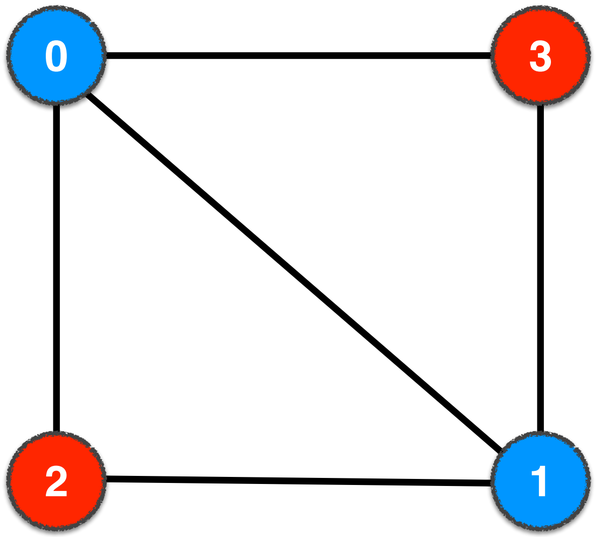
暴力求解。



0 → 1 → 2

哈密尔顿回路

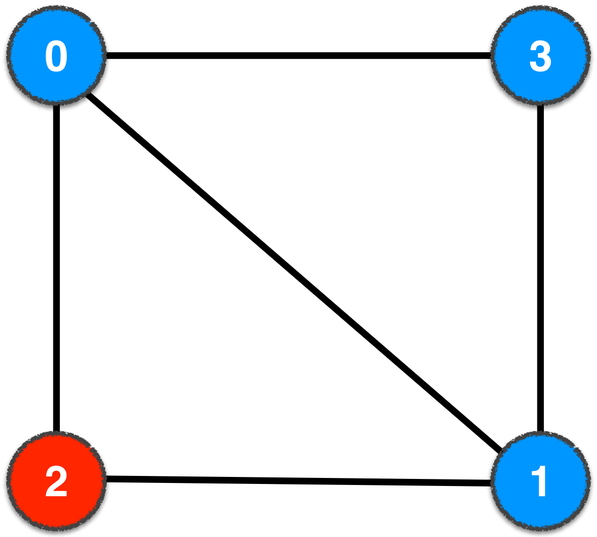
暴力求解。



0 → 1 → 2

哈密尔顿回路

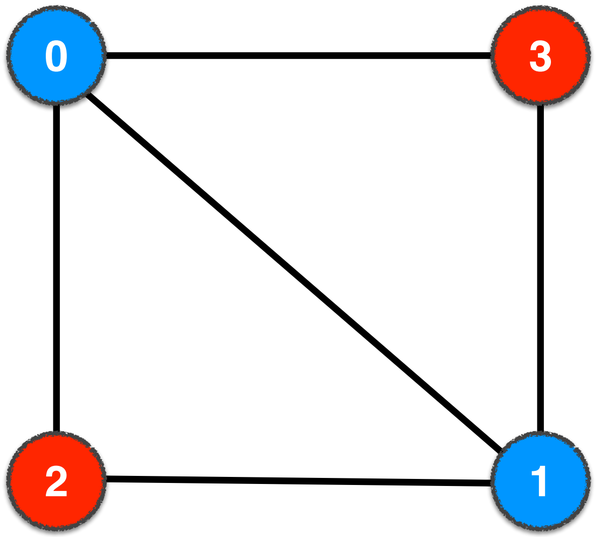
暴力求解。



0 → 1 → 3

哈密尔顿回路

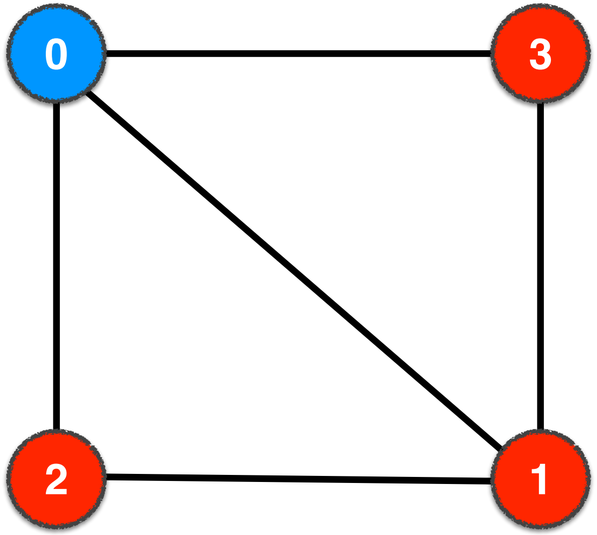
暴力求解。



0 → 1 → 3

哈密尔顿回路

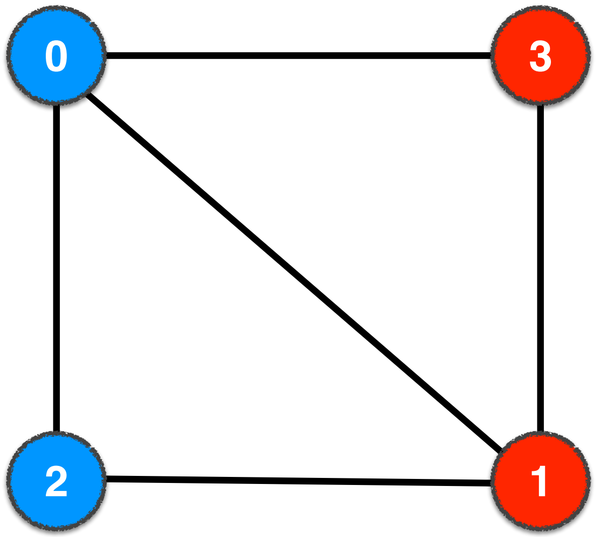
暴力求解。



0 → 1

哈密尔顿回路

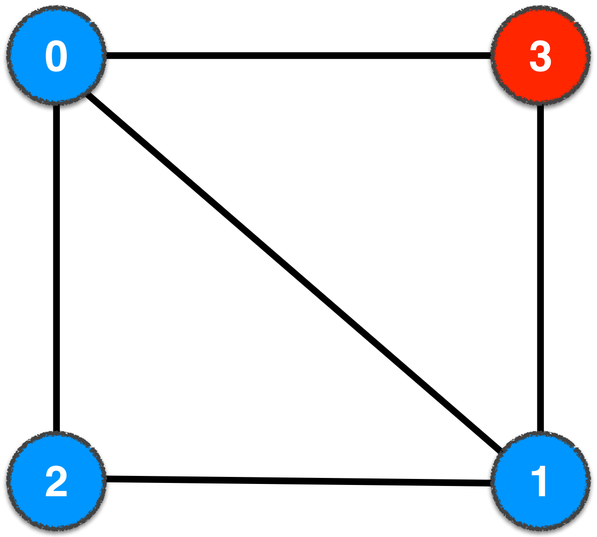
暴力求解。



0 → 2

哈密尔顿回路

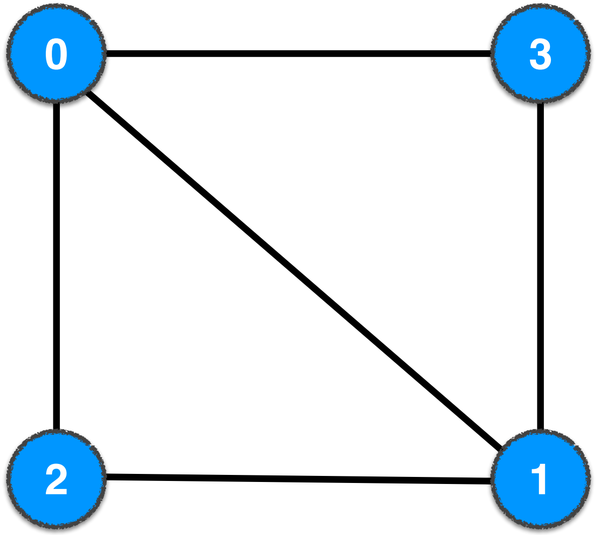
暴力求解。



0 → 2 → 1

哈密尔顿回路

暴力求解。



0 → 2 → 1 → 3

回溯算法

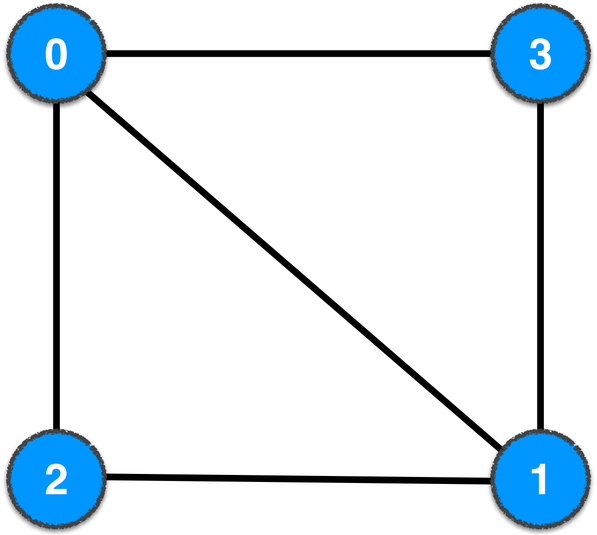
哈密尔顿回路

回溯算法


/ 实战 / 玩转算法面试— Leetcode真题分门别类讲解
收藏
玩转算法面试 从真题到思维全面提升算法思维
为了面试，更为了提升你的算法思维
¥ 266.00
花呗付款
京东白条
难度 中级 · 时长 18小时10分钟 · 学习人数 4257 · 好评度 100%
进入课程

哈密尔顿回路

暴力求解。



0 → 2 → 1 → 3

只从一个顶点开始就好

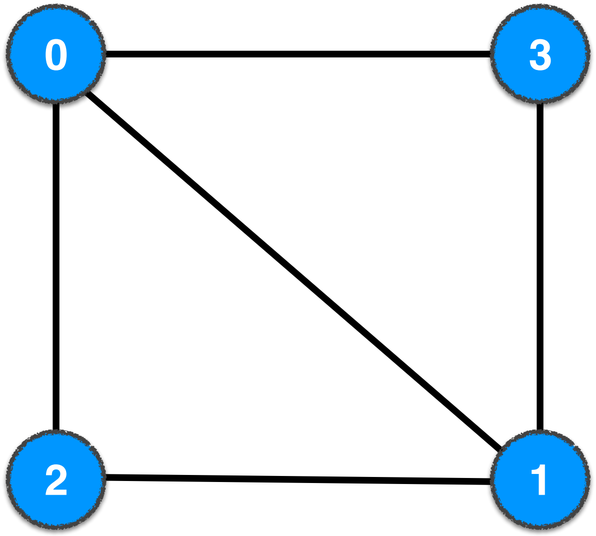
O(n!)

实现哈密尔顿回路算法

liuyubobobo

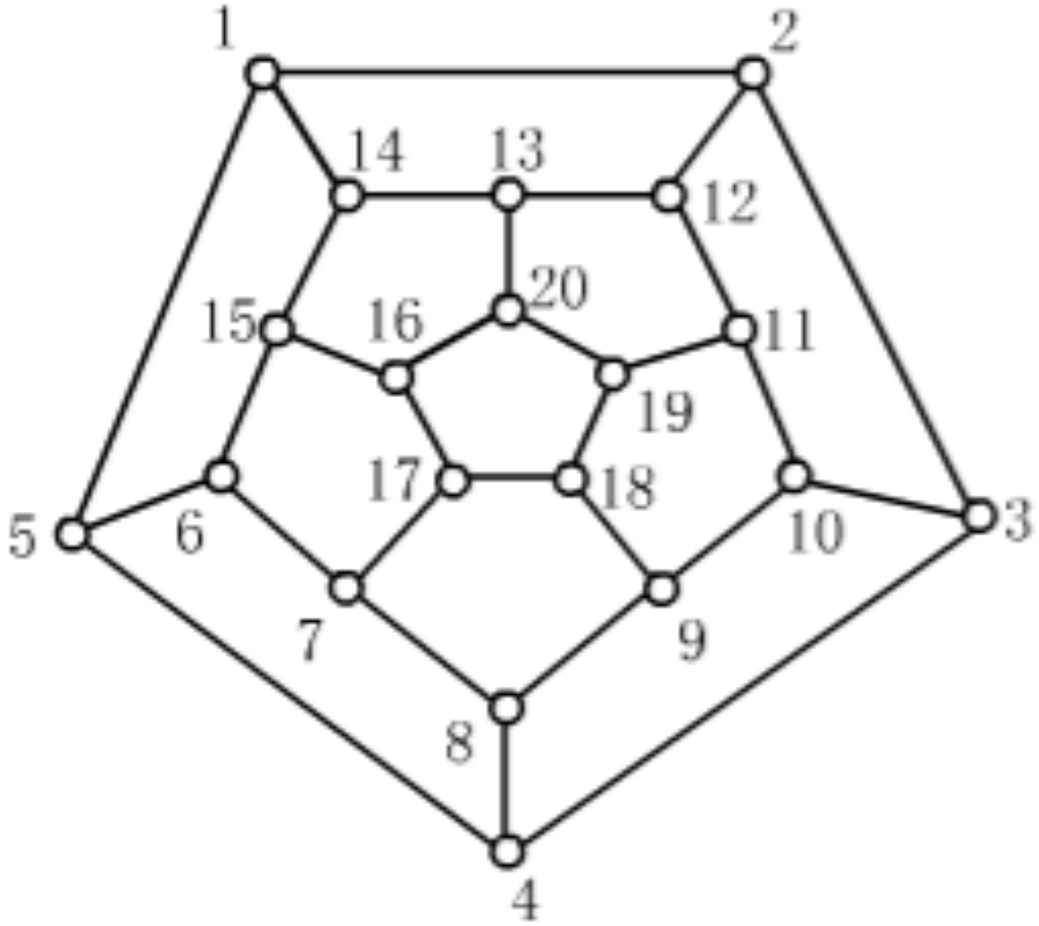
编程实践：实现哈密尔顿回路算法

哈密尔顿回路



0 → 2 → 1 → 3

实现哈密尔顿回路算法



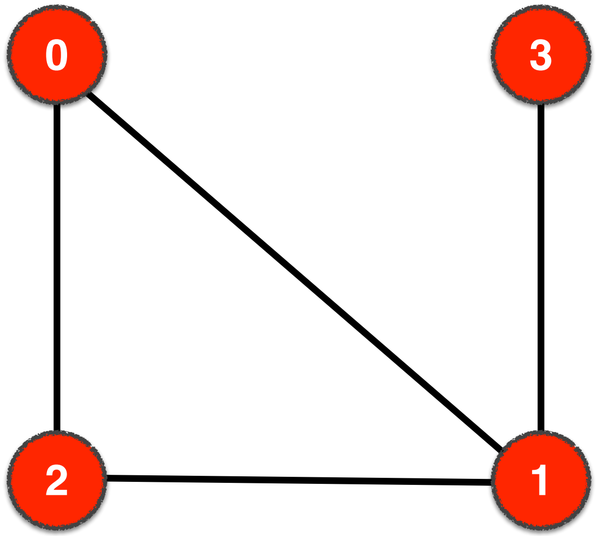
哈密尔顿回路算法的一个优化

liuyubobobo

编程实践：哈密尔顿回路算法的一个优化

哈密尔顿路径

起始点很重要



从0开始走，有哈密尔顿路径

从1开始走，没有哈密尔顿路径

递归终止条件?

哈密尔顿路径算法

liuyubobobo

文字：哈密尔顿路径算法

Leetcode 上的哈密尔顿路径问题

liuyubobobo

Leetcode 上的哈密尔顿路径问题

Leetcode 980： 不同路径 III

编程实践：Leetcode 980

状态压缩

liuyubobobo

状态压缩

二维坐标用一个数字表示

两个桶的水量用一个两位数表示

状态压缩

visited 数组可以使用一个数来表示

visited	0	1	2	3
	true	false	true	false

想成一个二进制数：

1 0 1 0 → 0b0101 → 5

状态压缩

visited 数组可以使用一个数来表示

visited	0	1	2	3
	true	false	true	false

想成一个二进制数：

1 0 1 0 → 0b0101 → 5

状态压缩

十进制　　5　　二进制　　0b0101

看第0位是否是1　　5 & 1 == 0?

```
  0b0101
& 0b0001
--------
  0b0001
```

看第1位是否是1　　5 & 2 == 0?

```
  0b0101
& 0b0010
--------
  0b0000
```

看第2位是否是1　　5 & 4 == 0?

看第3位是否是1　　5 & 8 == 0?

状态压缩

十进制　　5

看第0位是否是1　　5 & 1 == 0?

看第1位是否是1　　5 & 2 == 0?

看第2位是否是1　　5 & 4 == 0?

看第3位是否是1　　5 & 8 == 0?

假设某个状态visited

看其第 i 位是否为1?

visited & (2^i) == 0?

visited & (1<<i) == 0?

状态压缩

十进制　　5　　二进制　　0b0101

将第1位设为1　　5 + 2

```
  0b0101
+ 0b0010
--------
  0b0111
```

将第3位设为1　　5 + 8

```
  0b0101
+ 0b1000
--------
  0b1101
```

如果第i位为0，设为1：　　visited + (1<<i)

状态压缩

十进制　　5　　二进制　　0b0101

将第0位设为0　　5 - 1

```
  0b0101
- 0b0001
--------
  0b0100
```

将第2位设为0　　5 - 4

```
  0b0101
- 0b0100
--------
  0b0001
```

如果第i位为1，设为0：　　visited - (1<<i)

状态压缩

看其第 i 位是否为1?	visited & (1<<i) == 0?
如果第i位为0，设为1：	visited + (1<<i)
如果第i位为1，设为0：	visited - (1<<i)

状态压缩

看其第 i 位是否为1?　　visited & (1<<i) == 0?

如果第i位为0，设为1：　visited + (1<<i)

如果第i位为1，设为0：　visited - (1<<i)

```
private boolean dfs(int v, int parent, int left){

    visited[v] = true;
    pre[v] = parent;
    left --;
    if(left == 0 && G.hasEdge(v, 0)){
        end = v;
        return true;
    }

    for(int w: G.adj(v))
        if(!visited[w]){
            if(dfs(w, v, left)) return true;
        }

    visited[v] = false;
    return false;
}
```

状态压缩

位运算

基于状态压缩的实现

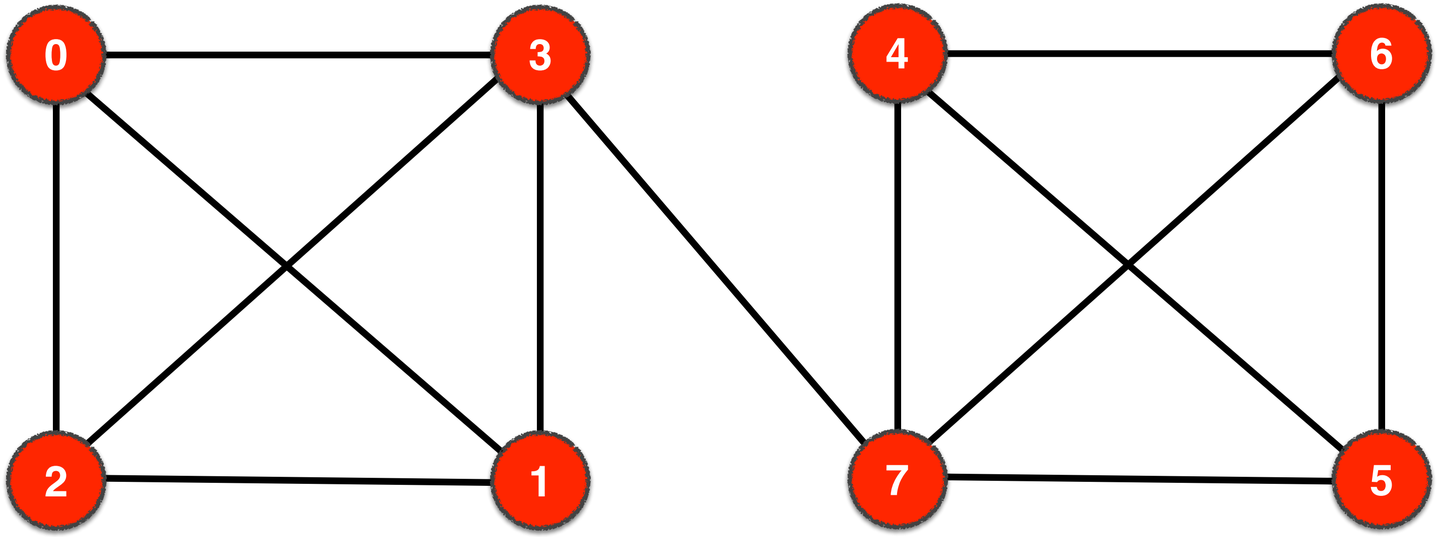
liuyubobobo

编程实践：基于状态压缩的实现

记忆化搜索

liuyubobobo

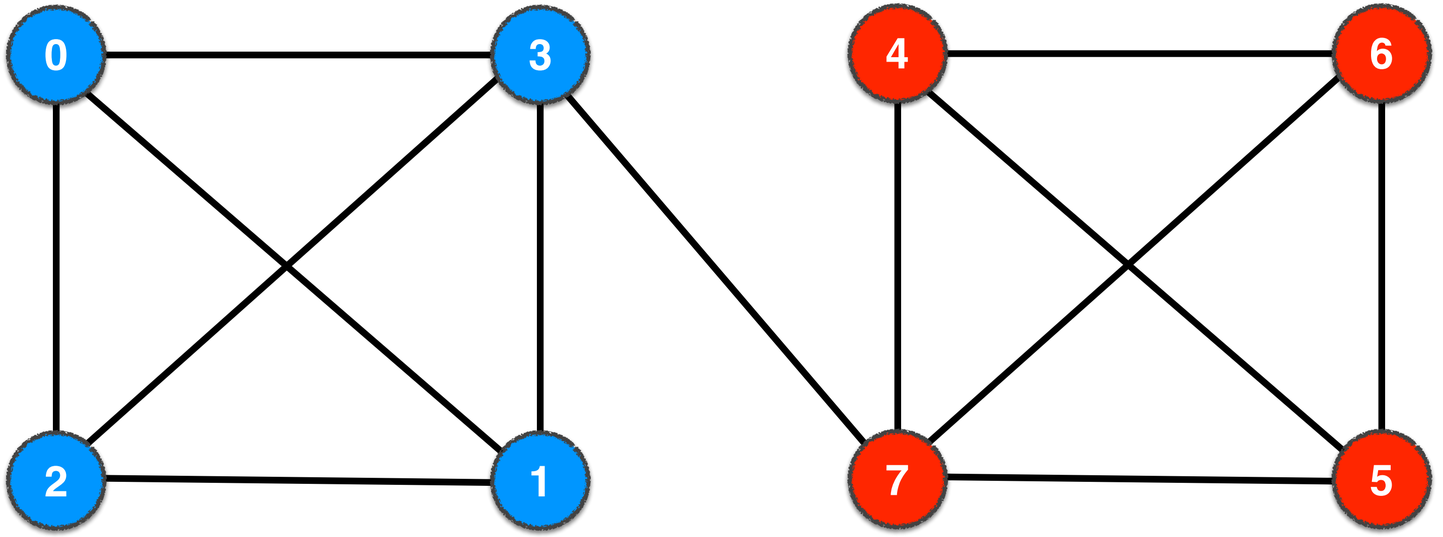
记忆化搜索



0 → 1 → 2 → 3 → ....

0 → 2 → 1 → 3 → ....

记忆化搜索



0 → 1 → 2 → 3 → ····

0 → 2 → 1 → 3 → ····

dfs(visited, v)

0b00001111 3

记忆化搜索

dfs(visited, v)

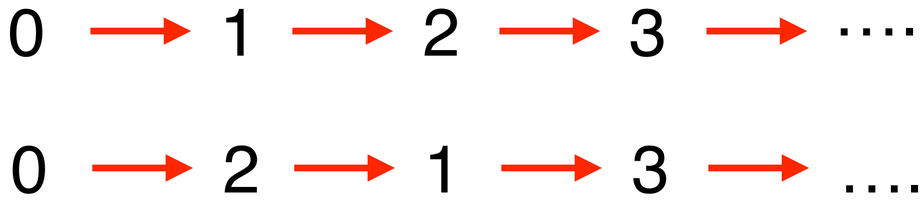
0b00001111　　3

memo[1<<G.V()][G.V()]

查看 memo[visited][v] 之前是否计算过?

如果计算过，直接返回之前计算的结果。

否则，计算，将结果存在 memo[visited][v] 中



O(n*2^n)　　O(n!)

编程实践：记忆化搜索

本章小结

liuyubobobo

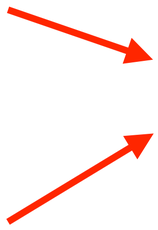
本章小结

哈密尔顿回路和哈密尔顿路径

回溯法

状态压缩

记忆化搜索



面试不考

本章小结

回溯算法，记忆化搜索，动态规划

本章小结

下一章：欧拉回路，欧拉路径

大家加油！

欢迎大家关注我的个人公众号：是不是很酷

坚持有质量的技术原创

用技术人的视角看世界

「是不是很酷」

玩儿转图论算法

liuyubobobo